CS 520 : Probabilistic Sensing

Project 3 - Probabilistic Sensing

Karan Ashokkumar Pardasani(kp955)
Naishal Chiragbhai Patel(np781)
Himaniben Hareshkumar Patel(hhp46)

November 16, 2021

[1]

karan.pardasani@rutgers.edu
naishal.patel@rutgers.edu
himani.h.patel@rutgers.edu

Contents

1

Glossary

1. Agent 6: The Blindfolded Agent who is unaware of the position of target and makes probabilistic assumptions for position of the target by considering the probability of a cell containing the target .
2. Agent 7: The Blindfolded Agent, similar to agent 6 who is unaware of the position of target and makes probabilistic assumptions for position of the target by considering the probability of successfully finding the target in a given cell.
3. Agent 8: The Blindfolded Agent who is unaware of the position of target and makes probabilistic assumptions for position of the target by considering the utility function; based on both probability of successfully finding the target in given cell and the distance of cell from the source.
4. Agent 9: The Blindfolded Agent who is unaware of the position of moving target but can sense the presence of target in its 8 neighbourhood cells (N,S,E,W,NW,NE,SW,SE). Also, the agent can detect whether the target is present in the current cell of the agent. The agent will maintain a belief state and predict the next state and make move accordingly.
5. $p\_containing$: the probability of a cell containing the target.
6. $p\_finding$: the probability of successfully finding the target in a given cell.
7. fn: Value of the False negative of the discovered unblocked cell based on the terrain type of the respective cell.
8. Agent Grid : Grid updated by the agent drawing on the knowledge gained using its field of view during traversal of the grid. Initially, based on free space assumption, it is assumed that all cells are unblocked in this grid.
9. Known Grid : Grid containing full knowledge of the environment.
10. Heuristic :
    - g(n) : shortest path length discovered from the *start cell* to *current cell.*

- h(n) : heuristic value which estimates the remaining distance from *current cell* to *goal cell*.
- f(n) : g(n) + h(n), i.e. total path length estimate from *start cell* to *goal cell*

11. Traversed Path : Cells explored by agent while traversing planned path to find the target.
12. Trajectory Length : Length of traversed path(with backtracking).

2

Introduction

2.1 Problem Statement

Here, we design and implement agents that travel a grid of 100 $\times$ 100 matrix where the probability of each cell being blocked is $p = 0.3$. The agent starts at random cell in the grid which can move between unblocked cells in North, South, East, West directions. It can examine the current cell to find the target and the examination can yield false negatives according to the terrain of the unblocked cell. An unblocked cell has an equal likelihood of being each of the following three terrains.

- Flat terrain - False Negative = 0.2
- Hilly terrain - False Negative = 0.5
- Thick forests - False Negative = 0.8

The target cell is also hidden in a random unblocked cell in the grid.
In this report, we have analysed the performance of Agent 6, Agent 7, Agent 8 and Agent 9 in the grid world as follows:

1. Agent 6: A blind folded agent which can identify the cell with the highest probability of **containing** the target using its observations. In case of multiple cells with equal highest probabilities, distance of the cells from current cell is used as a tie breaker. If the distance is also same, then a cell is chosen at random (uniformly) as an assumed target among these cells.

2. Agent 7: The only difference between Agent 6 and Agent 7 is that Agent 7 takes into consideration the highest probability of **successfully finding** the target using its observations.

3. Agent 8: The only difference between Agent 6,7 and Agent 8 is that Agent 8 takes into consideration the utility function of **probability of successfully finding** the target and the **distance** of that cell from current cell using its observations.

4. Agent 9: For this agent, the target moves one cell in each timestep in the above defined gridworld. The agent can sense if the target is present in its 8 immediate neighbours(N, S, E, W, NE, NW, SE, SW), uses its observations and changes the belief according to these observations to find the moving target.

In the first three agents, the inference is done using the knowledge gained after entering a cell. Agents 6, 7 and 8 are blindfolded and they are only able to sense information from the current cell. These agents use this information in different ways to gain more knowledge about the grid. Finally, we evaluate the performance of these 3 agents on different metrics.

2.2 Grid Environment

The grid environment is defined by constraint of knowledge. The agent gains knowledge of the grid as it traverses through the grid via observing terrain type of the cells and by encountering blocks.

2.2.1 Cell States

- Blocked cell : Cells agent can not pass through.
- Unblocked cell : Cells agent can occupy and know the terrain type of as follows.
  - Flat terrain
  - Hilly terrain
  - Thick forests

2.2.2 Parameters

The following parameters regulates the formation of the Grid World.

- Grid Dimension : It determines the size of the Grid World.
- Blocked cell density(p) : Its the probability that a given cell will be blocked. For all the analyses, the value of p is 0.3.
- Terrain type : Unblocked cells are of three terrain types: Flat, Hilly and Forest. Each unblocked cell is equally likely to be of any terrain type.
- Start cell : Initial starting point in the Grid World for the agent. Start cell is chosen at random among the unblocked cells while designing the grid.
- Goal cell : End Target of the agent. Target cell is chosen at random among the unblocked and reachable (from start) cells while designing the grid.
- Constraint: To compare the agents, they are running only on the solvable grids.

2.3 Calculating Probabilities

Some notations required to calculate the probabilities are:

- $P_{(i,j)}(t)$: Probability that cell (i, j) contains the target at time t, given the observations collected till time t.

Question 1: Prior to any interaction with the environment, what is the probability of the target being in a given cell?

Prior to any interaction with the environment, the agent does not know which cells are blocked or unblocked. Also, it does not know the terrain type of each cell. So, initially, there is no reason with agent to believe that one cell has more probability of containing the target than any the other cell. Hence, initially, the probability of a cell containing target will be same for all the cells.

$$P_{t=0}(cell(i,j)\ contains\ the\ target) = P_{(i,j)}(0) = \frac{1}{dim * dim}$$

Question 2: Let $P_{i,j}(t)$ 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 $P_{i,j}(t + 1)$ be for each cell (i, j) under the following circumstances:

2a. At time $t + 1$, what is the probability $P_{(i,j)}(t + 1)$, when the agent attempts to enter the cell (x, y) and the cell is found to be blocked.

Final Answer:

$$\begin{aligned} &\therefore for\ i, j \neq x, y \\ &\quad P_{(i,j)}(t+1) = \frac{P_{(i,j)}(t)}{1 - P_{(x,y)}(t)} \\ &\therefore for\ i, j = x, y \\ &\quad P_{(i,j)}(t+1) = 0 \end{aligned} \tag{2.1}$$

Proof:

$$\begin{aligned} P_{(i,j)}(t+1) &= P(\ cell\ (i,j)\ contains\ Target \mid x, y\ is\ blocked\ ) \\ &\quad (Using\ Bayes\ Theorem) \\ &= \frac{P(\ x, y\ is\ blocked \mid cell(i,j)\ contains\ Target\ ) * P(\ cell(i,j)\ contains\ Target\ )}{P(\ x, y\ is\ blocked\ )} \end{aligned} \tag{2.2}$$

Solving for Denominator using Law of Total Probability and Conditional Factoring:

$$\begin{aligned} P(\,x,y\,is\,blocked\,) &= P(\,x,y\,is\,blocked\,\cap cell\ 1\,contains\,Target\,) \\ &+ P(\,x,y\,is\,blocked\,\cap\ cell\ 2\,contains\,Target\,) \\ &. \qquad (2.3) \\ &. \\ &+ P(\,x,y\,is\,blocked\,\cap\ cell\ dim^2\,contains\,Target\,) \end{aligned} \tag{2.3}$$

(2.3)

$Using\ Bayes\ Theorem$

$$\begin{aligned} \therefore P(\,x,y\,is\,blocked\,) &= P(\,x,y\,is\,blocked\,|\,cell1\,contains\,Target\,) * P(cell\ 1\,contains\,Target\,) \\ &+ P(\,x,y\,is\,blocked\,|\,cell\ 2\,contains\,Target\,) * P(cell\ 2\,contains\,Target\,) \\ &+ P(\,x,y\,is\,blocked\,|\,cell\ 3\,contains\,Target\,) * P(cell\ 3\,contains\,Target\,) \\ &. \\ &. \\ &+ P(\,x,y\,is\,blocked\,|\,(x,y)\,contains\,Target\,) * P(cell\ x,y\,contains\,Target\,) \\ &. \\ &. \\ &+ P(\,x,y\,is\,blocked\,|\,cell\ dim^2\,contains\,Target\,) * P(cell\ dim^2\,contains\,Target\,) \\ \\ \therefore P(\,x,y\,is\,blocked\,) &= P(\,x,y\,is\,blocked\,|\,cell\ 1\,contains\,Target\,) * P(cell\ 1\,contains\,Target\,) \\ &+ P(\,x,y\,is\,blocked\,|\,cell2\,contains\,Target\,) * P(cell\ 2\,contains\,Target\,) \\ &+ P(\,x,y\,is\,blocked\,|\,cell3\,contains\,Target\,) * P(cell\ 3\,contains\,Target\,) \\ &. \\ &. \\ &+ 0 * P(cell\ x,y\,contains\,Target\,) \\ &. \\ &. \\ &+ P(\,x,y\,is\,blocked\,|\,cell\ dim^2\,contains\,Target\,) * P(cell\ dim^2\,contains\,Target\,) \end{aligned} \tag{2.4}$$

We analysed that if a cell ***a*** contains target then probability of cell ***b*** being blocked will increase. This is because since we want to find the probability that ***a*** contains target, we will first need to account that ***a*** is unblocked. When ***a*** is assumed to be unblocked, then probability of undiscovered cells being blocked should increase (since before (x, y) was discovered, there was some probability of the (x, y) being blocked, but then it became 0. So, that probability will redistribute to other cells). Let's denote this probability by α :

$$P(x,y\,is\,blocked) = \frac{(total\ undiscovered\,blocks)}{(total\ undiscovered\,cells)} = \alpha \tag{2.5}$$

$$P(\, x,y\, is\, blocked\, |\, cell\ (i,j)\ contains\, Target\,) = \alpha\ (for\ (i,j) \neq (x,y)\ and\ (i,j)\ is\, not\, blocked) \tag{2.6}$$

This probability α will be same for all cases. This is because, let's consider the two probabilities:

1. Probability of x, y being blocked given that the target is in (i, j)
2. Probability of x, y being blocked given that the target is in (p, q),.

Now, if the cells (i,j) and (p,q) are both discovered and unblocked, then the two probabilities should be same since, the target being in two different cell doesn't change anything for the other cell to become blocked. So, the probabilities should be same. **So, α is same for all cells.**

Also, if the cell (i, j) and (p, q) are undiscovered, then by free space assumption we assume them to be unblocked and hence they also will have same probability α as in discovered unblocked cell.

$$\begin{aligned} P(\, x,y\, is\, blocked\,) &= \alpha * P(cell\ 1\ contains\, Target\,) \\ &+ \alpha * P(cell\ 2\ contains\, Target\,) \\ &. \\ &. \\ &+ 0 * P(cell\ (x,y)\ contains\, Target\,) \\ &. \\ &. \\ &+ \alpha * P(cell\ dim^2\ contains\, Target\,) \end{aligned} \tag{2.7}$$

$\because$ At time t:

$$P(cell\ 1\ contains\, Target\,) + ... + P(cell\ (x,y)\ contains\, Target\,) + ... + P(cell\ dim^2\ contains\, Target\,)\ =\ 1 \tag{2.8}$$

$$P(cell\ 1\ contains\, Target\,) + ... + P(cell\ dim^2\ contains\, Target\,)\ =\ 1 - P(cell\ (x,y)\ contains\, Target \tag{2.9}$$

$$\begin{aligned} \therefore P(\, x,y\, is\, blocked\,) &= \alpha\left(P(cell\ 1\ contains\, Target\,) + ... + P(cell\ dim^2\ contains\, Target\,)\right) \\ &= \alpha(1 - P(cell\ (x,y)\ contains\, Target\,)) \end{aligned} \tag{2.10}$$

Simplifying *equation* 2.1 using *equation* 2.6 and *equation* 2.9

$$\begin{aligned}
P_{(i,j)}(t+1) &= \frac{P(\,x,y\,is\,blocked \mid cell\ i,j\ contains\ Target\,) * P(cell\ i,j\ contains\ Target\,)}{\alpha(1 - P(cell\ x,y\ contains\ Target\,))} \\
&= \frac{\alpha * P(cell\ i,j\ contains\ Target\,)}{\alpha * (1 - P(cell\ x,y\ contains\ Target\,))} \\
\therefore for\ i,j &\neq x,y \\
P_{(i,j)}(t+1) &= \frac{P_{(i,j)}(t)}{1 - P_{(x,y)}(t)} \\
\therefore for\ i,j &= x,y \\
P_{(i,j)}(t+1) &= 0
\end{aligned} \tag{2.11}$$

This is basically weighted distribution of $P_{(x,y)}(t)(\because P_{(x,y)}(t+1) = 0)$ to all other cells such that $\sum P_{(i,j)}(t+1) = 1$

2b. At time t + 1 you attempt to enter (x, y), find it unblocked, and also learn its terrain type?

Final Answer

$$P_{(i,j)}(t+1) = P_{(i,j)}(t)\ for\ all\ (i,j)\ in\ grid \tag{2.12}$$

Proof

It is reasonable to assume from free space assumption that the probability of containing the target when cell is undisovered is same as the probability of containing target when cell is unblocked. It will not change when an unblocked cell is discovered. Also, as discussed by Prof. Cowan, if we do not assume the free space assumption, then probability calculations becomes complex.
If the agent attempts to enter (x,y) and finds it to be unblocked, then the value of probability will not change.

2c. At time t + 1 you examine cell (x, y) of terrain type flat, and fail to find the target?

Final Answer

$$\begin{aligned}
\therefore for\ i,j &\neq x,y \\
P_{(i,j)}(t+1) &= \frac{P_{(i,j)}(t) * 1}{P_{(0,0)}(t) + ... + FN * P_{(x,y)}(t) + ... + P_{(dim,dim)}(t)} \\
\therefore for\ i,j &= x,y \\
P_{(i,j)}(t+1) &= \frac{P_{(i,j)}(t) * FN}{P_{(0,0)}(t) + ... + FN * P_{(x,y)}(t) + ... + P_{(dim,dim)}(t)}
\end{aligned} \tag{2.13}$$

For Plain Terrain Type:

$$\begin{aligned}
&\therefore for\ i,j \neq x,y\\
&P_{(i,j)}(t+1) = \frac{P_{(i,j)}(t) * 1}{P_{(0,0)}(t) + ... + 0.2 * P_{(x,y)}(t) + ... + P_{(dim,dim)}(t)}\\
&\therefore for\ i,j = x,y\\
&P_{(i,j)}(t+1) = \frac{P_{(i,j)}(t) * 0.2}{P_{(0,0)}(t) + ... + 0.2 * P_{(x,y)}(t) + ... + P_{(dim,dim)}(t)}
\end{aligned} \tag{2.14}$$

Proof

$$\begin{aligned}
&P(\ cell\ i,j\ contains\ Target\ |Target\ was\ not\ found\ in\ x,y)\\
&\qquad = \frac{P(Target\ was\ not\ found\ in\ x,y\ |\ cell\ i,j\ contains\ Target\ ) * P(cell\ i,j\ contains\ Target\ )}{P(Target\ was\ not\ found\ in\ x,y)}\\
&for\ i,j \neq x,y\\
&\qquad = \frac{1 * P(cell\ i,j\ contains\ Target\ )}{P(Target\ was\ not\ found\ in\ x,y)}\\
&for\ i,j = x,y\\
&\qquad = \frac{FN * P(cell\ i,j\ contains\ Target\ )}{P(Target\ was\ not\ found\ in\ x,y)}
\end{aligned}$$

(2.15)

Solving for Denominator using Law of Total Probability and Conditional Factoring:

$$\begin{aligned}
P(Target\ was\ not\ found\ in\ x,y) = {} & P(Target\ was\ not\ found\ in\ x,y \cap Target\ in\ (0,0))\\
& .\\
& .\\
& + P(Target\ was\ not\ found\ in\ x,y \cap Target\ in\ (x,y)\\
& .\\
& .\\
& + P(Target\ was\ not\ found\ in\ x,y \cap Target\ in\ (dim,dim)
\end{aligned} \tag{2.16}$$

$Using\ Bayes\ Theorem$

$\therefore P(Target\ was\ not\ found\ in\ x,y)$

$$= P(Target\, was\, not\, found\, in\, x,y \mid Target\, in\, (0,0)) * P(Target\, in\, (0,0))$$
$$.$$
$$.$$
$$+ P(Target\, was\, not\, found\, in\, x,y \mid Target\, in\, (x,y) * P(Target\, in(x,y))$$
$$.$$
$$.$$
$$+ P(Target\, was\, not\, found\, in\, x,y \mid Target\, in\, (dim,dim) * P(Target\, in\, (dim,dim))$$

For the above equation, we know that P(Target was not found in x,y | Target in (x,y)) will be FN (false negative for the respective cell) and for other cells $(i,j \neq x,y)$ P(Target was not found in i,j | Target in (x,y)) will be 1.

$$\therefore \mathrm{P}(Target\, was\, not\, found\, in\, x,y)$$

$$= 1 * P(Target\, in\, (0,0)) + ... + FN * P(Target\, in\, (x,y)) + ... + 1 * P(Target\, in\, (dim,dim))$$

$$\therefore P(Target\, was\, not\, found\, in\, x,y) = P_{(0,0)}(t) + ... + FN * P_{(x,y)}(t) + ... + P_{(dim,dim)}(t) \tag{2.17}$$

Now, simplifying *equation* 2.11 using *equation* 2.13

$$\begin{aligned} &\therefore for\ i,j \neq x,y \\ &P_{(i,j)}(t+1) = \frac{P_{(i,j)}(t) * 1}{P_{(0,0)}(t) + ... + FN * P_{(x,y)}(t) + ... + P_{(dim,dim)}(t)} \\ &\therefore for\ i,j = x,y \\ &P_{(i,j)}(t+1) = \frac{P_{(i,j)}(t) * FN}{P_{(0,0)}(t) + ... + FN * P_{(x,y)}(t) + ... + P_{(dim,dim)}(t)} \end{aligned} \tag{2.18}$$

For Plain Terrain Type:

$$\begin{aligned} &\therefore for\ i,j \neq x,y \\ &P_{(i,j)}(t+1) = \frac{P_{(i,j)}(t) * 1}{P_{(0,0)}(t) + ... + 0.2 * P_{(x,y)}(t) + ... + P_{(dim,dim)}(t)} \\ &\therefore for\ i,j = x,y \\ &P_{(i,j)}(t+1) = \frac{P_{(i,j)}(t) * 0.2}{P_{(0,0)}(t) + ... + 0.2 * P_{(x,y)}(t) + ... + P_{(dim,dim)}(t)} \end{aligned} \tag{2.19}$$

2d. At time t + 1 you examine cell (x, y) of terrain type hilly, and fail to find the target?

For Hilly Terrain Type:

$$\begin{aligned}
&\therefore for\; i,j \neq x,y \\
&P_{(i,j)}(t+1) = \frac{P_{(i,j)}(t) * 1}{P_{(0,0)}(t) + ... + 0.5 * P_{(x,y)}(t) + ... + P_{(dim,dim)}(t)} \\
&\therefore for\; i,j = x,y \\
&P_{(i,j)}(t+1) = \frac{P_{(i,j)}(t) * 0.5}{P_{(0,0)}(t) + ... + 0.5 * P_{(x,y)}(t) + ... + P_{(dim,dim)}(t)}
\end{aligned} \tag{2.20}$$

2e. At time t + 1 you examine cell (x, y) of terrain type forest, and fail to find the target?

For Thick Forest Terrain Type:

$$\begin{aligned}
&\therefore for\; i,j \neq x,y \\
&P_{(i,j)}(t+1) = \frac{P_{(i,j)}(t) * 1}{P_{(0,0)}(t) + ... + 0.8 * P_{(x,y)}(t) + ... + P_{(dim,dim)}(t)} \\
&\therefore for\; i,j = x,y \\
&P_{(i,j)}(t+1) = \frac{P_{(i,j)}(t) * 0.8}{P_{(0,0)}(t) + ... + 0.8 * P_{(x,y)}(t) + ... + P_{(dim,dim)}(t)}
\end{aligned} \tag{2.21}$$

2f. At time t + 1 you examine cell (x, y) and find the target?
If at $t+1$, we find Target at x, y then $P_{(i,j)}(t+1) = 0\; for\; (i,j) \neq (x,y)$
$P_{(i,j)}(t+1) = 1\; for\; (i,j) = (x,y)$

Question 3: (8 points) At time t, with probability $P_{(i,j)}(t)$ of cell (i, j) containing the target, what is the probability of finding the target in cell (x, y):

Final Answer:

General Equation (for discovered unblocked cell):

$$P(\,finding\,Target\,in\,i,j) = (1 - FN) * P(i,j\; contains\; Target) \tag{2.22}$$

Proof:
Solving using Law of Total Probability and Conditional Factoring:

$$\begin{aligned} P(\,finding\,Target\,in\,i,j) = {} & P(\,finding\,Target\,in\,i,j \cap Target\,in\,(0,0)) \\ & . \\ & . \\ & + P(\,finding\,Target\,in\,i,j \cap Target\,in\,(i,j) \\ & . \\ & . \\ & + P(\,finding\,Target\,in\,i,j \cap Target\,in\,(dim,dim) \end{aligned} \tag{2.23}$$

$Using\,Bayes\,Theorem$
$\therefore P(\,finding\,Target\,in\,i,j)$

$$\begin{aligned} & = P(\,finding\,Target\,in\,i,j \mid Target\,in\,(0,0)) * P(Target\,in\,(0,0)) \\ & . \\ & . \\ & + P(\,finding\,Target\,in\,i,j \mid Target\,in\,(i,j) * P(Target\,in(i,j)) \\ & . \\ & . \\ & + P(\,finding\,Target\,in\,i,j \mid Target\,in\,(dim,dim) * P(Target\,in\,(dim,dim)) \end{aligned}$$

For the above equation, we know that P(Finding target in i,j | Target in (x,y)) where $x, y \neq i, j$ will be 0 and for $i, j = x, y$ P(Finding target in i,j | Target in (i,j)) will be (1-FN) (if target is in a cell, the probability of agent successfully finding it will be 1 - FalseNegative based on the terrain type). For discovered block cells Probability of finding the target will be 0 since their probability of containing the target is 0.

General Equation (for discovered unblocked cell):

$$P(\,finding\,Target\,in\,i,j) = (1 - FN) * P(i,j\,contains\,Target) \tag{2.24}$$

If (x, y) is hilly?

$$P(\,finding\,Target\,in\,i,j) = (1 - 0.5) * P(i,j\,contains\,Target) \tag{2.25}$$

If (x, y) is flat?

$$P(\,finding\,Target\,in\,i,j) = (1 - 0.2) * P(i,j\,contains\,Target) \tag{2.26}$$

If (x, y) is forest?

$$P(\,finding\,Target\,in\,i,j) = (1 - 0.8) * P(i,j\,contains\,Target) \tag{2.27}$$

If (i,j) is never visited

If cell (i,j) is never visited, then we consider the expected value of False Negative for calculating the probability of finding the target. Hence, Expectation = $\frac{1}{3}0.2 + \frac{1}{3}0.5 + \frac{1}{3} * 0.8 = 0.5$ the False Negative

$$P(\,finding\,Target\,in\,i,j) = (1 - 0.5) * P(i,j\,contains\,Target) \tag{2.28}$$

3

Agent 6

Agent 6 is a blind folded agent which can only know if a cell is blocked by attempting to move to a cell and failing. If the agent is able to successfully enter a cell (unblocked type); then it is able to sense terrain type of that cell. The agent will maintain a belief state which keeps track of the probabilities of a cell containing the target. This knowledge is updated when agent makes more observations about environment. For traversing across the grid; since agent is unaware of the location of target, it will assume the location of target based on the probability of a cell containing target. The preference in assuming target is considered in following order:

1. Choose cell with maximum probability of containing the target.
2. If there are multiple cells with equal probability (maximum) then check for cells with lowest distance from the current source using DFS.
3. If again there are multiple cells which have maximum probability and are equidistant; then any one cell is chosen uniformly at random.

3.1 Knowledge Representation

1. **class Agent6**:
    - **Agent Grid**: 2-D array that represents current knowledge of the agent about the grid.
    - **Grid**: 2-D array that represents full knowledge of the grid. This grid is used when the agent wants to know whether the cell it enters is blocked or not. Also, this grid is used to know the terrain type of the unblocked cell.

    The blocked cells are represented as **X** and the unblocked cells are represented with their respective terrain type: **P** for terrain type 'Flat', **H** for terrain type 'Hilly' and **F** for terrain type 'Forest'.

2. **class GridWorld**:

- **P__containing**: Dictionary that stores probability of a cell containing the target. The key is the tuple representing the cell and the value is the corresponding probability of containing the target.
- **fn**: Dictionary that stores the value of false negative of the discovered unblocked cell based on the respective terrain type; as the agent traverses along the planned path. The key is the tuple representing the cell and the value is:
  - 0.2 if terrain type is Flat (P)
  - 0.5 if terrain type is Flat (H)
  - 0.8 if terrain type is Flat (F)

3. **class MyPriorityQueue**
   - **current__heuristic**: Variable used to specify the heuristic that should be used to calculate priority. **m** denotes Manhattan distance, **p** represents Manhattan distance plus Probability of a cell being blocked(will use this for Agent 5)
   - **__data**: Object of class Sorted Set which is used to store cells of the priority queue.
   - **g**: Dictionary used to store the g(x) for each cell x. It represents the distance of the cell x from source.
   - **h**: Dictionary used to store the h(x) for each cell x. It represents the estimation of the distance of the cell x from target.

3.2 Implementation

Intuitive steps for Agent 6 implementation:

1. The agent starts from source cell provided. According to the free space assumption, initially, all the cells of the *agent__grid* are unblocked.
2. Also, in the beginning, all the cells will have equal probability of containing the target: 1/dim*dim.
3. The agent plans a path using the A-star algorithm from it's current position to the assumed target.
4. The agent traverses the cells along it's planned path. There are two cases when the agent encounters an undiscovered cell:
   (a) If the cell is unblocked, agent will know its false negative based on terrain type and this knowledge will be updated in *fn* dictionary.
   (b) If a blocked cell is encountered then the probabilities of a cell containing target will be updated and agent will choose a new assumed target based on the new belief state. The agent will again plan a new path(step 3).
5. While traversing the path, if the agent reaches the assumed target and fails to find the target; then the probabilities of a cell containing the target will be updated and agent will again choose a new assumed target and plan a new path(step 3).

3.2.1 Pseudocode

TRAVERSE_PLANNED_PATH(planned_path):

1. Traverse through each cell of the planned path. For each cell do:
    - (a) cell ← planned_path[i], currx ← row_index , curry ← col_index
    - (b) If the cell is not visited by the agent do:
        - i. Mark the cell as block or update its Terrain type (unblocked) in the agent grid based on the grid containing full knowledge.
        - ii. If an unblocked cell is discovered, the fn dictionary is updated with cell index as key and the respective value of false negative based on the terrain type.
    - (c) If the current cell is blocked, then set the **restart_cell** to the previous cell of the planned path. **restart_cell** is the cell from which the Repeated A-star will start replanning. Break from the loop that traverses from the planned path. **block_cell** is set to the current cell and **block_encounter** flag is set to true which will be used to call the function **propogate_probability_foundblock(block_cell)** where **block_cell** is passed as parameter.
    - (d) If the current cell is unblocked, then set the **restart_cell** to the current cell of the planned path.
2. Once the agent traverses the full planned_path, return the traversed_path, restart_cell, block_encounter and block_cell.

COMPUTE_PATH():

1. First calls **initialize_probability()**. Probability of all the cells containing the target will be initialized as $\frac{1}{dim^2}$.
2. Calls **new_target(current_source)**. This function returns assumed target based on current belief state.
3. Get planned path from current source to the assumed target by calling **a_star(current_cell)**.
    - (a) If empty path is returned from A* call; then it means that our assumed target is not reachable. This cell is *as good as block cell*, so we propagate probability by calling **propogate_probability_foundblock(block_cell)** where the index of the unreachable cell is passed as parameter. After this new target is assumed by calling **new_target(current_source)** (continue from step 3).
    - (b) If path returned by A* is not empty, then **traverse_planned_path(planned_path)** is called; which returns traversed_path, restart_cell, block_encounter and block_cell.
        - i. If *block_encounter* flag returned is true; that means a block cell was encountered while traversing the planned path and we need to propagate the probability of containing the target by calling **propogate_probability_foundblock(block_cell)**.

ii. If the last cell in the list of traversed_path is equal to our assumed target; means that our agent has reached the assumed target and now *examination* will take place. The agent will examine the current cell; if agent fails to find the target (either target was actually not present in the assumed target or target was missed due to false negative of the terrain) the probabilities of containing the target are updated by calling **propogate_probability_notfoundt()**. If the agent successfully finds the target then the program exits and list of total path traversed is returned.

iii. In either of the above cases where a block cell was encountered in the planned path or the agent failed to find the target during examination; we have to repeat the process by setting current cell as *restart_cell* (returned by traverse_planned_path function) and the new target is assumed by calling **new_target(current_cell)** (continue from step 3).

INITIALIZE_PROBABILITY():

1. Probability of cells containing the target for every cell will be initialized as $\frac{1}{dim^2}$ at time t=0 due to free space assumption.

PROPOGATE_PROBABILITY_FOUNDBLOCK(BLOCK_CELL):

1. This function updates probability as shown in equation 2.10 (Question 2a).
2. First we calculate *denominator* $1 - P_{(x,y)}(t)$.
3. For *numerator* of the *block_cell* it will be 0; since probability of containing the target in a blocked cell is 0. For cells (i,j) other than the block_cell the *numerator* will be $P_{(i,j)}(t)$ (Probability of cell i,j containing the target at time t).
4. The probability of cell i,j at time t+1 will be $\frac{numerator}{denominator}$ (as calculated above).

PROPOGATE_PROBABILITY_NOTFOUNDT():

1. This function updates probability as shown in equation 2.15, 2.16 and 2.17 (Question 2c, 2d and 2e).
2. First we calculate *denominator* $P_{(0,0)}(t) + ... + FN * P_{(x,y)}(t) + ... + P_{(dim,dim)}$.
3. For *numerator* of the *cell* where agent failed to find the target will be $P_{(i,j)}(t) * FN$ and for cells i,j other than the cell where agent failed to find the target will be $P_{(i,j)}(t)$ (where $P_{(i,j)}(t)$ is the probability of cell i,j containing the target at time t).
4. The probability of cell i,j at time t+1 will be $\frac{numerator}{denominator}$ (as calculated above).

NEW_TARGET(CURRENT_CELL):

1. The new target is assumed on the basis of probabilistic knowledge gathered till that time.
2. New target is chosen by giving following preference:
    (a) Cells with maximum **probability of containing the target** ($P_{(i,j)}(t)$): A list *tmp* containing all such cells is made.
    (b) If there are more than one cell in *tmp* list; then choose cells with lowest distance from current source (using DFS) and add them to a different list *tmp*2.
    (c) If there are more than one cell in *tmp*2 list; then choose any cell at random.
    (d) This cell will be **new assumed target**.

4

Agent 7

Agent 7 is also a blind folded agent which has the same characteristics as agent 6. Except the probability of successfully finding the target in a cell will be taken into account when assuming the new target rather than the probability of containing the target in a cell as done by agent 6.
For agent 7, we will store and update probability of containing the target as done earlier in agent 6. Just at the time of assuming new target we compute probability of successfully finding the target from the probability of cell containing the target.
As discussed in Section 2.3,

If cell (i, j) is discovered and unblocked

$$P(successfully\,finding\,the\,target\,at\,cell\,(i,j)) = (1 - Fn) * P(target\,in\,cell\,(i,j)) \tag{4.1}$$

If cell (i, j) is discovered and blocked,

$$P(successfully\,finding\,the\,target\,at\,cell\,(i,j)) = 0 \tag{4.2}$$

If cell (i, j) is undiscovered

$$P(successfully\,finding\,the\,target\,at\,cell\,(i,j)) = 0.5 \cdot P(containing\,the\,target\,at\,cell\,(i,j)) \tag{4.3}$$

Agent 7 considers the probability of finding the target in the cell for estimating new target location. This probability connects the terrain type of discovered unblocked cells with the probability of a cell containing the target.
If the cell is of terrain type forest; its probability of finding the target will be penalised more. Since, the forest cell will not tell us the correct answer with higher probability, if target is present in the cell.
Also, the consequence of considering this probability is that the if the probability of containing the target is same for two cells of different terrain, then the agent will given preference of going to the cell with lower false negative rate. This makes sense in a way since it should be more beneficial to travel to the cell with lower false negative value.

Since we consider probability of finding the target of a given cell for making assumptions about new target; it will be (1 - FN) * (Probability of a cell containing the target). So the factor (1 - FN) will be less for higher FN values. Hence, discovered cells with terrain type forest will be penalised more with respect to cells with terrain type hilly and flat. So Agent 7 has better probabilistic assumptions for the next position of target as compared to Agent 6.

4.1 Implementation

Intuitive steps for Agent 7 implementation:

1. The agent starts from source cell provided. According to the free space assumption, all the cells of the *agent_grid* are unblocked initially.
2. Initially, all the cells will have equal probability of containing the target: 1/dim*dim.
3. A path is planned using the A-star algorithm from current source to the assumed target.
4. The agent traverses the cells along it's planned path. If the cell is unblocked, agent will know its false negative based on terrain type and this knowledge will be updated in *fn* dictionary. If a blocked cell is encountered then the probabilities of a cell containing target will be updated and agent will choose a new assumed target based on the new belief state and probability of finding the target in a cell (eqn 2.15). In case of block cell is encountered there will be no change in the probability of cell finding the target; it will be same as the probability of cell containing the target.
5. If the agent reaches the assumed target and fails to find the target; then the probabilities of a cell containing the target will be updated and agent will again choose a new assumed target based on the probability of finding the target in a cell (eqn 2.15).

4.1.1 Pseudocode

NEW_TARGET(CURRENT_CELL):

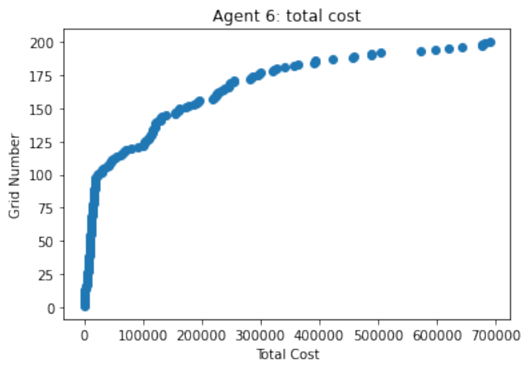
1. The new target is assumed on the basis of probabilistic knowledge gathered till that time.
2. New target is chosen by giving following preference:
   (a) Cells with maximum **probability of finding the target**; for discovered unblocked cells $((1-FN)*P_{(i,j)}(t))$ and for discovered block or undiscovered cells $(P_{(i,j)}(t))$ (Equations 2.18 - 2.22 Question 3): A list *tmp* containing all such cells is made.
   (b) If there are more than one cell in *tmp* list; then choose cells with lowest distance from current source (using DFS) and add them to a different list *tmp*2.

(c) If there are more than one cell in $tmp2$ list; then choose any cell at random.

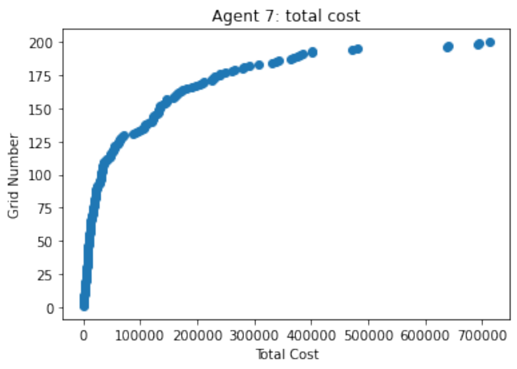
(d) This cell will be **new assumed target**.

	Agent 6	Agent 7
Average Examine Cost	13364.65	12594.77
Average Movement Cost	106276.22	85982.92
Average Total Cost	119640.87	98577.69

Table 4.1: Cost: Agent 6 vs Agent 7



(a)



(b)

Figure 4.1: (a) Agent 6: total cost (b) Agent 7: total cost

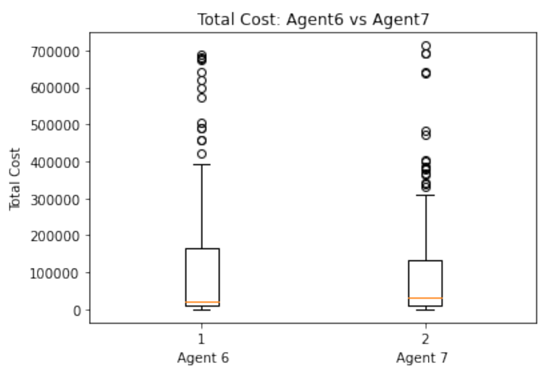
4.2 Analysis

Question 4: Implement Agent 6 and 7. For both agents, repeatedly run each agent on a variety of randomly generated boards (at constant dimension) to estimate the number of actions (movement + examinations) each agent needs on average to find the target. You will need to collect enough data to determine which of these agents is superior. Do you notice anything about the movement/examinations distribution for each agent? Note, boards where the target is unreachable from the initial agent position should be discarded.

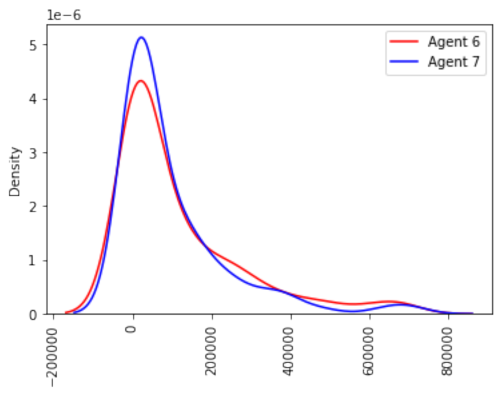
Distribution of Total Cost

Agent 7 outperforms Agent 6 in terms of number of actions as seen in the Table 4.1. From the data shown in figure 4.1 and 4.2 and comparing average values as shown in table 4.1: we can see that there is small decrease in average examine cost from agent 6 to agent 7. But there is major decrease in average movement cost between agent 6 and agent 7. We can conclude that the probabilities with which estimations of target location are made by agent 7 are more accurate than that by agent 6. On average, Agent 7 performs better than Agent 6.

From Figure 4.2(a), we can see the comparison of boxplots between Agent 6 and Agent 7, we can see that the 100-th percentile of total cost are all lower in Agent 7 than Agent 6. So, after removing the possible outliers in the boxplot, we can see from boxplot that Agent 7 outperforms than Agent 6.



(a)



(b)

Figure 4.2: Agent 6 vs Agent 7: Total Cost (a) Box Plot (b) Density Plot

Also, we can see that the median values (in box plot) for Agent 6 is higher than that in Agent 7. But average is much less in Agent 7 than in Agent 6. This huge increase in values of outliers is responsible for huge average in Agent 6. So, considering only the outliers, Agent 7 performs better than Agent6
If we see the 100-th percentile, then also Agent 7 performs better than Agent 6.

So, from the observation from the plots, we can interpret that Agent 7 performs better than Agent 6. This is because Agent 7 considers the cost of successfully finding the target. So, when we are finding a new target location, for cells having same probability of containing target, we tend to prioritise the assumed target to a cell with lower False Negative Value. This is correct because there are higher chances of successfully finding the target in plain cell than in forest cells.
Figure 4.2 (b) denotes the KDE plot of total cost of the agent. From the plot we can see that the density of grids near the peak of the KDE plot is greater than in agent 7 than in agent 6. Hence, we can conclude that the most of the grids are solved in less movements in Agent 7 than in Agent 6. Also, as movement increases, the density at higher movement of Agent 6 gets higher as compared to Agent 7 which means that more number of grids are solved using high movement cost in Agent than Agent .

Box Plot of Ratio between Movement Cost and Examination Cost

From Figure 4.3, we can see the box plot of the ratio between movement cost and examination cost. From this metric, we can interprete after how many movement does the agent examines a cell. In Figure 4.3, we can see that 75th and 100th percentile is lower in Agent 7 than in Agent 6. This means that it takes less movement to travel to the next assumed target. So, the agent 7 spends less time in movement before doing an examination than Agent 6.

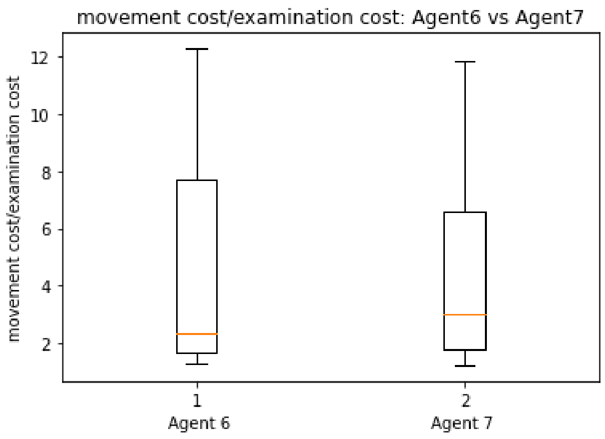


Figure 4.3: Box Plot: Agent 6 vs Agent 7: movement cost/examination cost

5

Agent 8

5.1 Design

Question 5: Describe your algorithm, be explicit as to what decisions it is making, how, and why. How does the belief state $P_{(i,j)}(t)$ enter into the decision making? Do you need to calculate anything new that you didn't already have available?

Agent 8 is also a blindfolded agent which moves in the exact same way as agent 6 and agent 7. For making assumptions about the position of the next assumed target, we will not only take probability into account like in agent 6 and 7 but also take into consideration the distance of the assumed target from current cell. This is helpful since, if a cell with probability 0.4 at distance 10 and another cell with probability 0.42 is at distance 20, then we can check the cell at distance 10 since the difference in probabilities is negligible as compared to the difference in distance.

Here, we formulate a utility function on the basis of which our agent will estimate the next location of the target. In agent 6 and 7 probability was the only factor considered for the next location of the target. Now, lets assume a case where a cell with probability of finding the target is 0.2 and is at far distance from the source; we have couple of cells along the path with probability of finding the target as 1.9 which are at much smaller distance. All those cells will not be examined by agent 6 or 7. So here, we will also take distance into account for making our estimation of next location of the target. Since we want to consider cells with more probability and lesser distance from source we designed our utility function as: Probability of finding the target at a given cell / Distance of cell from the source i.e.

$$Utility\ Function = \\ P(finding\ target\ in\ cell\ (x,y))/Distance\ of\ cell\ from\ the\ source$$

Since probability values range from 0 to 1; we will **normalize** our **distance** in the same range. For that we will divide our distance by the maximum possible value of Manhattan distance which is (dim + dim). This is how the agent will calculate the utility function, which calculates about the location of the target.

This takes into account both probability of finding the target at a given cell and distance of cell from the source. Since agent 8 takes into account multiple factors as compared to agent 6 and 7; it will have much better estimate for target locations (lesser cost factor).

Do you need to calculate anything more that you did not have to already calculate?

For agent 8, we have to calculate the utility value for each cell based on the probability of successfully finding the target in the cell and distance of the agent from the cell. In this agent, we have introduced a new concept for making decision as to which cell to examine next. Hence, to make a more informed decision we have to calculate utility from probabilities and distances that we already have in Agent 6 and Agent 7.

5.2 Implementation

Intuitive steps for Agent 8 implementation:

1. The agent starts from source cell provided. According to the free space assumption, all the cells of the *agent_grid* are unblocked initially.
2. Initially, all the cells will have equal probability of containing the target: 1/dim*dim.
3. A path is planned using the A-star algorithm from current source to the assumed target.
4. The agent traverses the cells along it's planned path. If the cell is unblocked, agent will know its false negative based on terrain type and this knowledge will be updated in *fn* dictionary. If a blocked cell is encountered then the probabilities of a cell containing target will be updated and agent will choose a new assumed target based on the new belief state. The new target will be chosen based on the utility function; which considers both probability of finding a target in the given cell and the distance (found by DFS) from cell to the source.
5. The utility function implemented by us is (Probability of finding the target in a given cell / Distance of cell to the source). The new target is assumed according to the maximum utility; in case of multiple cells with utility score, the preference is given to cell with least distance and if still multiple such cells are found then one is chosen at random as new assumed target.
6. If the agent reaches the assumed target and fails to find the target; then the probabilities of a cell containing the target will be updated and agent will again choose a new assumed target based on the utility score mentioned above.

	Agent 6	Agent 7	Agent 8
Average Examine Cost	13364.65	12594.77	11129.65
Average Movement Cost	106276.22	85982.92	19683.58
Average Total Cost	119640.87	98577.69	30813.23

Table 5.1: Cost: Agent 6 vs Agent 7 vs Agent 8

5.2.1 Pseudocode

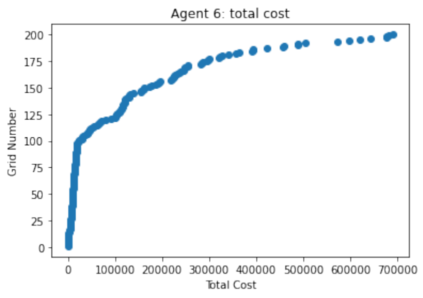
NEW_TARGET(CURRENT_CELL):

1. The new target is assumed on the basis of probabilistic knowledge gathered till that time.
2. New target is chosen by giving following preference:
   (a) Utility score is defined by considering both *probability of finding the target in a cell* and *distance of that cell from current source.* Utility function used in our algorithm is $\frac{probability}{distance}$. Probability of finding the target in a cell is same as calculated in Agent 7 (Equations 2.18 - 2.22 Question 3) and distance of a cell from the current source is calculated using DFS.
   (b) Cells with maximum **utility score**: A list *tmp* containing all such cells is made.
   (c) If there are more than one cell in *tmp* list; then choose cells with lowest distance from current source (using DFS) and add them to a different list *tmp*2.
   (d) If there are more than one cell in *tmp*2 list; then choose any cell at random.
   (e) This cell will be **new assumed target**.

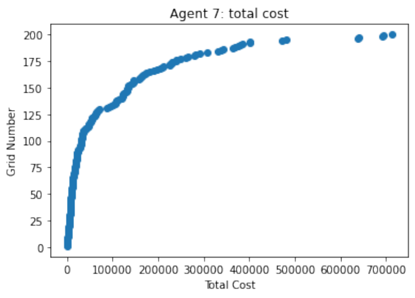
5.3 Analysis

Question 6: Implement Agent 8, run it sufficiently many times to give a valid comparison to Agents 6 and 7, and verify that Agent 8 is superior.
From the table 5.1, it is clear that the average of the total number of actions are far lesser for Agent 8 as compare to Agent 6 and Agent 7. Also, the overall average cost of examination and movement is also less in Agent 8 than in Agent 6, 7.

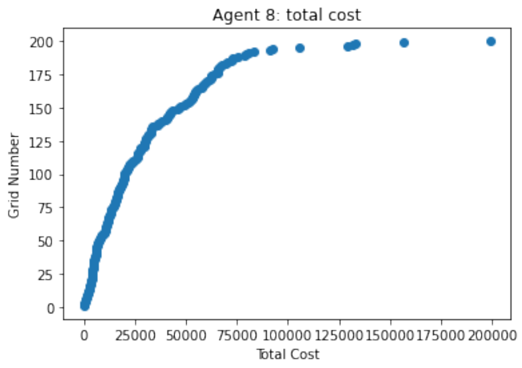
Also, the decrease in examination cost is not much as compared to agent 6 ad agent 7 but the decrease in movement cost is significant as compared to agent 6 and agent 7. Thus we can conclude that the performance by Agent 8 is better than performance compared to agent 6 and agent 7.



(a)



(b)



(c)

Figure 5.1: Result Plots: (a) Agent 6: total cost (b) Agent 7: total cost (c) Agent 8: total cost

Comparing Scatter Plots of Agent 6 vs Agent 7 vs Agent 8

From Figure 5.1 (a), (b), (c), we can see that for Agent 6 and Agent 7, the maximum cost in grid is almost same, but the density of the grids at higher at big values of movement in Agent 6 than in Agent 7. So, Agent 6 is better in terms of total cost than Agent 7.

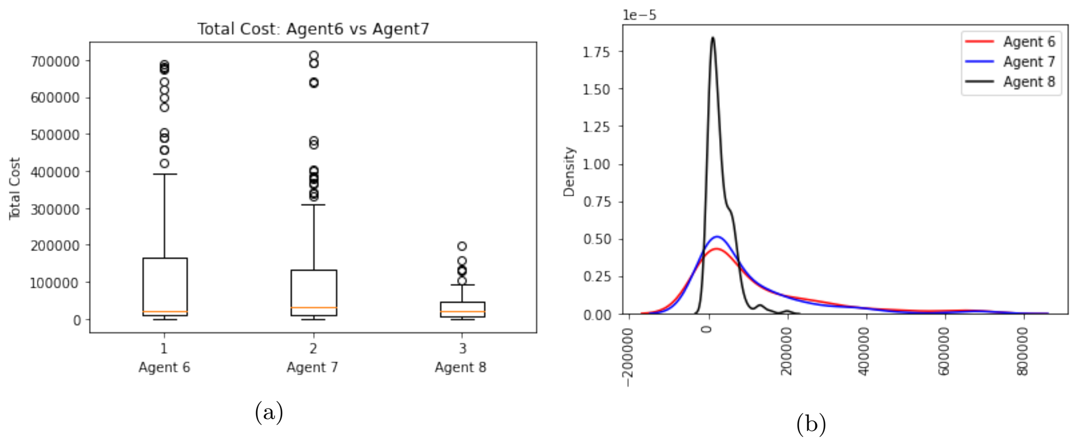
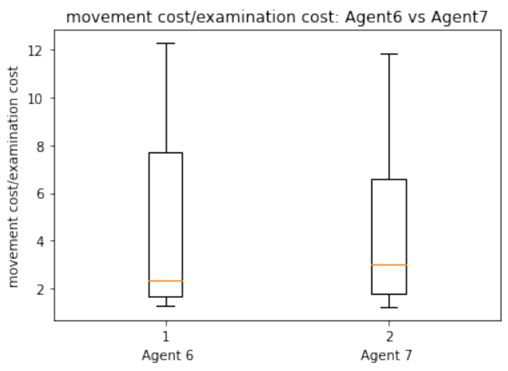


Figure 5.2: Agent 6 vs Agent 7 vs Agent 8: Total Cost (a) Box Plot (b) Density Plot: Density vs Total Cost

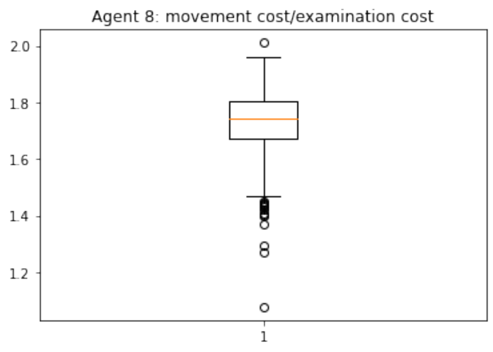
Comparing Boxplots and KDE of Agent 6 vs Agent 7 vs Agent 8 for Total Cost

From Figure 5.2(a), from the box plot, we can see that Agent 8 greatly outperforms Agent 6 and Agent 7. Along with the lower 25th, 50th and 75th percentiles, we can also see that the value of movement in outliers is also much lower than that of Agent 6 and Agent 7. Comparison of box plots of Agent 6 and Agent 7 is done in Question 4.

From the KDE plots of Agent 6, 7, 8 in Figure 5.2(b), we can see that significantly more grids are solved at lower cost in Agent 8 as compared to agent 6 and 7. Also, the maximum movements that are required by Agent 8 is much lower than the previous two agents. This drastically reduces the total cost for agent 8 with respect to Agent 6 and 7.



(a)



(b)

Figure 5.3: (a) Box Plot: Agent 6 vs Agent 7: movement cost/examination cost (b) Box Plot: Agent 8: movement cost/examination cost

Comparison between movement/examination of Agent 6, 7, 8

Fig 5.3 (a), (b), shows the boxplot of movement cost / examination cost for agent 6, 7, 8. From the graph, we can see that the ratio is minimum for agent 8. It is almost around 6 times to that of Agent 6 or Agent 7. This is because in agent 8, we are having the utility as probability/distance. So, consider cells at distance 1 and distance 2, then the probabilities at distance 2 has to be greater than the double of cell at distance 1 to be chosen as assumed target. So, it will be very difficult to have the cell with maximum utility at distance greater than 2.

Question 7: How could you improve Agent 8 even further? Be explicit as to what you could do, how, and what you would need.

In Agent 8, we are currently only estimating the next location of the target, but further improvement can be done as follows:

1. We can estimate the next location of the target as we are doing in Agent 8. And then, we can again find the next location of the target given that the target is not found in previous location. Then we have two assumed target location - One, where we are currently going to and the other where we will go next, given that the target is not found where we are currently going. Hence, we can take the addition of these two utilities instead of considering only the next target location.

2. We can find a cluster of cells in the grid that have combined good probability of successfully finding the target and thus the agent can reduce its movement cost even further.

3. Also, at the same place, the agent can avoid a high probability cell in the middle of a wasteland(i.e. cluster of cells having very low finding probability). Therefore, the agent does not have to travel to and from less useful cells. Thus reducing the movement cost.

6

Agent 9

Bonus: For Agent 9, we let the target move each time step, but only to one of its immediate neighbors (uniformly at random among unblocked neighbors). The Agent can additionally sense in each cell whether the target is in one of its 8 immediate neighbors, but not which one. Build an Agent that adapts to this moving target and extra partial sensing. How does it decide where to go next, and what to do? How is the belief state updated? Implement, and generate enough data for a good comparison.

6.1 Problem Statement

The following points represent the characteristics of the Agent 9:

1. The agent starts from a random unblocked cell.
2. The target is initially present on a random unblocked cell.
3. The objective is to find the target.
4. The target randomly moves to one of it's neighbouring unblocked cell.
5. The agent does two observations:
   (a) It can examine in it's current cell and check whether the target is present in the current cell
   (b) It senses whether the target is present in one of the 8 neighbouring cell in the direction: (N, S, W, E, NW, NE, SW, SE).
   It is important to note that the target can only know whether the target is in vicinity but not in which cell the target is present.

6.2 Knowledge Representation

We have defined the following classes and data structures to implement Agent-9:

1. **class GridWorld**:

    (a) m, n: Stores the number of rows and columns respectively

    (b) start: Represents initial position of the agent

    (c) target: Represents the current position of the target

    (d) grid: 2-d array representing the grid. Each cell can be one of the following types:
    - **X** : Cell is blocked
    - · : Cell is undiscovered
    - **F** : Cell is present in Forest area
    - **P** : Cell is present in flat/plain surface
    - **H** : Cell is present in Hilly area

2. **class Agent9**:

    (a) **n, m** : Represents the dimension of the grid

    (b) **agent_grid**: Represents the current knowledge of the agent

    (c) **grid**: Represents the complete knowledge of the grid

    (d) **target**: Represents the current location of the target

    (e) **start**: Starting position of the agent

    (f) **assumed_target**: The location to which the agent is planning path. For Agent 9, it will be the target with the highest value of utility.

    (g) **fn**: Dictionary representing False Negative Values for each terrain type. So, fn = 'P':0.2,'H':0.5,'F':0.8, '.': 0.5

    (h) **p_containing**: 2-d array that represents the probability of target in a cell.

    (i) **prediction**: 2-d array that represents the prediction of the location of the target in the next time step

    (j) **pre_vicinity**: Boolean variable that denotes whether target was present in the vicinity in the previous timestamp.

    (k) **dist**: 2-d array where dist[i][j] represents the distance of shortest path from i-th node to j-th node.

    (l) **examination_cost**: Number of times the agent examines the current cell to check whether target is present in the cell or not.

    (m) **movement_cost**: Number of times the agent move until it successfully finds the target.

6.3 Design

6.3.1 Initial Phase

Initially, there is no information about any cell, so the target can be present in any cell. Hence, the probability of every cell containing target is equal. Hence, in the initial belief at time 0, the probability of each cell containing target is $\frac{1}{dim^2}$.

6.3.2 Sensing Phase

Initially, the agent is present in an unblocked cell. The agent will sense whether the target is present in the surrounding cells or not.
If the target is present in the vicinity then, we will update the cells in the following way:

$$\begin{aligned} P_t((i,j)\,|target\,is\,in\,vicinity\,of\,cell\,(x,y)) &= 0\;(i,j)\;\notin\;neighbourhood((i,j)) \\ &= \frac{P_{t-1}((i,j))}{1-\sum_{(i,j)\,\notin\,neighbours((x,y))}P_{t-1}(i,j)}\;(i,j)\;\in\;neighbourhood((i,j)) \end{aligned} \tag{6.1}$$

Also, when the target is in the vicinity, the **pre_vicinity** variable, which denotes whether the target was present in the vicinity in the previous time step, is changed to **True**. After updating the belief at time t and updating the **pre_vicinity** variable, the agent goes to pre_planning phase
If the target is not present in the vicinity then, first **pre_vicinity** variable is set to False.

Then there can be a case where the value of **pre_vicinity** is True. In this case, the agent will check whether the target is present in the current cell or not. If the target is present, then the algorithm exits. But if the target is not present the belief of the cells are changed in this way:

$$\begin{aligned} P_t((i,j)\,|\,target\,is\,not\,found\,in\,cell\,(x,y)) &= \frac{P_{(i,j)}(t)*1}{P_{(0,0)}(t)+...+FN*P_{(x,y)}(t)+...+P_{(n,n)}(t)},for\;i,j\neq x,y \\ &\ \frac{P_{(i,j)}(t)*FN}{P_{(0,0)}(t)+...+FN*P_{(x,y)}(t)+...+P_{(n,n)}(t)},for\;i,j=x,y \end{aligned} \tag{6.2}$$

After updating the belief for all cells, we have already sensed that the target is not present in the vicinity, so we will update the cells in the following way:

$$\begin{aligned} P_t((i,j)\,|target\,is\,not\,in\,vicinity\,of\,cell\,(x,y)) &= 0\;,for\;(i,j)\;\in\;neighbourhood((i,j)) \\ &= \frac{P_{t-1}((i,j))}{1-\sum_{(i,j)\,\in\,neighbours((x,y))}P_{t-1}(i,j)}\;,for\;(i,j)\;\notin\;neighbourhood((i,j)) \end{aligned} \tag{6.3}$$

After the algorithm updates the belief based on the sensing information, the agent will go into the pre-planning phase, where it will find the assumed target in the grid.

6.3.3 Pre-Planning Phase

Case 1: The target is in the vicinity of the agent

If the target is in the vicinity, then the agent does not move from it's current position. Only the target will move from it's current position. This is a reasonable decision since if the target is to the N, S, E, W direction of the agent, and the agent moves when the target is in vicinity, then they will never meet. But if the agent does not move, then there is a chance that the target from the neighbouring cell comes to the same cell as the agent. So, the agent just waits for the target to come to the agent's cell.

So, if the target is in the vicinity of the agent, then the target moves to the neighbouring cells. Since, the target moves to the neighbouring cell, our prediction for the next time step will become the belief after the target moves.

Why does the prediction for the next time step becomes the belief after the target moves?

In this probabilistic model, the agent can make following observation at each cell:

1. Whether the target is present in vicinity.
2. Whether the target is present in current cell
3. The terrain of the current cell
4. The target moves uniformly in the neighbouring unblocked cell

The above claim is derived from the agent's last observation. Let's say we have the belief at time t and we want to find the probability of the target being in a particular cell in the next time step. Formally, we want the prediction at time $t+1$, given we have the observations till time t. To compute them, we used the following formula:

$$Pred_{t+1}((i,j) \mid observations\ until\ time\ t) = \sum_{(i',j')} P_t(target\ is\ at\ (i',j')) \cdot Prob(X_{t+1} = (i,j) | X_t = (i',j')) \tag{6.4}$$

Now, we can compute the prediction of the target at cell (i', j') at time t. Then, after the agent and target moves, we don't know where the target is but we d know that wherever the target is, it has moved uniformly to it's neighbouring unblocked cell.

Now, when the target moves to the neighbouring cells, we need to update probability that considers the fact that the target has moved to it's neighbouring cell. So, to consider this observation into probability, we can use the same probability update equation (6.4). According to the equation, we can interprete it as, the probability of the target being in the cell at time $t+1$, is the sum of probability of the target being in the neighbouring unblocked cell times the probability of going to that cell from neighbouring cell.

Hence, to compute the prediction of location of target at time $t+1$ we use the formula 4.6, where belief at time t is the P_t. Also, to compute the belief at time $t+1$, after the target moves to the neighbouring cell is also, computed using equation 6.4. Hence,

$$Pred_{t+1} = Belief_{t+1}((i,j) \mid target\ has\ moved\ randomly\ to\ the\ neighbouring\ unblocked\ cell) \tag{6.5}$$

Case 2: If the target is not in the vicinity of the agent
Now, if the target is not in the vicinity of the agent, then after updating probabilities in the sensing phase, it will predict the next location of the target using equation 6.5. After predicting the next location of target, the agent moves into the planning phase.

Algorithm to find the new position of target based on prediction:

- Calculate prediction using equation 6.5
- Calculate the actual distance of all the cells from the agent using Depth First Search.
- After getting the prediction for the location of the target in the next time step, we find the location with the maximum utility (prediction at time t+1 / distance of cell from the agent).
- The new location of the assumed target will be the cell with the maximum utility.

6.3.4 Planning Phase

During the planning phase of the algorithm, the agent tries to find the path from the current position to the target. There can be two cases:

1. There is no path in the agent's knowledge from current position to the assumed target. In this case, the assumed target is as good as a blocked cell, so he algorithm marks the cell as blocked and updates the belief to the equation 6.1. Then the agent will again find the new target using he algorithm explained in the previous section. It will continue finding new target until planned path is not found.
2. If there is a planned path in the agent's knowledge from the current position to the assumed target, then the agent will then try to move onto the next cell.

6.3.5 Execution Phase

After getting a planned path from the current location of agent, it tries to go to the next cell. There can be two cases:

1. **The next cell of the planned path is blocked:** In this case, the agent will attempt to move in the next cell, but since it is blocked, the belief of the agent will update according to equation 6.1.

   Also, the target will move since the agent tried to move into the next cell. The algorithm will update the belief to be the prediction of the previous time stamp. After getting the belief of target at the next timestamp, the agent will again predict the next location of the target. Then, it will find a new target with maximum utility using the prediction matrix. After finding the new target, the agent will again go to the planning phase to get the new planned path.

2. **The next cell of the planned path is unblocked:** In this case, the agent will move on to the next cell. The target will randomly move to the next neighbouring unblocked cell. Also, the prediction of the previous time step will become the belief of the current time step. Again the agent will go to the sensing phase, to check whether the target is in the vicinity of the agent.

6.4 Implementation

6.4.1 Pseudocode

```
SENSE(curr_pos)):
Inputs: curr_pos - current position of the agent.

1. Initially, the function initialises variables
    (a) vicinity = False, denotes whether the target is in the vicinity
    (b) sum_prob = 0, denotes the sum of the belief in the neighbourhood of the cells.
2. Traverse through the neighbours of the current position. For each neighbour cell $(i, j)$ do:
    (a) If the neighbouring cell $(i, j)$ is not blocked in the agent's knowledge of the grid, do the following:
        i. Add the $(i, j)$'s probability to the sum_prob.
        ii. If $(i, j) == target.x, target.y$, then set vicinity = True
3. Set total_prob = p_containing.sum(), denotes the total_probability of the belief matrix. This is always close to 1, but due to python limitations someimes it has an error of $10^{-7}$ i.e. the sum ranges from $1 - 10^{-7}$ to $1 + 10^{-7}$
4. If vicinity == False:
    (a) Traverse through each cell $x$ in the grid and do the following:
        i. If the cell $x$ is present in the neighbourhood of the current position, then set p_conaining[$x$] = 0
        ii. If the cell $x$ is not present in the neighbourhood of the current position, then set $p\_containing[x] = \frac{p\_containing[x]}{total\_prob - sum\_prob}$
5. If vicinity == True:
    (a) Traverse through each cell $x$ in the grid and do the following:
        i. If the cell is present in the neighbourhood of the current position, then set $p\_containing[x] = \frac{p\_containing[x]}{total\_prob - sum\_prob}$
        ii. If the cell is not in the neighbourhood of the cell, then set $p\_containing[x] = 0$
6. Return **vicinity**
```

UPDATE_PROB_NOT_CONTAINING_TARGET(curr_pos))

```
1. Obtain the current terrain of the cell. terrain ← agent_grid.grid[curr_pos]
2. $fn \leftarrow fn[terrain]$
3. $prob \leftarrow p\_containing[curr\_pos]$
4. total_prob = p_containing.sum()
5. We need to update the following:
     • $denominator = total\_prob + (FN - 1) * prob$
     • If cell $(i, j) == curr\_pos$, then $p\_containing[cell] = \frac{p\_containing[cell]}{denominator}$
     • If cell $\neq curr\_pos$, then $p\_containing[cell] = \frac{p\_containing[cell]}{denominator}$
   So, the algorithm does the following:
     (a) $denominator = total\_prob + (FN - 1) * prob$
     (b) $self.p\_containing = self.p\_containing/denominator$
     (c) $self.p\_containing[cell] = FN * prob/denominator$
```

EXAMINATION_CELL(curr_pos))

```
1. examination_cost += 1
2. If target is not at current_position:
     • UPDATE_PROB_NOT_CONTAINING_TARGET(curr_pos)
     • return FALSE
3. If target is at current position:
     (a) Generate random number between 0 and 1,
     (b) If the random number is $\leq FN$, call UPDATE_PROB_NOT_CONTAINING_TARGET (curr_pos) and return $False$
     (c) If random number $> FN$ return $True$
```

GENERATE_PREDICTION_MATRIX()

```
1. prediction = np.zeros((m,n)) - Set prediction matrix to 0.
2. Traverse each cell in the grid and do the following:
     (a) If the cell in the grid is blocked, continue
     (b) For each cell traverse it's unblocked neighbours in the four directions(directions from which target can arrive in the current cell of the loop) and **count** will store the number of unblocked neighbours of the grid.
```

(c) Again for each cell traverse it's unblocked neighbours in the four directions(directions from which target can arrive in the current cell of the loop) and update the following:

$$prediction[neighbour] = prediction[neighbour] + p\_containing[current]/count \tag{6.6}$$

3. Return the **prediction matrix**.

CALCULATE_DISTANCE(curr_pos)
This function calculates the shortest distance from the current position of the agent to all the other cells in the grid using Breadth First Search.

1. Initialisation -
   (a) for each cell, $dist[cell] \leftarrow \infty$
   (b) for each cell, $visited[cell] \leftarrow 0$
   (c) $dist[curr\_pos] \leftarrow 0$
   (d) $visited[curr\_pos] \leftarrow 1$
   (e) Queue q.append(curr_pos)
2. while Queue gets empty, repeat the following:
   (a) Pop a cell from the queue.
   (b) Iterate through the four neighbouring cells (N, S, E, W) and do the following:
      i. If the $neighbouring\_cell$ is not blocked and $visited[neighbouring\_cell] = True$, then
      mark the neighbouring cell visited,
      update the dist[neighbouring cell] = dist[curr] + 1,
      append the neighouring cell to the queue.
3. After the queue gets empty, we will get the minimum distance of all the cells from the location of the agent.

UPDATE_PROBABILITY_CONTAINING_BLOCK(block_pos) We will update the following if cell (x, y) is blocked:

$$\begin{aligned} P_t(i,j) &= \frac{P_t(i,j)}{Total_Prob - P_t(x,y)} \; when \; (i,j) \neq (x,y) \\ &= 0 \; when \; (i,j) = (x,y) \end{aligned} \tag{6.7}$$

1. $total\_prob \leftarrow p\_containing.sum()$
2. $prob \leftarrow p\_containing[block\_pos]$
3. p_containing = p_containing/(total_prob - prob)
4. $p\_containing[block\_pos] = 0$

MOVE_TARGET()

1. Initialisation - neighbours = []
2. From the current_location of the target, traverse each neighbour of the cell.
   - Append the neighbouring cell only if it is unblocked.
3. Select the random cell from the list that you have generated.

PRE-PLANNING
This function is used to calculate the cell with the maximum utility. The utility is the ratio of **prediction of successfully finding the target in the cell in the next time step** and **the distance of the source from the target**.

1. CALCULATE_DISTANCE(curr_pos)
2. GENERATE_PREDICTION_MATRIX()
3. Initialisation:
   (a) **assumed_target** = None
   (b) **max_utility_cells** = []
4. Traverse through all the cells in the grid and do the following for each cell:
   (a) If the cell is unblocked and the distance from the agent > 0, then do the following:
      i. Get the terrain and False negative value of the cell
      ii. Calculate the current_utility of the cell. Use the formula

$$curr\_utility = \frac{(1 - FN) * prediction[curr\_cell]}{dist[curr\_cell]} \tag{6.8}$$

      iii. If the curr_utility == max_utility, then append the cell to the max_utility_cell
      iv. If the curr_utility > max_utility, then create a new max_utility_cell list:

$$max\_utility\_cell = [(curr\_cell)] \tag{6.9}$$

PLANNING
This function calculates the planned_path to the target. It returns the path, that is reachable to the assumed_target.

1. planned_path = self.planning(curr_pos)
2. while(len(planned_path) == 0):
   (a) This means that the assumed_target is not reachable from the current_cell. Since the target is always reachable from the agent, it is not possible for the agent to be in this assumed_target location. Hence, we can mark this location blocked.
   (b) self.agent_grid.grid[cell] = 'X' $\longrightarrow$ Marking the cell blocked in the agent's knowledge.

   (c) UPDATE_PROBABILITY_CONTAINING_BLOCK (assumed_target) $\longrightarrow$ marks the assumed_target location to be blocked.
   (d) After failing to reach the assumed_target, the agent is required to compute the new assumed_target and hence we need to call PRE_PLANNING(curr_pos)
   (e) After getting the new assumed_target, the agent will find the new path using a-star algorithm.

3. Return planned_path $\longrightarrow$ The loop will end after the agent can compute a planned_path.

EXECUTION In this function, the algorithm will compute the next_cell that the agent can go to.

1. First, we need to compute the planned_path. Hence, **planned_path = self.planning(curr_pos)**
2. **next_cell** will be the first cell that is present after the location of the agent. Hence, **next_cell = planned_path[1]**.
3. There can be two cases:
   (a) The next_cell is an unblocked cell. If this is true, the function **return next_cell**.
   (b) If the next_cell is blocked. Then the algorithm does the following:
      i. Mark the next_cell of the grid as **blocked**. $\therefore$ **self.agent_grid.grid[cell] = X** $\longrightarrow$ Marking the next_cell as blocked.
      ii. UPDATE_PROBABILITY_CONTAINING_BLOCK(self.assumed_target) $\longrightarrow$ Update belief since we came to know that the next_cell is blocked.
      iii. MOVE_TARGET() $\longrightarrow$ Since, the agent attempted to move to the next_cell, the target will also move to the next_cell.
      iv. assumed_target = PRE_PLANNING(curr_pos) $\rightarrow$ Need to find the new assumed_target since the belief has changed.
      v. p_containing = GENERATE_PREDICTION_MATRIX() $\longrightarrow$ Since, the target has moved, the prediction will become the belief.
      vi. planned_path = PLANNING(curr_pos) $\rightarrow$ The agent will plan a new path from curr_pos. It will return a new path planned, that is reachable according to the agent's current knowledge.
      vii. next_cell = planned_path[1] $\rightarrow$ This will again fetch the next_cell in the planned_path.
4. Return next_cell $\rightarrow$ The next_cell will be the unblocked_cell which the agent

---

FIND_TARGET()
This function is the main function which uses all the above functions:

1. Initialisation:
   (a) curr_pos = start

   (b) assumed_target = pre_planning(curr_pos)

2. **while(True)**:

   (a) The agent is present in an unblocked_cell, now if the current_cell is ·, then we need to update the agent_grid as follows: $self.agent\_grid.grid[curr] = self.grid.grid[curr]$ → This will update the terrain in the agent's grid.

   (b) vicinity = SENSE(curr_pos) → The agent senses whether the target is present in the vicinity, hence this funcion updates the belief as required and tells whether the target is present in the vicinity.

   (c) If vicinity == True, then do the following:

      i. The agent will not move if the target is in the vicinity. So, the target will only move.

      ii. MOVE_TARGET()

      iii. p_containing = GENERATE_PREDICTION_MATRIX()

      iv. pre_vicinity = True

      v. **continue**

   (d) if **pre_vicinity == TRUE**:

      i. pre_vicinity = False

      ii. present = EXAMINATION_CELL(curr_pos) → The algorithm will examine the current_location for the target.

      iii. if present == TRUE:
      Means that the target is found ad the function returns.

   (e) If target is not found, then the agent will find a new assumed_target with updated belief. **assumed_target = pre_planning**

   (f) next_cell = EXECUTION(curr_pos) → This funcion will return the next_cell which is unblocked on which agent can move.

   (g) curr_pos = next_cell → The agent moves to the next_cell

   (h) p_containing = GENERATE_PREDICTION_MATRIX() → The prediction will become belief in the next time stamp if the target moves from the current cell.

   (i) MOVE_TARGET() → the function in which target will move from it's location

   (j) movement_cost += 1 → Will increase the movement_cost by 1

3. The loop will never end, until the agent finds the target. Once the agent finds the target, the function will return.

	Agent 6	Agent 7	Agent 8	Agent9
Average Examine Cost	13364.65	12594.77	11129.65	5.87
Average Movement Cost	106276.22	85982.92	19683.58	3572.17
Average Total Cost	119640.87	98577.69	30813.23	3578.04

Table 6.1: Cost: Agent 6 vs Agent 7 vs Agent 8 vs Agent 9

6.5 Analysis

Scatter Plot

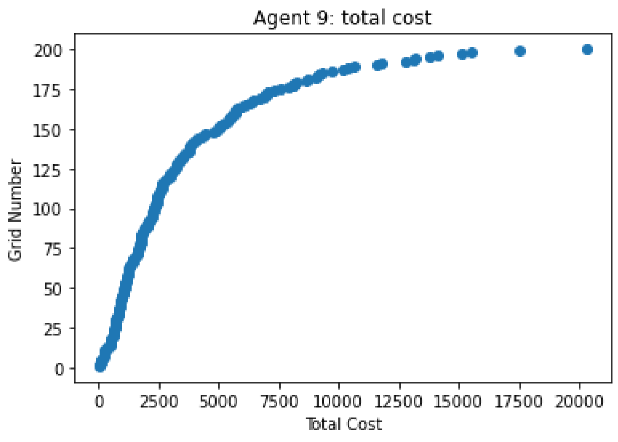
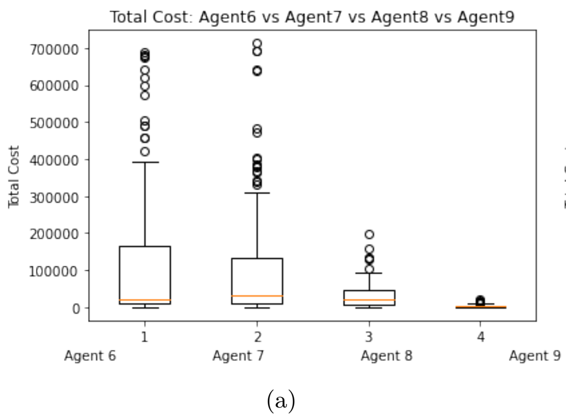


Figure 6.1: Total Cost: Agent 9

Figure 6.1 denotes the total cost of Agent 9. The maximum cost of Agent 9 is around 20,000, which is much lower than maximum value of Agent 8(2,00,000). So, agent 9 greatly outperforms Agent 8. So, if a target is moving, we can build a much efficient model than Agent 8. From the table 6.1, we can see the average total cost for Agent 9 is around 10 times lower than that of Agent 8.

Box Plot - Total Cost

From the Figure 6.2 (a), (b) we can see that the Agent 9 performance is much better than Agent 8, since the all the grids, except the outliers of box plot, are solved under the cost of around 11000 movement + examination cost. But for agent 8, the 25th percentile is around 10,000, which is approximately equal to the 100th percentile of the agent 9. This means agent 9 is solved at the total cost at which 25% of the grids are solved in agent 8. Also, Agent 9 is much efficient than Agent 6, 7.



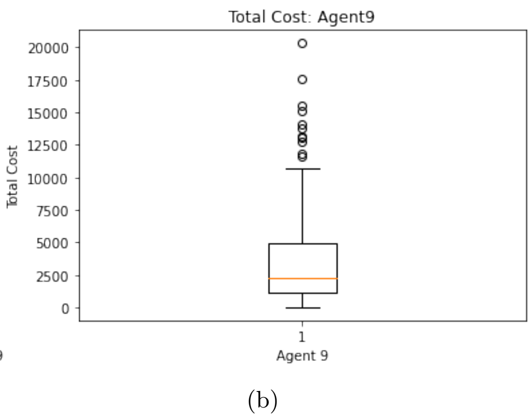
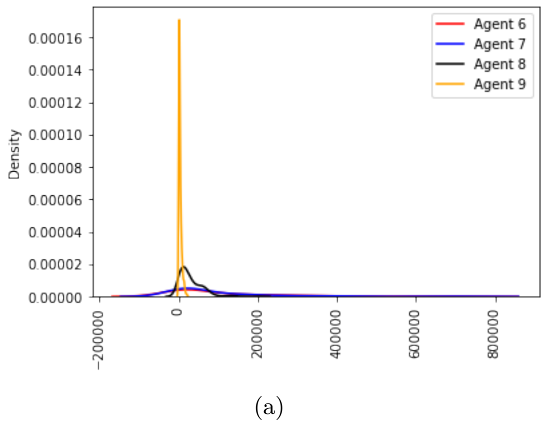


Figure 6.2: Box Plots: Total Cost (a) Agent 6 vs 7 vs 8 vs 9 (b) Agent 9



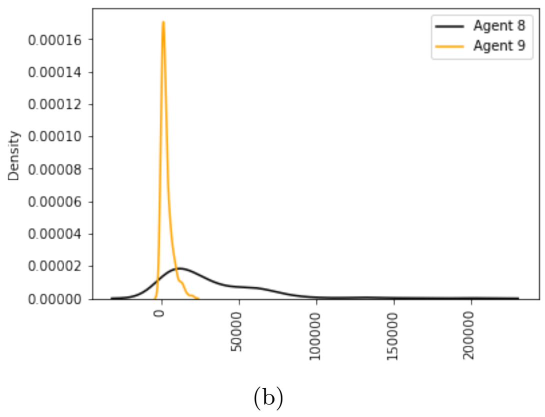
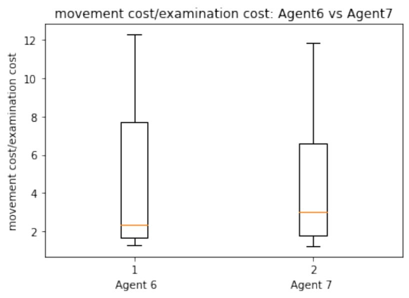


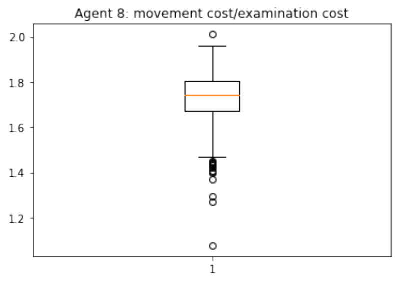
Figure 6.3: Density Plots: Total Cost (a) Agent 6 vs Agent 7 vs Agent 8 vs Agent 9 (b) Agent 8 vs Agent 9

Density Plots - Agent 6 vs Agent 7 vs Agent 8 vs Agent 9

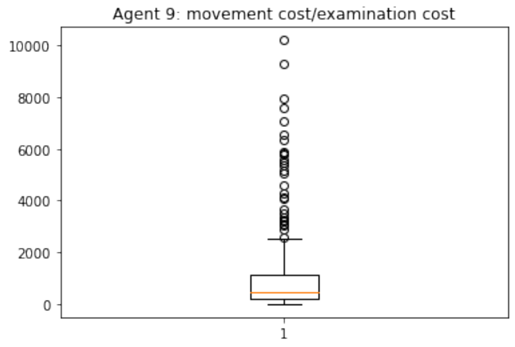
From the KDE plots in Figure 6.3 (a), (b) we can see that the density of the plots being solved at low value of movement is much higher than the density of the plots that are solved at that value in Agent 8. So, from the above plots, we can see that Agent 9 performance is much better than Agent 8.



(a)



(b)



(c)

Figure 6.4: (a) Box Plot: Agent 6 vs Agent 7: movement cost/examination cost: total cost (b) Agent 8: movement cost/examination cost (c) Agent 9: movement cost/examination cost

Ratio of Movement Cost and Examination Cost in Agent 6 vs Agent 7 vs Agent 8 vs Agent 9

From the Box plots in Fig 6.4 (a), (b) and (c) we can see that the values of ratio is much higher than that in Agent 6, Agent 7 and Agent 8, which is a good thing since in application where examination cost is much higher than movement cost, in such scenarios Agent 9 will always prove to be a better agent than Agent 8.

One of the most important reason for good performance of Agent 9 is that it has the ability to sense whether the target is present in the neighbouring cells.