Contents

1 Question 1

Prior to any interaction with the environment, what is the probability of the target being in a given cell?

Answer:

In this situation, where the target could be located anywhere in the grid of density dim*dim, its probabililty of being in a particular cell is :

$$P = \frac{1}{dim * dim} \tag{1}$$

2 Question 2

Let Pi,j (t) be the probability that cell (i, j) contains the target, given the observations collected up to time t. At time t + 1, suppose you learn new information about cell (x, y). Depending on what information you learn, the probability for each cell needs to be updated. What should the new Pi,j (t + 1) be for each cell (i, j) under the following circumstances:

- **At time t + 1 you attempt to enter (x, y) and find it is blocked?**

$$P(\text{examine fails/hill}) = 0.5 \tag{2}$$

$$P(\text{examine fails/forest}) = 0.8 \tag{3}$$

$$P(\text{examine succeeds/flat}) = 0.8(1 - 0.2) \tag{4}$$

$$P(\text{examine succeeds/hilly}) = 0.5(1 - 0.5) \tag{5}$$

$$P(\text{ examine succeeds/forest}) = 0.2(1 - 0.8) \tag{6}$$

Pij(t+1) is P(cell contains Target / x,y is blocked)=

P(x,y is blocked / cell contains Target)*P(cell contains Target)/P (x,y is blocked)

P(x,y is blocked) =
P(x,y is blocked) P(cell 1 contains target) + P(x,y is blocked) P(cell 2 contains target) +.....+ P(x,y is blocked) P(cell n2 contains target)

Here,
cell 1 = (0,0)
cell n2= (n,n)

P(x,y is blocked) =
P(x,y is blocked / cell1 contains target)*P(cell1 contains target)
+
P(x,y is blocked / cell2 contains target)*P(cell2 contains target)
+.....
+
P(x,y is blocked / cell (x,y) contains target)

P(cell (x,y) contains target) +....+ P(x,y is blocked / cell n2 contains target)

P(cell n2 contains target)

We assume that if a cell i contains the target then the probability of cell j being blocked will increase. This probability will be:

$$\text{Total undiscovered blocks/Total undiscovered cells} = \alpha \tag{7}$$

This probability will be the same for all cases.

$\alpha = P$(x,y is blocked /cell i contains target)

i = 1,2,3,4,.....,n2

i(x,y)

So,

P(x,y is blocked) =
* P(cell 1 contains the target) + * P(cell 2 contains the target) + ... + 0 *
P(cell x,y contains the target) +.....+ * P(cell n2 contains the target)

At time t :

P(cell 1 contains the target) +P(cell 2 contains the target) +.....+ P(cell x,y contains the target) +.....+ P(cell n2 contains the target) = 1

Therefore,

P(cell 1 contains the target) +P(cell 2 contains the target) +....+ P(cell n2 contains the target)
= 1 - P(cell x,y contains the target)
So,

P(x,y is blocked)=
*[P(cell 1 contains the target) +.....+P(cell n2 contains the target)]

P(x,y is blocked)
= * (1 - P(cell x,y contains the target))

Therefore,

Pij(t+1)
= [P(x,y is blocked — cell i,j contains target)*
P(cell i,j contains target)]/ * (1 - P(cell x,y contains the target))

Pij(t+1)= * P(cell i,j contains target)
/ * (1 - P(cell x,y contains the target))

Pij(t+1) = Pij(t)/ (1-Px,y(t))- equation 2.1a
This is basically a weighted distribution of Px,y(t) (since, Px,y(t+1)= 0) to all other cells such that sum of all Pij(t+1) = 1

All observations till time t will be $P_ij(t)$

- **At time t + 1 you attempt to enter (x, y), find it unblocked, and also learn its terrain type??**

  No change in probability

$$P_ij(t+1) = P_i, j(t) \tag{8}$$

Since at time (t+1), cell x,y was found unblocked and this has no effect on Pij(t+1) of other cells.

- **At time t + 1 you examine cell (x, y) of terrain type flat, and fail to find the target?**

  Here we know that the terrain type is flat, which has a False Negative Rate of 0.2 is:
  P (i,j contains Target — Target was not found in x,y)

  = P (T was not found in x,y — i,j contains T) / P(T was not found in x,y)

  So,
  Two Cases:

  Case1: (i,j)(x,y)

  = 1* P(i,j contains T)/ P(T was not found in x,y)

  Case2: (i,j)=(x,y)

  = FNR* P(i,j contains T)/ P(T was not found in x,y)

  So,

  P(T was not found in x,y) =

  P(T was not found in x,y target in cell 1) + P(T was not found in x,y target in cell 2) +....+P(T was not found in x,y target in cell x,y) +.....+P(T was not found in x,y target in cell n,n)

  P(T was not found in x,y)=

  P(T was not found in x,y — T in 0,0) * P(T in (0,0)) +.....+P(T was not found in x,y — T in x,y) * P(T in (x,y)) +.....+P(T was not found in x,y — T in n,n) * P(T in (n,n))

  Therefore,

  P(T was not found in x,y) =

  1* P(T in (0,0) +.....+ FNR * P(T in (x,y))+.....+ 1* P(T in (n,n))

  So,

Pt+1(x,y)= Pt(x,y)* FNR(terrain) / (Pt(x,y) (FNR(terrain)-1))+1

Here we know that the terrain type is flat, which has a False Negative Rate of 0.2 is:

Pt+1(x,y)= Pt(x,y)* FNR(terrain) / (Pt(x,y) (FNR(terrain)-1))+1

Pt+1(x,y)= Pt(x,y)* 0.2 / (Pt(x,y) (0.2-1))+1

$$P_t + 1(x, y) = \frac{P_t(x, y) * FNR(\text{terrain})}{(Pt(x, y)(FNR(\text{terrain}) - 1)) + 1} \tag{9}$$

$$P_t + 1(x, y) = \frac{P_t(x, y) * 0.2}{P_t(x, y)(0.2 - 1)) + 1} \tag{10}$$

- **At time t + 1 you examine cell (x, y) of terrain type hilly, and fail to find the target?**
  Here we know that the terrain is hilly, which has a False Negative Rate of 0.5 is:

$$P_t + 1(x, y) = \frac{P_t(x, y) * 0.5}{P_t(x, y)(0.5 - 1)) + 1} \tag{11}$$

- **At time t + 1 you examine cell (x, y) of terrain type forest, and fail to find the target?**
  Here we know that the terrain is a forest, which has a False Negative Rate of 0.8 is:

$$P_t + 1(x, y) = \frac{P_t(x, y) * 0.8}{(P_t(x, y)(0.8 - 1)) + 1} \tag{12}$$

- **At time t + 1 you examine cell (x, y) of terrain type forest, and fail to find the target?**

  If at (t+1), we find target at x,y then

$$P_ij(t + 1) = 0 \tag{13}$$

3 Question 3

At time t, with probability Pi,j (t) of cell (i, j) containing the target, what is the probability of finding the target in cell (x, y):
Answer:

$$P(\text{finding T in i,j}) = (1 - FNR) * P_i, j(\text{contains}T)) \tag{14}$$

- **if (x, y) is hilly?**

$$P(\text{finding T in } i, j) = (1 - 0.5) * P_i, j\text{contains T}) \tag{15}$$

- **if (x, y) is flat?**

$$P(\text{finding T in } i, j) = (1 - 0.2) * P_i, j\text{contains T}) \tag{16}$$

- **if (x, y) is forest?**

$$P(\text{finding T in } i, j) = (1 - 0.8) * P_i, j\text{contains T}) \tag{17}$$

- **if (x, y) is has never been visited?**

$$P(\text{finding T in } i, j) = P_i, j\text{contains T}) \tag{18}$$

4 Question 4

4.1 Agent 6 Implementation

We are finding the probability of containing the target in agent 6 which is the same as question 2.

We need to follow below steps in order to develop agent 6:

- **Step 1:** Initialize the belief state of each cell with probability from formula Equation (1). Select the cell with maximum probability and make this cell as the assumed target.

```
def init_dict(self,x = 0,y = 0):
    initProb=1/(self.n*self.n)
    for i in range(self.n):
        for j in range(self.m):
            self.dKprob[i,j]=[initProb,"NA"]

    #Initializing Dictionary with default values, Here "NA" is terrain not defined.
```

- **Step 2:** Here, we start our planning phase in the knowledge gridworld, we will plan our path by calling A-star, taking the cell with max probability as target. Now the path that we get from A-star will be traversed in the final gridworld inside execution phase.
- **Step 3:** Now, we will start our execution phase, agent will start iterating the path that we get from the planning phase. We are updating the terrain in knowledge gridworld while doing the iteration.
- **Step 4:** While iterating the path in execution phase if we get a blocked cell then we will update the belief state of whole grid as mentioned in formula from Question 2 (a) . Now, the agent will start replanning, from the cell before the blocked cell and will repeat the process from step 2. During Iteration we will increase the count for movements by 1.

```
    examination=0
    if len(path)>0:
        count=count+1
        #Initializing default values
        p=Phases(Klst,Flst,101,101,target)
        exploredSet=set()
        trajlst=[]
        bx=0
        by=0
        TargetNotFound=True
        source=(0,0)
        while TargetNotFound:
            # Calculating target with max probability.
            Target=p.TargetwithMaxProb()
            # Iterating from source to target in planning phase
            path=p.Planning(source,Target)

            trajlst.append(path)
            # if target that we assumed is reachable then proceed to execution phase
            if len(path)>0:

                # Iterating path in execution phase till it gets the actual target
                source,TargetNotFound,moves=p.Execution(path)
                movement=movement+moves
                examination=examination+1

        print("Success")
        print(path)
        # Calculating Total Moves
        movements6.append(movement)
        # Calculating Total examinations
        examinations6.append(examination)
```

```
for item in range(len(path)):

    # if we get the blocked cell while iterating planned path
    if len(path)>1 and item!=0:
        x=path[item][0]
        y=path[item][1]
        if self.Flst[x][y]==-1:
            self.Klst[x][y]=-1
            # update whole grid when the cell is blocked
            self.updateKnowledgeBase(x,y,"NA","B")
            compLoop=False
            bcell=path[item-1]
            movement=movement+1
            break

        else:

            #Agent 6
            # if we get the unblocked cell while iterating the path
            #updating terrains without affecting probability
            ubcell=(x,y)
            if (x,y) in self.dKprob:
                initProb=self.dKprob[x,y]
                initProb=self.dKprob[x,y][0]
                movement=movement+1 #Total Moves Counter
                self.dKprob[x,y]=[initProb,self.IdentifyTerrain(self.Flst[x][y])]
```

```
initProb=self.dKprob[x,y][0]
#print("initProbbefore=",initProb)
fnrH=0.5
fnrF=0.2
fnrTF=0.8
if cond=="B":
    sumt=0
    #Blocked cell probability would be 0 and all other grid probabilities will change accordingly
    self.dKprob[x,y]=[0,terrain]
    for i in range(self.n):
        for j in range(self.m):
            sumt=sumt+self.dKprob[i,j][0]

    for i in range(self.n):
        for j in range(self.m):
            if (i,j)!=(x,y):

                probij=self.dKprob[i,j][0]
                terraini=self.dKprob[i,j][1]
                self.dKprob[i,j]=[probij/sumt,terraini]
```

```
x=ubcell[0]
y=ubcell[1]
#Examination of the cell starts
if compLoop==False:# cell was blocked
    return bcell,True,movement
elif ubcell==self.target:#cell was unblocked and find the target

    #Random Generator function was called in order to compare its value with FNR of terrain
    val=self.generateFalseTarget()
    targetFound=self.TargetMeasure(val,self.IdentifyTerrain(self.Flst[x][y]))
    # If random value greater than FNR value of terrain for that particular cell then target found is true
    if targetFound==True:
        return (0,0),False,movement
    else:
        #If random value is lesser than FNR value of that particular cell then target not found
        #whole grid will be updated
        self.updateKnowledgeBase(x,y,self.IdentifyTerrain(self.Flst[x][y]),"UB")
        return ubcell,True,movement
else:
    # cell was unblocked and target not found
    self.updateKnowledgeBase(x,y,self.IdentifyTerrain(self.Flst[x][y]),"UB")
    return ubcell,True,movement#last element
```

- **Step 5:** When the agent reached the assumed target. It will examine and compare with coordinates of the actual target, If that doesn't match then the belief state of the whole gridworld will change according to the formula given in Question 2 (c), (d), (e). We will increase the examination count by 1 and repeat the whole process of replanning from step 2.

```
elif cond=="UB":
    #Unbloced cell with target not found, will update probabilities of whole grid including unblocked cell
    if terrain =="F":
        probxy=initProb*fnrF
    elif terrain =="H":
        probxy=initProb*fnrH
    elif terrain=="TF":
        probxy=initProb*fnrTF
    sumt=0
    for i in range(self.n):
        for j in range(self.m):
            if (i,j)!=(x,y):

                sumt=sumt+self.dKprob[i,j][0]

    probxy=probxy/(sumt+probxy)
    self.dKprob[x,y]=[probxy,terrain]
    sumt=sumt+probxy
    for i in range(self.n):
        for j in range(self.m):
            if (i,j)!=(x,y):
                probij=self.dKprob[i,j][0]
                terraini=self.dKprob[i,j][1]
                self.dKprob[i,j]=[probij/sumt,terraini]
```

- **Step 6:**

  When the agent reached the assumed target. It will examine and compare with coordinates of the actual target, if it matches, then we will call a random generator, which will generate random value between 0 and 1. If generated value is lesser than the false negative rate of terrain present at the assumed target then examination will result in failure to find the target at that particular cell. Belief state of the whole gridworld will change according to the formula given in Question 2 (c), (d), (e). We will increase examination count by 1 and repeat the whole process of replanning from step 2.

```
# Calculating if random generated value is greater than FNR of terrain in order to find the target
def TargetMeasure(self,val,terrain):
    if terrain=="F":
        if val>self.fnrF:
            return True
        else:
            return False
    elif terrain=="H":
        if val>self.fnrH:
            return True
        else:
            return False
    else:
        if val>self.fnrTF:
            return True
        else:
            return False
```

- **Step 7:** When the agent reached the assumed target. It will examine and compare with coordinates of the actual target, if it matches, then we will call a random generator, which will generate random value between 0 and 1. If generated value is greater than the false negative rate of terrain present at the assumed target then examination will result in successfully finding the target . We will increase the examination count by 1 and execution phase will end.

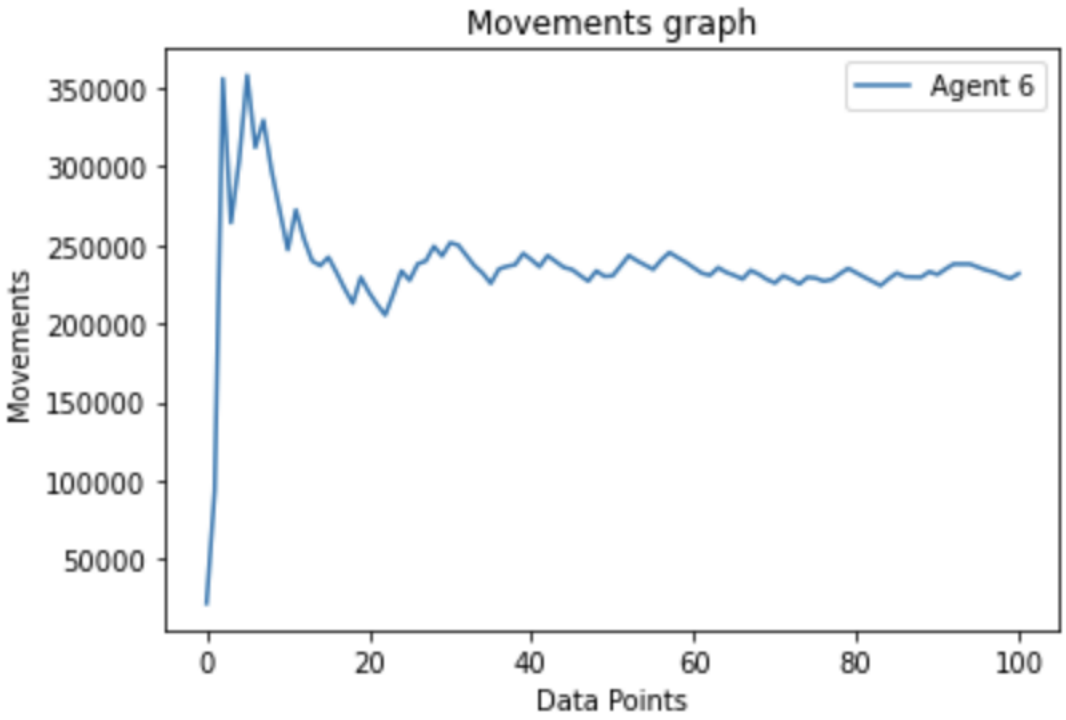

Figure 1: Movements for Agent 6

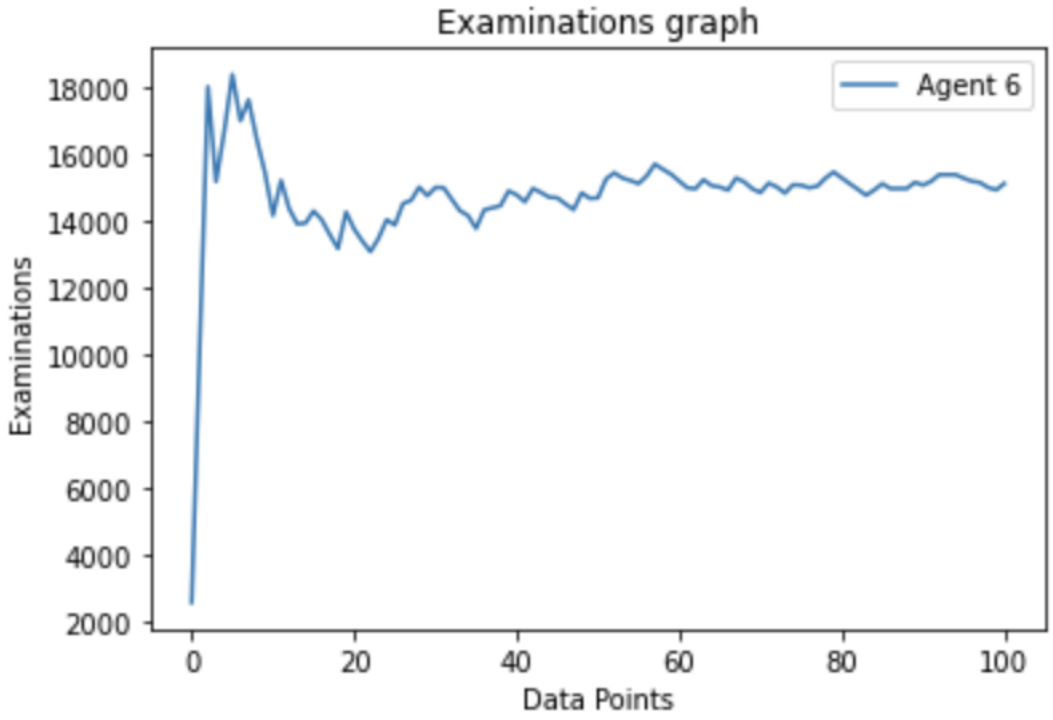

Figure 2: Examination Count for Agent 6

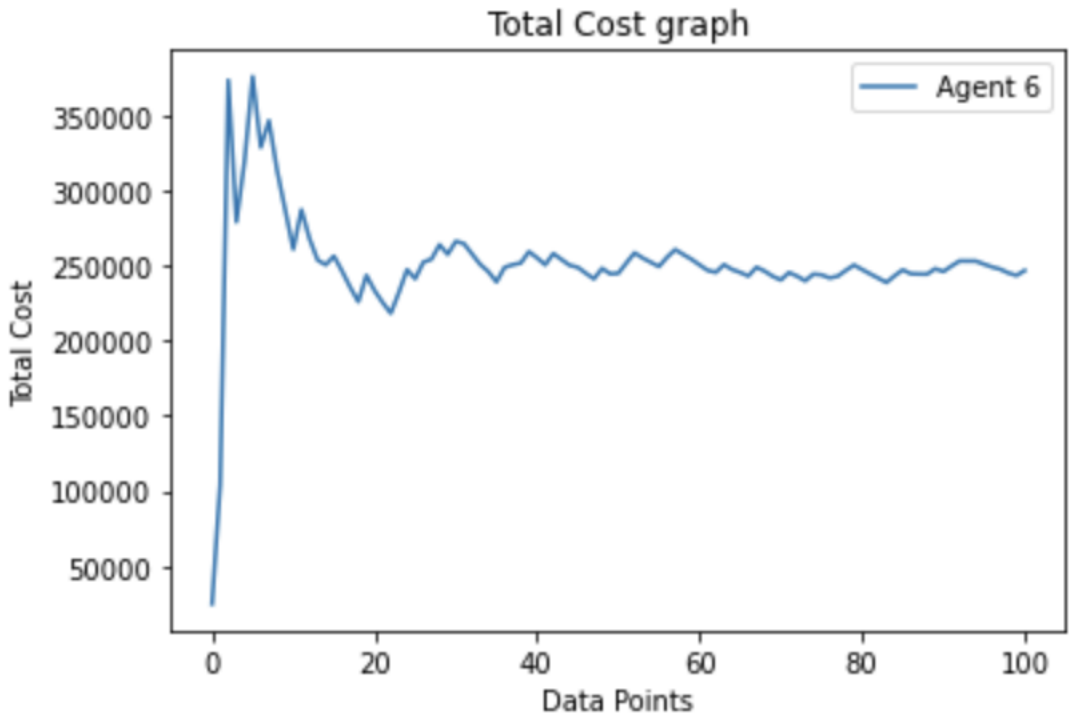

Figure 3: Total Cost= Movements + Examinations

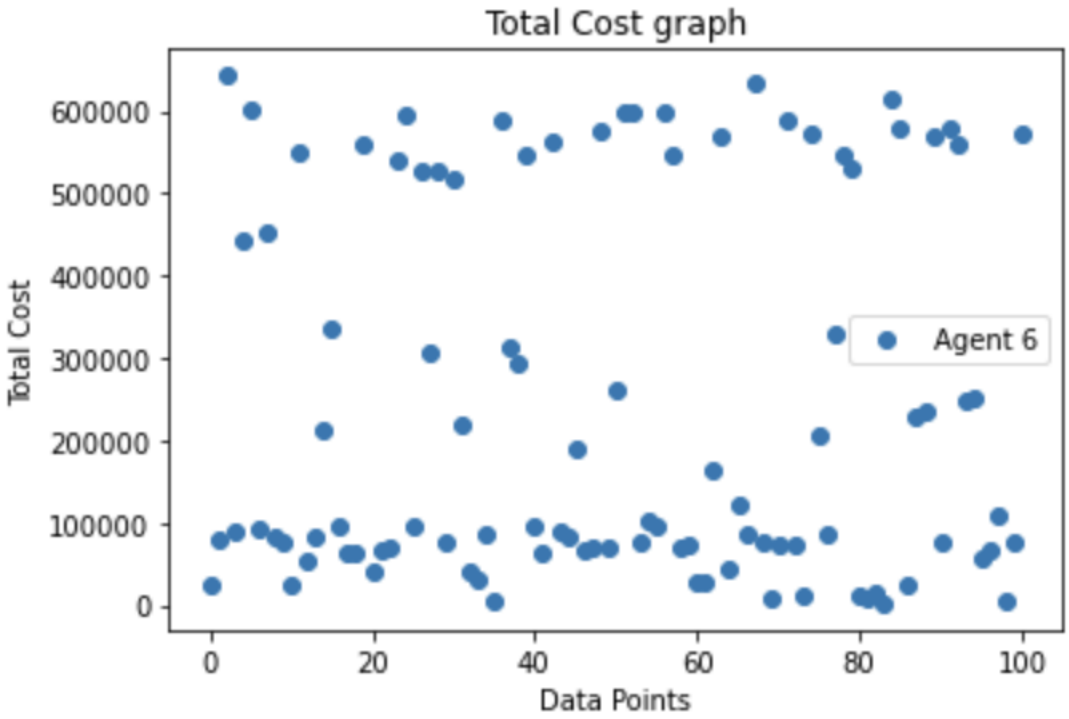

Figure 4: scatter plot for Total Cost

4.2 Agent 7 Implementation

We are finding the probability of successfully finding the target in agent 7 which is the same as question 3 probabilities.

We need to follow below steps in order to develop agent 7:

- **Step 1:** Initialize the belief state of each cell with probability from formula Question 3 (d). Select the cell with maximum probability and make this cell as the assumed target.
- **Step 2:** Here, we start our planning phase in the knowledge gridworld, we will plan our path by calling A-star, taking the cell with max probability as target. Now the path that we get from A-star will be traversed in the final gridworld inside execution phase.
- **Step 3:** Now, we will start our execution phase, agent will start iterating the path that we get from the planning phase. We are updating the terrain in knowledge gridworld as well as the probability of successfully finding the target in that particular cell with formula mentioned in Question 3 (a), (b), (c) while doing the iteration.
- **Step 4:** While iterating the path in execution phase if we get a blocked cell then we will update the belief state of whole grid as mentioned in formula Question 2 for blocked cell . Now, the agent will start replanning, from the cell before the blocked cell and will repeat the process from step 2. During Iteration we will increase the count for movements by 1.

```
x=ubcell[0]
y=ubcell[1]
if compLoop==False:#cell was blocked or we get the cell with max probability other than our assumed target
    if MaxprobChangedFlag==True:

        return ubcell,True,movement
    else:

        return bcell,True,movement

elif ubcell==self.target:#cell was unblocked and find the target
    val=self.generateFalseTarget()
    targetFound=self.TargetMeasure(val,self.IdentifyTerrain(self.Flst[x][y]))
    if targetFound==True:
        return (0,0),False,movement
    else:
        self.updateKnowledgeBase(x,y,self.IdentifyTerrain(self.Flst[x][y]),"UB")
        #Calculating probability of successfully finding the target
        self.SucceedingProbability(x,y)
        return ubcell,True,movement

else:# cell was unblocked and target not found
    self.updateKnowledgeBase(x,y,self.IdentifyTerrain(self.Flst[x][y]),"UB")
    #Calculating probability of successfully finding the target
    self.SucceedingProbability(x,y)
    return ubcell,True,movement#last element
```

- **Step 5:** During Iteration, we are also checking if we get the max probability of finding the target is different from our assumed target. If that happens, we will do the replanning from the cell in which the agent is there to the cell which has max probability of finding the target and repeat the process from step 2

```
                ubcell=(x,y)
                if (x,y) in self.dKprob:
                    initProb=self.dKprob[x,y]
                    initProb=self.dKprob[x,y][0]
                    terrain=self.dKprob[x,y][1]
                    if terrain =="NA":#First time you are discovering the terrain

                        self.dKprob[x,y]=[initProb,self.IdentifyTerrain(self.Flst[x][y])]
                        if item!=(len(path)-1):
                            # We are changing the probability of successfully finding the target
                            # when we observe the terrain
                            self.SucceedingProbability(x,y)

                    else:
                        self.dKprob[x,y]=[initProb,self.IdentifyTerrain(self.Flst[x][y])]
                movement=movement+1

        # we are checking if our target still having maximum probability, if it doesn,t then replaning occurs
                maxed=self.TargetwithMaxProb()
                if maxed!=path[len(path)-1]:
                    MaxprobChangedFlag=True
                    compLoop=False
                    break
```

- **Step 6:** When the agent reached the assumed target. It will examine and compare with coordinates of the actual target, If that doesn't match then the belief state of the whole gridworld will change according to the formula given in Question 3 (a), (b), (c). We will increase the examination count by 1 and repeat the whole process of replanning from step 2.

```
# We use this function in order to find the probability of successfully finding the target and update dictionary
def SucceedingProbability(self,x,y):
    initprob=self.dKprob[x,y][0]
    terrain=self.dKprob[x,y][1]
    newProb=initprob
    if terrain=="H":
        newProb=initprob*(1-self.fnrH)
    elif terrain=="F":
        newProb=initprob*(1-self.fnrF)
    else:
        newProb=initprob*(1-self.fnrTF)

    self.dKprob[x,y]=[newProb,terrain]
```

- **Step 7:** When the agent reached the assumed target. It will examine and compare with coordinates of the actual target, if it matches, then we will call a random generator, which will generate random value between 0 and 1. If generated value is lesser than the false negative rate of terrain present at the assumed target then examination will result in failure to find the target at that particular cell. Belief state of the whole gridworld will change according to the formula given in Question 2 (c), (d), (e) according to terrain. We will increase examination count by 1 and repeat the whole process of replanning from step 2.
- **Step 8:** When the agent reached the assumed target. It will examine and compare with coordinates of the actual target, if it matches, then we will call a random generator, which will generate random value between 0 and 1. If generated value is greater than the false negative rate of terrain present at the assumed target then examination will result in successfully finding the target . We will increase the examination count by 1 and execution phase will end.

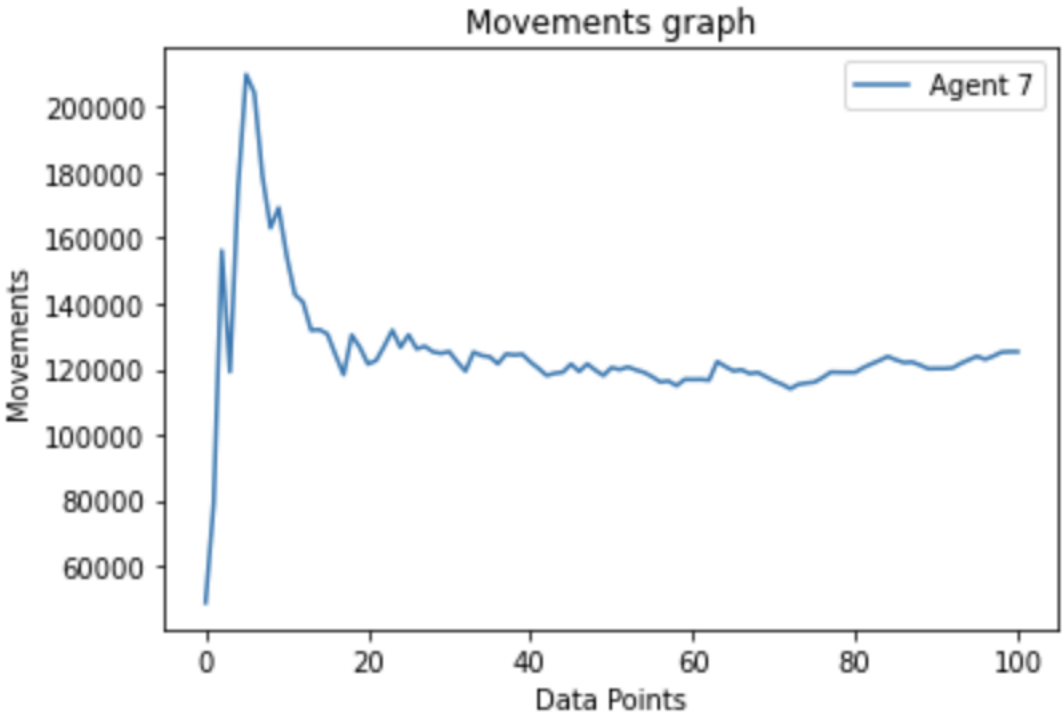

Figure 5: Movements of Agent 7

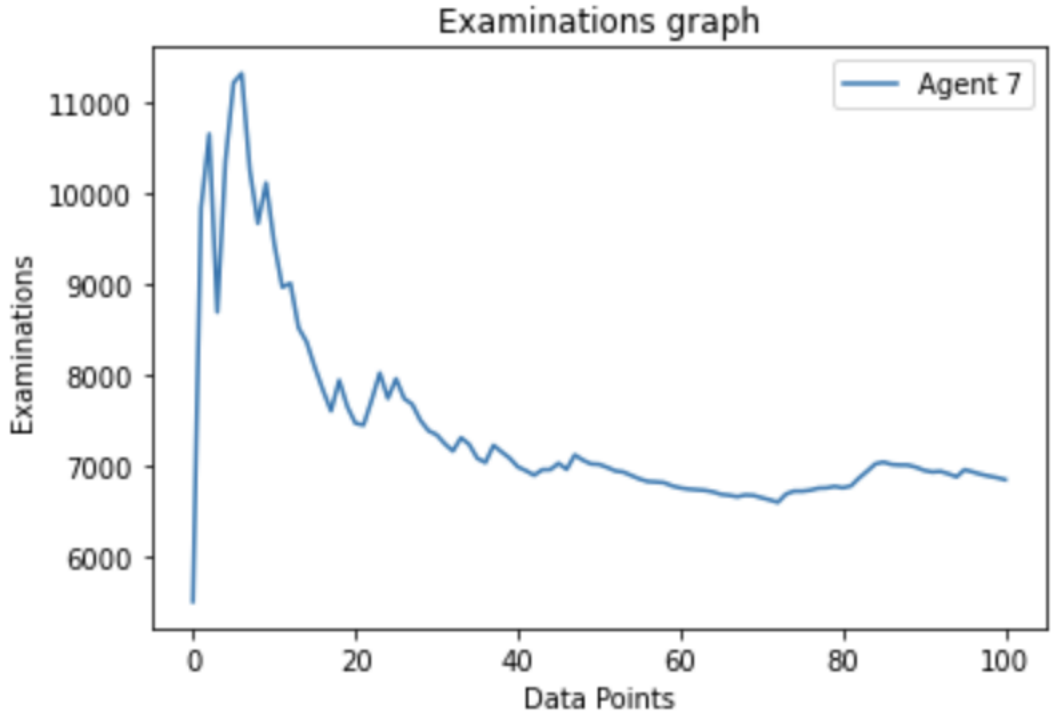

Figure 6: Examination Count of Agent 7

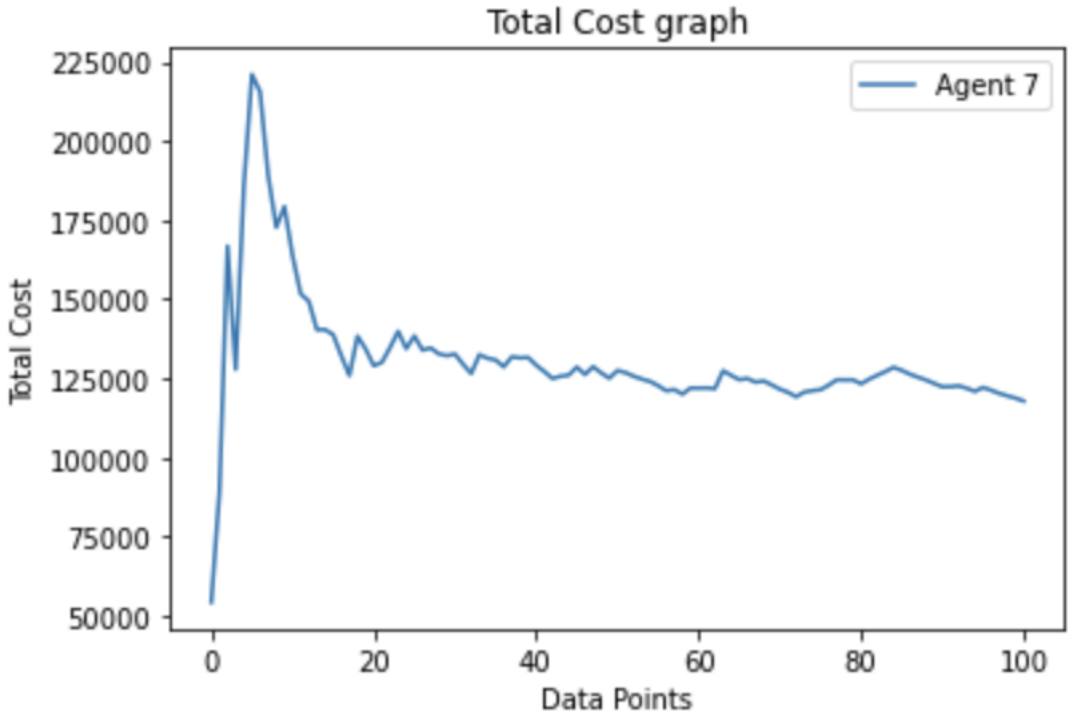

Figure 7: Total Cost= Movements + Examinations

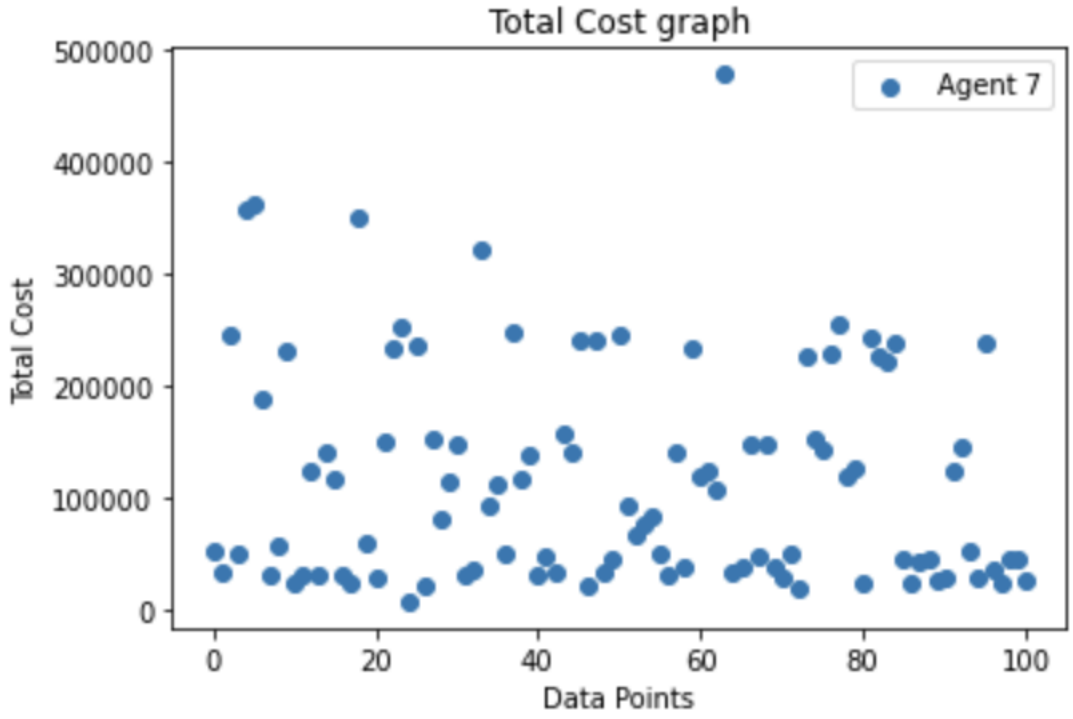

Figure 8: scatter plot for Total Cost

4.3 Comparing Agent 6 and 7

We can clearly see the performance of agent 7 comes out better as compared to agent 6 because while calculating the probability of successfully finding the target, we are changing the probability of a particular cell even while iterating planned path, when we discover the terrain for very first time in a cell. Agent 7 is able to find the target more efficiently as compared to agent 6.

- **Average Movements Comparison:**
  Average movements in cells happened for agent 6: 229400
  Average movements in cells happened for agent 7: 109920

- **Average Examinations Comparison:**
  Average examinations of cells happened for agent 6: 14944
  Average examinations of cells happened for agent 7: 6778

$$TotalCost = Movements + Examinations \tag{19}$$

- **Average Total Cost Comparison:**
  Average total cost (movements+examinations) of cells happened for agent 6: 14944
  Average total cost (movements+examinations) of cells happened for agent 7: 6778

- **Movements per Examination**
  There is another factor through which we can compare agent 8 with agent 7 and 6. From calculating Movements/Examination, we can notice that agent 8 is doing less movements per examination as compared to agent 7 and 6. This means agent 8 will examine closer cells more as compared to other agents. As we are increasing probabilities of closer cells more as compared to cells which are far away.

Analysis

Total cost comes out as a distinctive factor while comparing both of the agents. We can see that average total cost for agent 6 is greater than the average total cost of agent 7.

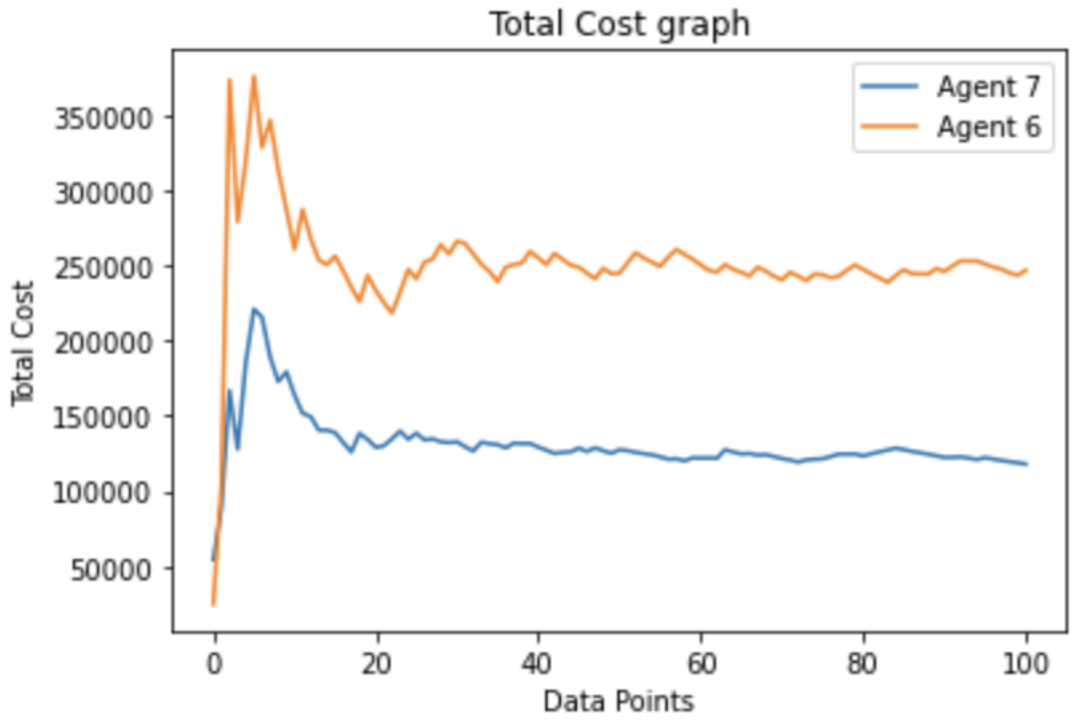

Figure 9: Plot of Total cost for Agent 6 and 7 comparison

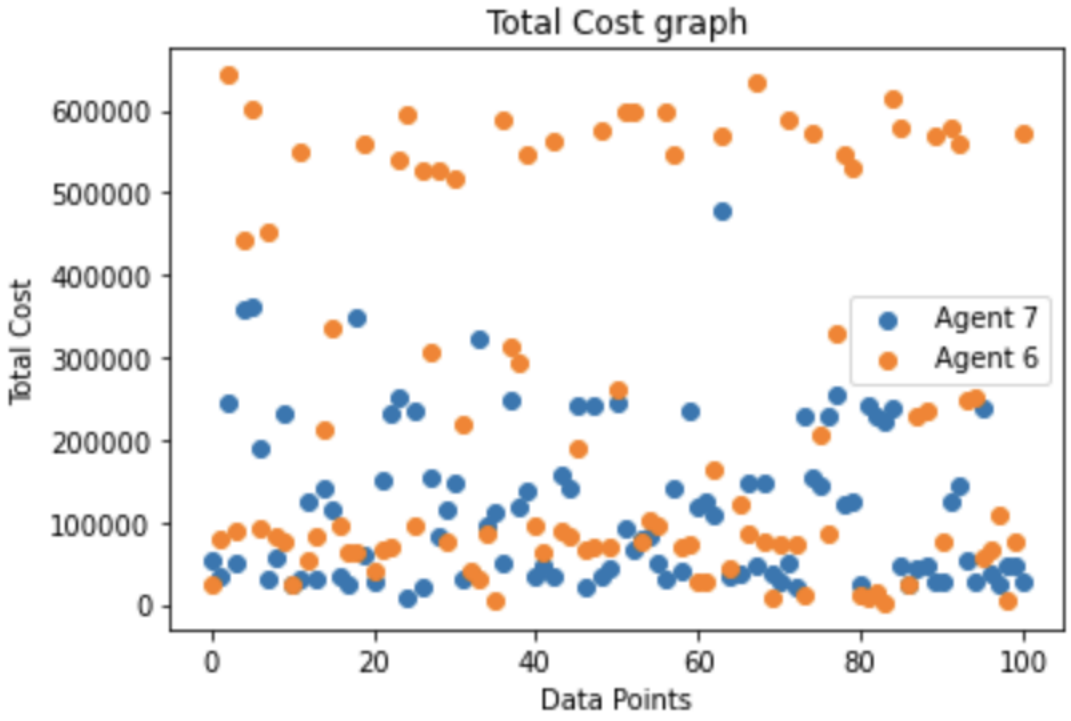

Figure 10: Scatter Plot of Total cost for Agent 6 and 7 comparison

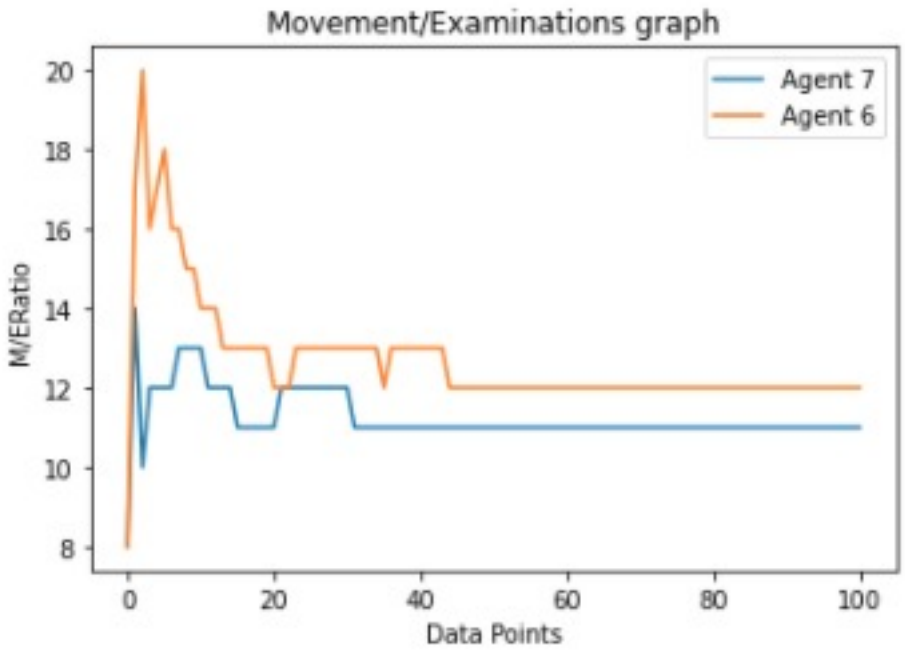

Figure 11: Movements/ Examination plot for Agent 6 and 7 comparison

5 Question 5

5.1 Idea behind the Designing of Agent 8:

We are implementing agent 8 using the probabilities that was used in agent 6 and 7. Here, we include distance factor in the probabilities for the whole grid, due to which the agent 8 decision matrix will change accordingly. We are taking Manhattan distance into account because it is more efficient than Euclidean and Chebyshev as seen in project 1. This will change the max probability of finding the target in the whole grid, which will further change belief state for agent 8 and moreover, decision making process will also change.

5.2 Calculations behind the Designing of Agent 8:

D- Distance between examined cell(x,y) and other unblocked or blocked cells in the whole grid.

$2*n(n =$ dim size of matrix)$-$ Maximum Manhattan distance between for dim*dim matrix (20)

$$Z = \frac{D}{(2 * n)} \tag{21}$$

Belief state of whole grid will change like below:

$$P(celli, j) = \frac{P(celli, j)}{Z} \tag{22}$$

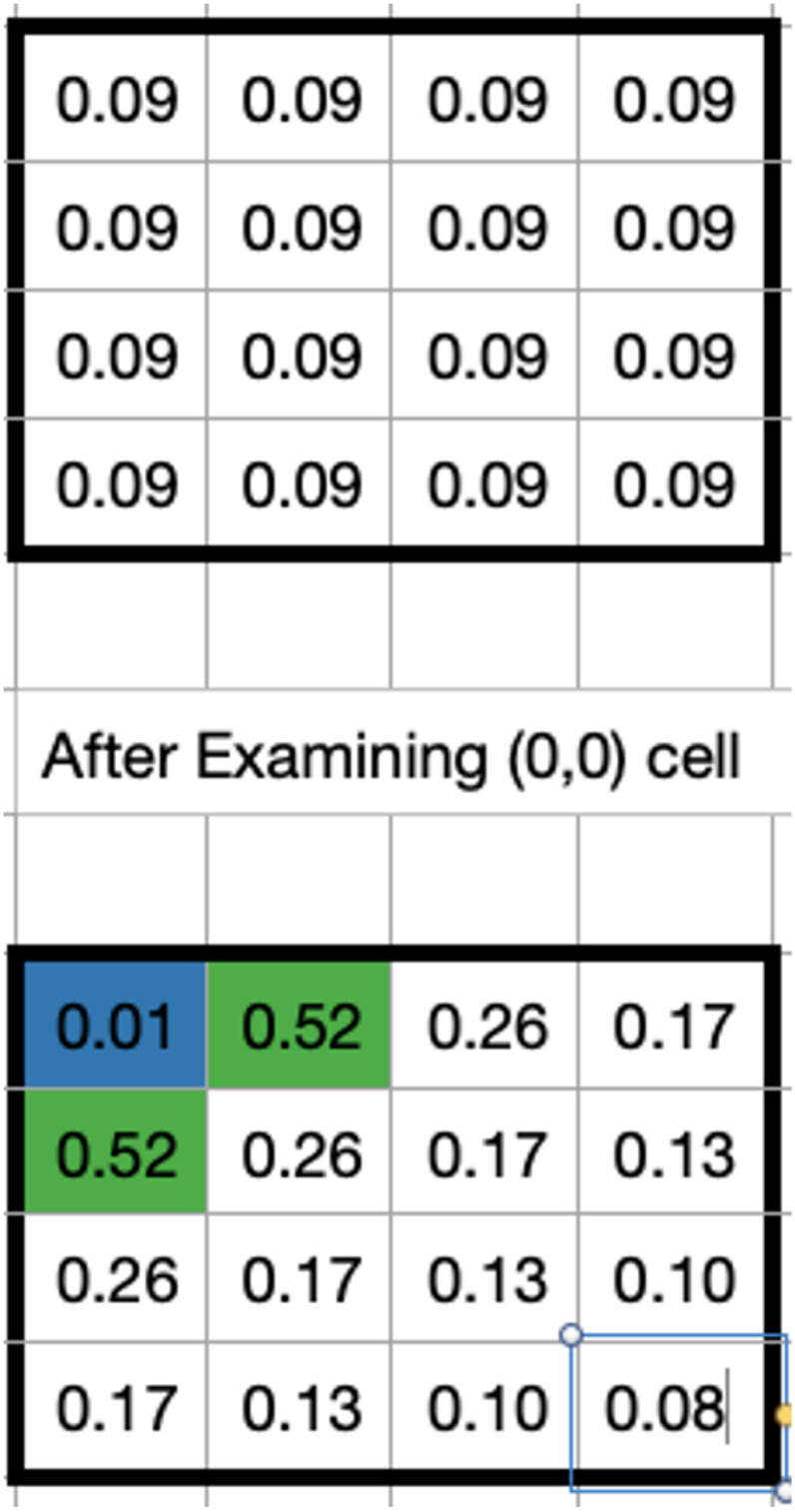

Figure 12: Movements/ Examination plot for Agent 6 and 7 comparison

Here we can see, by taking manhattan distance into the probability matrix, cells which are closer to examined cell will have higher probability as compared to cell which are farther away from examined cell.

5.3 Algorithm Design :

- **Step 1** Initialize the belief state of each cell with probability from formula Question 3 (d). Select the cell with maximum probability and make this cell as the assumed target.
- **Step 2** Here, we start our planning phase in the knowledge gridworld, we will plan our path by calling A-star, taking the cell with max probability as target. After getting the target with max probability, we will normalize whole grid probabilities in terms of manhattan distance. Now the path that we get from A-star will be traversed in the final gridworld inside execution phase.
- **Step 3** Now, we will start our execution phase, agent will start iterating the path that we get from the planning phase. We are updating the terrain in knowledge gridworld as well as the probability of successfully finding the target in that particular cell with formula mentioned in Question 3 (a), (b), (c) while doing the iteration.
- **Step 4** While iterating the path in execution phase if we get a blocked cell then we will update the belief state of whole grid as mentioned in formula Question 2 (a) . Belief state will be updated in terms of blocked cell
- **Step 5** Now we will make another update in terms of Manhattan distance. We will update whole grid probabilities as mentioned in formula

$$P(celli, j) = \frac{P(celli, j)}{Z} \tag{23}$$

  Now, the agent will start replanning, from the cell before the blocked cell and will repeat the process from step 2. During Iteration we will increase the count for movements by 1.

```
#Agent 8
# We are taking account of manhattan distance and updating probability of whole grid accordingly
for i in range(self.n):
        for j in range(self.m):
            if (i,j)!=(x,y):
                manDist=abs(i-x)+abs(j-y)
                prevProb=self.dKprob[i,j][0]
                terrain=self.dKprob[i,j][1]
                Z=manDist/(2*self.n)
                self.dKprob[i,j]=[prevProb/Z,terrain]
```

- **Step 6** During Iteration, we are also checking if we get the max probability of finding the target is different from our assumed target. If that happens, we will do the replanning from the cell in which the agent is there to the cell which has max probability of finding the target and repeat the process from step 2
- **Step 7** When the agent reached the assumed target. It will examine and compare with coordinates of the actual target, If that doesn't match then the belief state of the whole gridworld will change according to the formula given in Question 3 (a), (b), (c). We will increase the examination count by 1 and repeat the whole process of replanning from step 2.
- **Step 8** When the agent reached the assumed target. It will examine and compare with coordinates of the actual target, if it matches, then we will call a random generator, which will generate random value between 0 and 1. If generated value is lesser than the false negative rate of terrain present at the assumed target then examination will result in failure to find the target at that particular cell. Belief state of the whole gridworld will change according to the formula given in Question 2 (c), (d), (e). We will increase examination count by 1 and repeat the whole process of replanning from step 2.
- **Step 9** When the agent reached the assumed target. It will examine and compare with coordinates of the actual target, if it matches, then we will call a random generator, which will generate random value between 0 and 1. If generated value is greater than the false negative rate of terrain present at the assumed target then examination will result in successfully finding the target . We will increase the examination count by 1 and execution phase will end.

6 Question 6

6.1 Agent 8 performance:

Agent 8 performance will improve significantly in terms of total cost as its movements becomes less and examinations will increase because it will examine its neighbouring cells more as compared to other agents

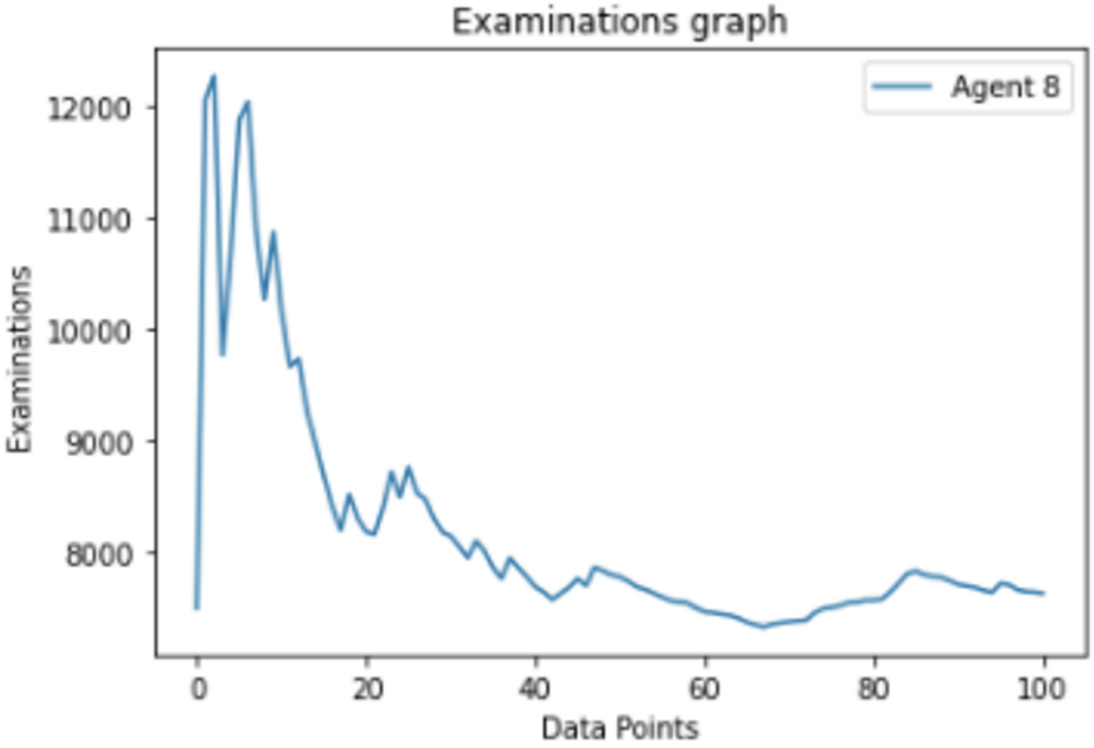

Figure 13: Examinations for Agent 8

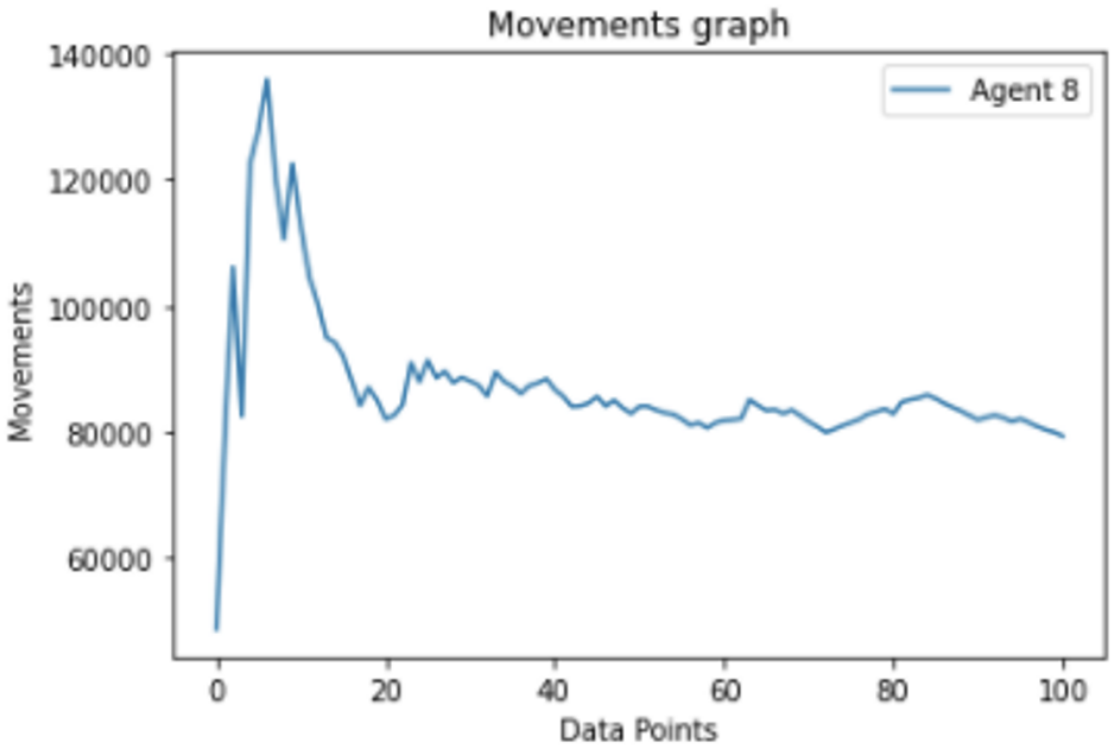

Figure 14: Movements for Agent 8

6.2 Comparison of Agents 6,7 and 8

Agent 8 performance will be going to be better than Agent 7 and 6 because total cost will be lesser in agent 8. But examination will going to increase and movements will decrease significantly.

- **Average Movements Comparison**
  Average movements in cells happened for agent 6: 229400
  Average movements in cells happened for agent 7: 109920
  Average movements in cells happened for agent 8: 78613

- **Average Examinations Comparison:**
  Average examinations of cells happened for agent 6: 14944
  Average examinations of cells happened for agent 7: 6778
  Average examinations of cells happened for agent 8: 7558

- **Average Total Cost Comparison:**
  Average total cost (movements+examinations) of cells happened for agent 6: 244345
  Average total cost (movements+examinations) of cells happened for agent 7: 116698
  Average total cost (movements+examinations) of cells happened for agent 8: 86171

- **Movements per Examination**
  There is another factor through which we can compare agent 8 with agent 7 and 6. From calculating Movements/Examination, we can notice that agent 8 is doing less movements per examination as compared to agent 7 and 6. This means agent 8 will examine closer cells more as compared to other agents. As we are increasing probabilities of closer cells more as compared to cells which are far away.

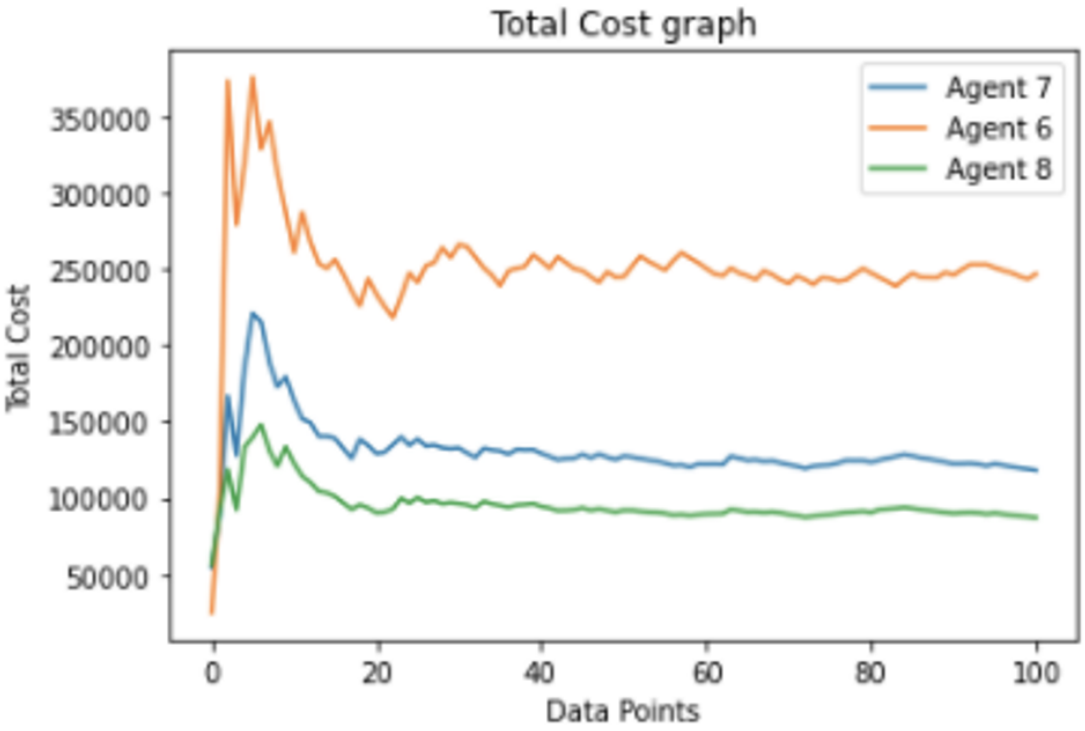

Figure 15: Plot of Total cost for Agent 6, 7, 8 comparison

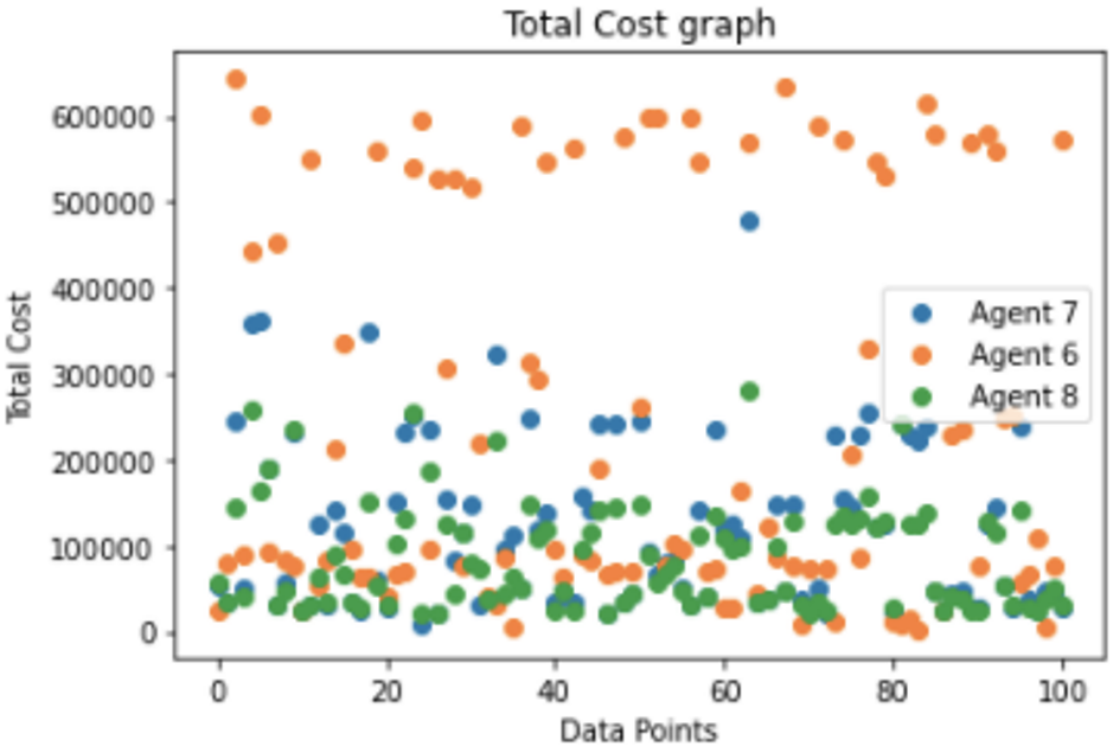

Figure 16: Scatter Plot of Total cost for Agent 6,7,8 comparison

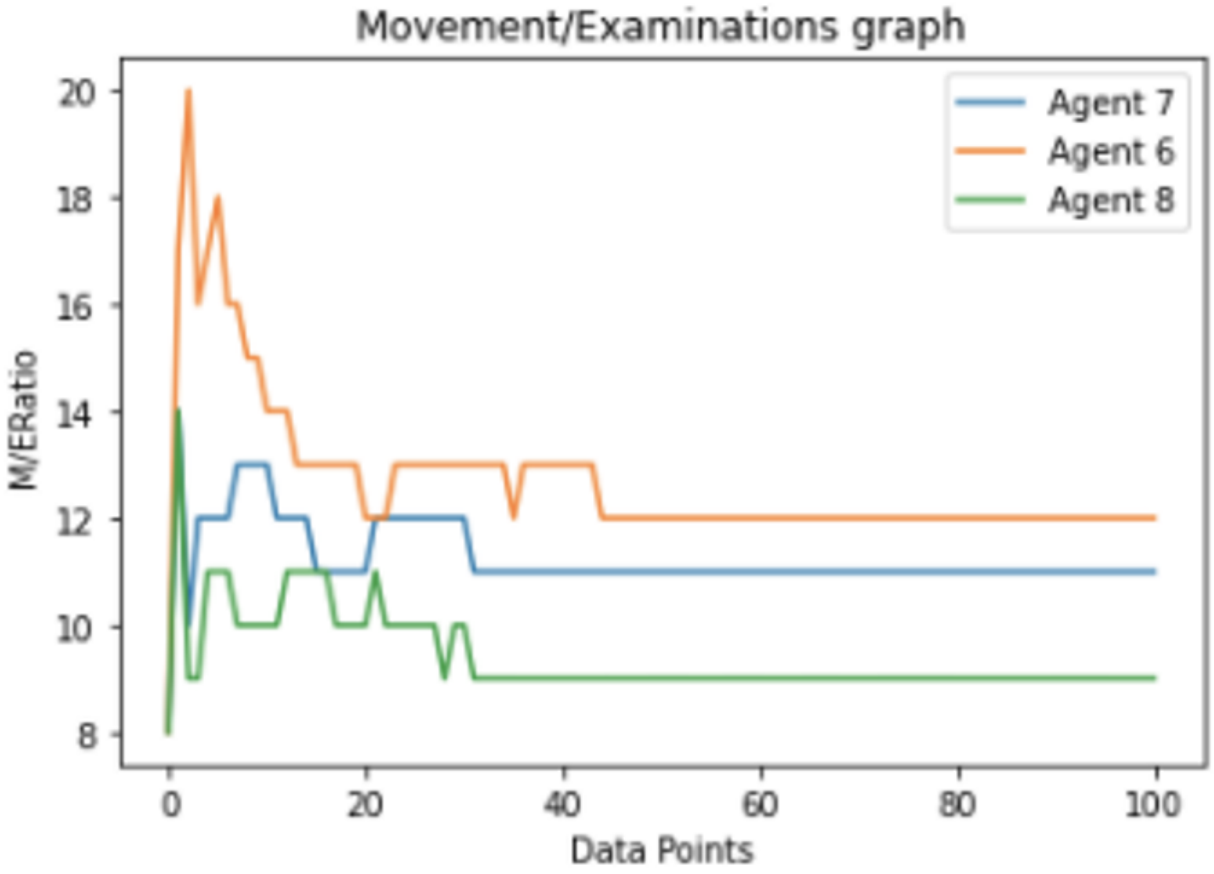

Figure 17: Movements/Examination plot for Agent 6,7,8 comparison

7 Question 7

We can improve agent 8 by introducing clusters in our algorithm. We can use the probabilities in the same manner as we have done in Agent 8. We can include clusters in the decision making, we will plan our path to the cell in the cluster with max probability from the available cluster in a grid.

7.1 Cluster Calculation :

We will add a cluster finding algorithm in the examination step of agent 8 that will improve the performance of agent 8. After we complete the examination step and make an update in probabilities of the whole grid, then we will take the average of all the neighbour's probabilities. And multiply it with cell probability individually. We will complete this process for the whole grid. Then, we will find the cell with max probability which will be present in cluster with maximum averaged probability and set this cell as target node. Furthermore, we will normalize the probabilities of the whole grid and repeat this process of finding clusters again until we find our target.

$$P(i,j) = P(i,j) * P(\text{ average probabilities of all the neighbors of cell}(i,j)) \tag{24}$$

P(average probabilities of all the neighbors of cell(i,j)) - Manhattan distance is already taken into account in calculating probabilities of these cells.

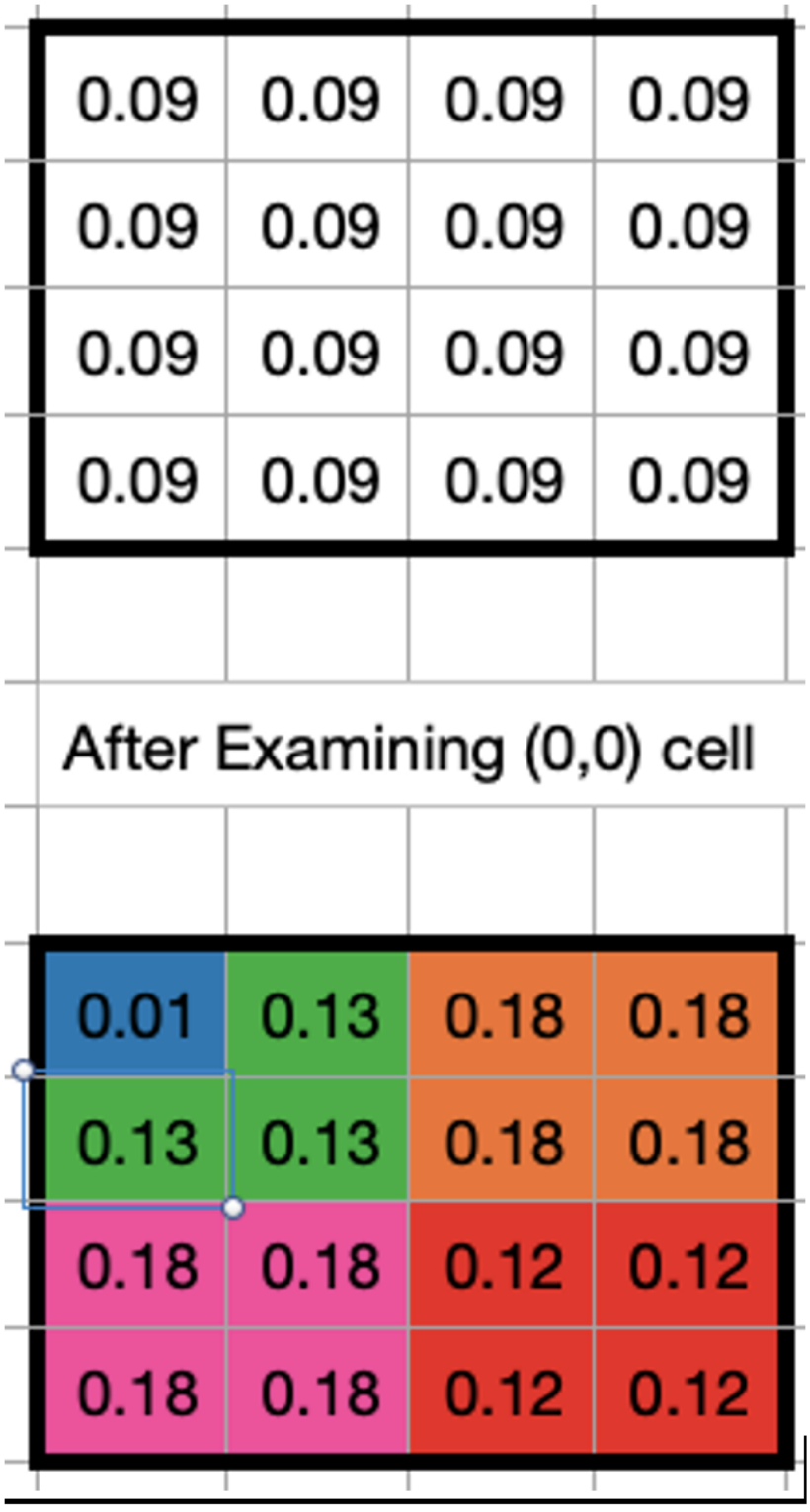

Figure 18: Movements/Examination plot for Agent 6,7,8 comparison

Cells with the same color shows different clusters. All clusters will be made according to manhattan distance and taking average of neighbor's probabilities and multiplying with individual cell.

8 Bonus Question

8.1 Agent 9: Motivation behind designing

Agent 9 can sense all of its 8 neighbors to conclude that target is in one of its neighbors. We will increase the probability of containing the target in all of its 8 neighbors if the target is in one of its neighbors and initiate the rest of the cells with 0 probability. We will make clusters by lowering the probability of cells that have already been sensed. We are book keeping of all the cells that have been sensed and on the basis of that we are searching the whole grid. We will repeat this whole process over and over till we get the target.

```
private boolean senseNeighbors(Node node) {
    for (int i = 0; i < rowD.length; i++) { //agent will check its neighbors
        int row = node.r + rowD[i];
        int col = node.c + colD[i];

        if (isValid(row, col)) {
            if (row == targetR && col == targetC) { //if a blocker found it will update knowledge grid
                return true;
            }
        }
    }
    return false;
}
```

Figure 19: Sensing if target is present in any of the neighbors:

```
private double calculateAverage(Node node) {
    double av = 0;
    for (int i = 0; i < rowD.length; i++) {
        int row = node.r + rowD[i];
        int col = node.c + colD[i];

        if (isValid(row, col)) {
            av += probGrid[row][col];
        }
    }
    return av;
}
```

Figure 20: Calculate Average of the neighbors:

```
private void updateNeighborsProb(Node node, double av) {//increase the prob of neighbors
    for (int i = 0; i < rowD.length; i++) {
        int row = node.r + rowD[i];
        int col = node.c + colD[i];

        if (isValid(row, col)) {
            probGrid[row][col] += av;
        }
    }
}
```

Figure 21: Update the probability of the neighbors:

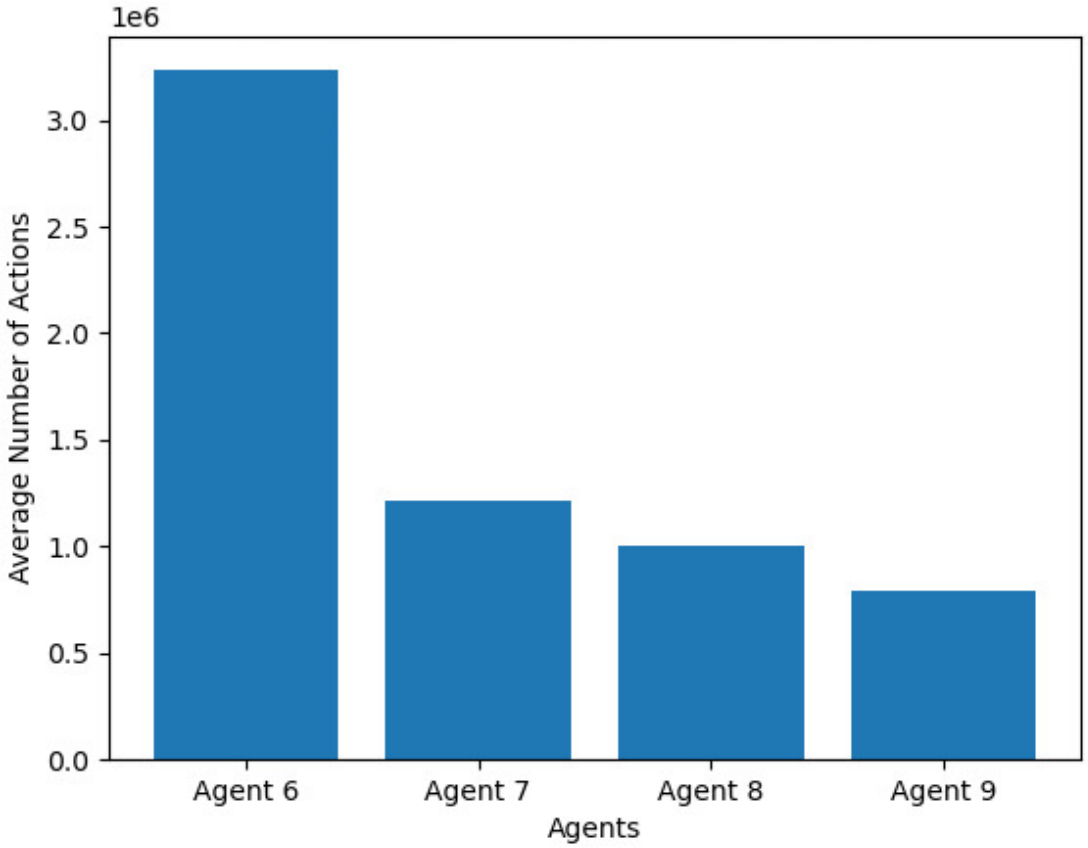

Figure 22: Average no of actions comparison : agent 6,7,8,9