2022

Project 2: Circle of Life

GYANA DEEPIKA DASARA [GD452]
TEJASHWINI VELICHETI [TV186]

Contents

1. Introduction

1.1 Problem Statement

An undirected graph of 50 nodes contains three entities - the agent, the prey, and the predator. The agent must catch the prey before the predator catches the agent. These three entities move in the graph until the game ends. There is a complete information environment (where the location of the predator and prey is known to the agent) and a partial information environment (where the agent is not always sure about the predator and prey information). We build various models (rule-based and probabilistic) to inform our decision-making process and increase our chances of winning the game.

2. Implementation

2.1 Environment Setup

This part of the implementation requires multiple steps revolving around graph creation, modelling prey and modelling predator.

2.1.1 Graph Creation

Initially, there are 50 nodes connected in a circular fashion. And then, random nodes with degree<3 are chosen, and an edge is added between this node and another random node whose degree<3 and it is within the radius of +5 to -5 of the chosen node. The process is repeated until no more edges can be added. The function *create_graph()* is used to create this graph which returns the required graph (a dictionary consisting of keys as nodes and values as the respective neighbours).

Graph Theory Question: With this setup, you can add at most 25 additional edges (why?). Are you always able to add this many? If not, what's the smallest number of edges you're always able to add?
Initially, we had created 50 nodes and had connected them in a circle like 1-2-3-.....-50-1. Within this setup, the degree of each node is already two. So every node can accommodate connecting to just one another node. When one edge is added between two nodes, the degree of both nodes gets incremented by one. After the node degree becomes 3, no more edges can be added to the node. Among 50 nodes, we can choose 25 pairs of nodes which are within the required 5+/- radius and add an edge between them. So at max we can add 50/2 = 25 undirected edges. Due to the fact that we can only add the additional edges between a node and other nodes within 5+/- steps and the randomness of choosing which node to connect to, it is not always possible to add 25 edges each time. The smallest number of edges we are able to add each time are 21. Consider, N = Nodes in a circular connected graph, here it is 50 E = random edges added in the graph 2*E = nodes with degree 3 K = nodes with degree 2 Let the arrangement be in a way that each node with degree 2 is positioned with <=5 other nodes separating them. So total number of nodes in a graph can be given by:

5*K + K <= N
6*K <= N
K<= N/6

Substituting N = 50,
K<= 8

The minimum value that K can take is 8. This says that 42 nodes in the graph were able to get connected through random edges, but 8 nodes were unable to get connected with the random edges to other nodes within +5/-5 radius. So 21 is the count of the minimum random edges that can be added to the graph. In case of avoiding duplicate random edges, this count goes further down. The maximum value that K can take is 0. That means that all 50 nodes are able to get connected by random 25 edges.

2.1.2 Initialization of Prey, Predator, and Agent

The agent is initialized first, followed by the prey, and the predator. While initializing the prey and the predator we make sure that they are not initialized in the same node as the agent. The function *initilise_prey_pred_agent()* is used for initializing the position of the prey, the predator, and the agent.

2.1.3 Modelling Prey Movements

Every time the Prey moves, it selects among its neighbours or its current cell, uniformly at random and moves to it. The prey's movements carry on this way until the game concludes. The function *move_prey()* contains the logic to move the prey and return the next move of the prey.

2.1.4 Modelling Predator Movements

2.1.4.1 Normal Predator

The strategy of the Predator is to move closer to the Agent. So, every time the Predator moves, it calculates the shortest distance to the Agent for each neighbour it can move to. It then moves to the neighbour with the shortest distance remaining to the Agent. If multiple neighbours have the same distance to the Agent, the Predator selects uniformly at random between them. The function *move_predator()* follows this logic to decide the next move of the predator.

Breadth First Search has been implemented to find the shortest distance between the agent and the predator. As the graph has a limited set of nodes and is connected, we don't run into any major efficiency-based issues while computing the shortest path using BFS. Based on this distance, the predator makes its decision.

2.1.4.2 Distracted Predator

This predator has two behaviours. With a probability of 0.6, it moves to its neighbouring cell which is closest to the agent. With a probability of 0.4, it moves randomly to any of its neighbours. The function *move_distracted_predator()* follows this logic to decide the next move of the predator.

2.2 Modelling Agents in Different Environments

The different environments that we will be modelling our Agents for are:

1. Complete Information Setting: Agent 1 and Agent 2

2. Partial Information Setting (Prey Unknown): Agent 3 and Agent 4
3. Partial Information Setting (Predator Unknown): Agent 5 and Agent 6
4. Combined Partial Information Setting (Prey and Predator Unknown): Agent 7 and Agent 8
5. Combined Partial Information Setting (Prey and Predator Unknown) with defective drone: Agent 7a, Agent 7b, Agent 8a, Agent 8b and Agent 9

2.2.1 The Complete Information Setting

The Agent is always aware of the location of the Predator and the Prey in this scenario.

2.2.1.1 Agent 1

2.2.1.1.1 Rules

Agent 1 examines each of its available neighbours and selects its next move from them in the following order while breaking ties at random.

1. Neighbours that are closer to the Prey and farther from the Predator.
2. Neighbours that are closer to the Prey and not closer to the Predator.
3. Neighbours that are not farther from the Prey and farther from the Predator.
4. Neighbours that are not farther from the Prey and not closer to the Predator.
5. Neighbours that are farther from the Predator.
6. Neighbours that are not closer to the Predator.
7. Stay in its position.

We compute the distances from the agent's current position to the prey and predator. A neighbour closer to the prey is chosen when this condition is satisfied: *distance(agent, prey)>distance(neighbour, prey)*. A neighbour farther to the predator is selected when this condition is satisfied: *distance(agent, predator)<distance(neighbour, predator)*.

For rule 1&2, if multiple neighbours satisfy the condition, the neighbour closest to the prey is chosen. If there are multiple neighbours that are closest to the prey, we break ties at random.

For rule 3, from the neighbours that satisfy the condition, we choose the neighbour that is the farthest away from the predator. We break ties at random in case there are multiple neighbours.

2.2.1.1.2 Results

Criteria	Agent 1 Results
How often does the Predator catch the Agent?	12.14%
How often does the Agent catch the Prey?	87.86%
How often does the simulation hang (no one has caught anything past the 5k step threshold)?	0%

2.2.1.2 Agent 2

2.2.1.2.1 Rules

While analysing the failure cases of Agent 1, we realize that the agent is still focussed on catching the prey even when it has the information to realize that the predator would catch the agent before the agent is able to catch the prey. The difference between Agent 2 and Agent 1 is that Agent 2 prioritizes moving away from the predator if the predator is in a position too close to the agent to avoid getting killed.

Let's take a test scenario and see how this increases the chances of Agent winning.

Given:
Agent Node has two neighbours: N1 and N2.
Agent Node: Distance(Agent, Prey) = 8 Distance(Agent, Predator) = 1
N1: Distance(N1, Prey) = 7 Distance(N1, Predator) = 1
N2: Distance(N2, Prey) = 9 Distance(N2, Predator) = 2

With the above data, Agent 1 chooses N1 but Agent2 chooses N2. As Agent1 chooses N1, we can see that the agent will be caught by the predator in the next move itself. Whereas Agent2 chooses N2, and increases its chances of survival by moving away from the predator. If the predator is less than 3 moves from the agent, Agent 2 focusses on moving away from predator than catching the prey.

Agent 2 selects its next move based on the following order of rules by breaking ties at random.

1. Neighbours that are closer to the Prey and farther from the Predator.
2. Neighbours that are farther away from Predator if the *Distance(agent, prey)> Distance(Agent, Predator)* and the *Distance(Agent,Predator) < 3* , that is the predator is less than 3 steps away from Agent.
3. Neighbours that are closer to the Prey and not closer to the Predator.
4. Neighbours that are not farther from the Prey and farther from the Predator.
5. Neighbours that are not farther from the Prey and not closer to the Predator.
6. Neighbours that are farther from the Predator.
7. Neighbours that are not closer to the Predator.
8. Stay in its position.

Step 1 is the ideal step to take. Step 2 contains the logic of prioritizing survival if agent is too close to the predator and there is a high chance of getting killed. Rest of the steps are similar to the rules of Agent1. The main purpose of adding this rule is to make sure that the Agent prioritizes to stay alive over catching the prey when the Predator is very close to the Agent.

2.2.1.2.2 Results

Criteria	Agent 2 Results
How often does the Predator catch the Agent?	1.84%
How often does the Agent catch the Prey?	98.16%
How often does the simulation hang (no one has caught anything past the 5k step threshold)?	0%

2.2.1.3 Agent 1 v/s Agent 2

Criteria	Agent 1 Results	Agent 2 Results
How often does the Predator catch the Agent?	12.14%	1.84%
How often does the Agent catch the Prey?	87.86%	98.16%
How often does the simulation hang (no one has caught anything past the 5k step threshold)?	0%	0%

We can see that there is an improvement in the success rates of Agent 1 and Agent2 by 10%. We see that Agent 2 success rate is very close to 100. This means that the idea of prioritizing moving away from the predator when its too close to the agent leads to success in most cases.

2.2.2 The Partial Prey Information Setting (Prey Location Unknown)

In this scenario, the Agent is always aware of the location of the Predator but is not aware of the location of the Prey. The agent must estimate the position of the prey through surveying and updating the beliefs.

2.2.2.1 Agent 3

2.2.2.1.1 Belief Updates

- **Initialization:** Initially, the only information we have about the prey is that it is not located in the agent's current location. Apart from that, the prey is equally likely to be present in any of the other nodes.

$$P(\text{Prey in Agent's current position}) = 0$$

$$\text{For other 49 nodes, each represented by } j\text{: } P(\text{Prey in } j) = \frac{1}{49}$$

- **Update after the survey:** When we survey the node with the maximum probability of containing the prey, we either find the prey or we don't find the prey. So we will have two cases of updates here.
  - **If survey is successful:**

$$P(\text{Prey in surveyed node}) = 1$$

$$\text{For nodes, each represented by } j\text{, where } j \text{ is not surveyed node:}$$
$$P(\text{Prey in } j) = 0$$

  - **If survey is not successful:**

$$\text{For nodes, each represented by } j\text{, where } j \text{ is not surveyed node:}$$
$$P(\text{Prey in } j \mid \text{prey not in surveyed node}) = \frac{P(\text{Prey in } j) * P(\text{Prey not in surveyed node} \mid \text{Prey in } j)}{P(\text{Prey not in surveyed node})}$$
$$P(\text{Prey in } j \mid \text{prey not in surveyed node}) = \frac{P(\text{Prey in } j)}{1 - P(\text{Prey in surveyed node})}$$

$$\text{For surveyed node:}$$
$$P(\text{Prey in surveyed node}) = 0$$

- **Update after the agent moves:** If the Agent finds the prey after it moves, the game ends. We need to update the belief when the agent did not find the prey in its new location.

$$\text{For nodes, each represented by } j\text{, where } j \text{ is not agent's new location}$$
$$\text{and } j \text{ is not previously surveyed node :}$$
$$P(\text{Prey in } j \mid \text{Prey not in agent's location}) = \frac{P(\text{Prey in } j) * P(\text{Prey not in agent's location} \mid \text{Prey in } j)}{P(\text{Prey not in agent's location})}$$
$$P(\text{Prey in } j \mid \text{Prey not in agent's location}) = \frac{P(\text{Prey in } j)}{1 - P(\text{Prey in agent's location})}$$

$$\text{For agent's location:}$$
$$P(\text{Prey in agent's location}) = 0$$

- **Update after the prey moves:** The prey can choose any of its neighbours at random to move to next or choose to remain at its position.

$$P(Prey\ in\ i\ after\ prey\ moves) = \sum P(Prey\ in\ j) * P(Prey\ moves\ from\ j\ to\ i \mid Prey\ in\ j)$$

$$where\ j\ consists\ of\ i\ and\ all\ the\ neighbours\ of\ i$$

2.2.2.1.2 Rules

1. If it is not certain about where the prey is, the agent surveys the node with the maximum belief of containing the prey.
2. Based on the survey result, the beliefs of the prey get updated.
3. Agent chooses the node with the maximum belief as the prey position and follows Agent 1 rules to decide which move to take next.
4. After the agent moves to the next position, we check if the prey/predator is present in the agent position. If the prey is not present in the agent position and the game has not yet ended, we update the prey's beliefs again with the new information that the prey is not present in the agent's position.
5. The prey moves to either of its neighbours or stays in place. We check if the prey is in the agent position, if yes, the agent wins.
6. The beliefs of prey are updated based on the transition model.
7. The predator moves, and we check if the predator catches the agent. If it catches the agent, the game ends as the agent loses, else we repeat the process from step 1 until the game ends.

2.2.2.1.3 Results

Criteria	Agent 3 Results
How often does the Predator catch the Agent?	16.96%
How often does the Agent catch the Prey?	83.03%
How often does the Agent know exactly where the Prey is during a simulation when the information is partial?	18.45%
How often does the simulation hang (no one has caught anything past the 5k step threshold)?	0%

2.2.2.2 Agent 4

2.2.2.2.1 Rules

The sequence of rules and the belief updates that Agent 4 follows and uses are similar to that of Agent 3. After finding the node that is most likely to contain the prey, Agent 4 follows the logic of Agent 2 instead of Agent 1 to decide its next move. Through this, Agent 4 also focusses on survival rather than catching the prey when there is a high risk of getting caught by the predator.

2.2.2.2.2 Results

Criteria	Agent 4 Results
How often does the Predator catch the Agent?	3.30%
How often does the Agent catch the Prey?	96.7%
How often does the Agent know exactly where the Prey is during a simulation when the information is partial?	18.57%
How often does the simulation hang (no one has caught anything past the 5k step threshold)?	0%

2.2.2.3 Agent 3 v/s Agent 4

Criteria	Agent 3 Results	Agent 4 Results
How often does the Predator catch the Agent?	16.96%	3.30%
How often does the Agent catch the Prey?	83.03%	96.7%
How often does the Agent know exactly where the Prey is during a simulation when the information is partial?	18.45%	18.57%
How often does the simulation hang (no one has caught anything past the 5k step threshold)?	0%	0%

We can see that there is an improvement in the success rates of Agent 3 and Agent 4 by 13%. As we are aware of the predator's position and since agent 4 calls agent 2, we reduce the chances of getting killed by the predator significantly. This leads to an improvement from the previous agent.

2.2.3 The Partial Predator Information Setting (Predator Location Unknown)

In this scenario, the Agent is always aware of the location of the Prey. Initially, the agent is aware of the starting location of the predator. After that, the agent does not know the location of the predator. The agent must estimate the position of the predator through surveying and updating the beliefs.

2.2.3.1 Agent 5

2.2.3.1.1 Belief Updates

- **Initialization:** Initially, we are aware of the predator's position. So we initialize the predator's position with 1 and the rest with zero.

$$P(\text{Pred in Pred's position}) = 1$$

$$\text{For other 49 nodes, each represented by } j\text{: } P(\text{Pred in } j) = 0$$

- **Update after the survey:** When we choose nodes with the maximum probability of containing the predator, and then choose the node closest to the agent to survey (breaking ties at random), we either find the prey or we don't find the prey. So, we will have two cases of updates here.
  - **If survey is successful:**

$$P(\text{Pred in surveyed node}) = 1$$
$$\text{For nodes, each represented by } j\text{, where } j \text{ is not surveyed node:}$$
$$P(\text{Pred in } j) = 0$$

  - **If survey is not successful:**

$$\text{For nodes, each represented by } j\text{, where } j \text{ is not surveyed node:}$$
$$P(\text{Pred in } j \mid \text{Pred not in surveyed node}) = \frac{P(\text{Pred in } j) * P(\text{Pred not in surveyed node} \mid \text{Pred in } j)}{P(\text{Pred not in surveyed node})}$$
$$P(\text{Pred in } j \mid \text{Pred not in surveyed node}) = \frac{P(\text{Pred in } j)}{1 - P(\text{Pred in surveyed node})}$$

$$\text{For surveyed node:}$$
$$P(\text{Pred in surveyed node}) = 0$$

- **Update after the agent moves:** If the Agent finds the predator in the cell after it moves, the game ends. We need to update the belief when the agent did not find the predator in its new location.

$$\text{For nodes, each represented by } j, \text{ where } j \text{ is not agent's new location and } j \text{ is not previously surveyed node :}$$

$$P(\text{Pred in } j \mid \text{Pred not in agent's location}) = \frac{P(\text{Pred in } j) * P(\text{Pred not in agent's location} \mid \text{Pred in } j)}{P(\text{Pred not in agent's location})}$$

$$P(\text{Pred in } j \mid \text{Pred not in agent's location}) = \frac{P(\text{Pred in } j)}{1 - P(\text{Pred in agent's location})}$$

$$\text{For agent's location:}$$

$$P(\text{Pred in agent's location}) = 0$$

- **Update after the predator moves:** The predator has two behaviours. With 0.4 probability, it can choose any of its neighbours at random to move to next or choose to remain at its position. With 0.6 probability, it chooses to move to a neighbour that is closer to the agent.

$$P(\text{Predator in } i \text{ after Predator moves}) =$$

$$\sum 0.6 * P(\text{Predator moves from } j \text{ to } i \mid \text{Predator in } j) +$$

$$\sum 0.4 * P(\text{Predator moves from } j \text{ to } i \mid \text{Predator in } j)$$

$$\text{where } j \text{ consists of and all the neighbors of } i$$

The term $0.6 * P(\text{Predator moves from } j \text{ to } i \mid \text{Predator in } j)$ only gets added when i is the neighbor of j which is closest to the agent, else $P(\text{Predator moves from } j \text{ to } i \mid \text{Predator in } j)$ becomes 0 . If j has multiple neighbors that are closest to agent then this term gets reduced to $0.6 * P(\text{Predator in } j) * \frac{1}{\text{Number of close neighbors of } j}$.

How do the probability updates for the Predator differ from the probability updates for the Prey? Be explicit and clear.

The belief updates after the survey and after the agent moves remain the same. The difference between the belief updates arises during the transition belief updates. The prey randomly chooses to move to any of its neighbours or remain in its position. However, the distracted predator showcases two behaviours - with 0.4 probability it moves randomly to any of its neighbours, with 0.6 probability it moves towards the agent.

Prey Transitional Model

$$P(\text{Prey in } i \text{ after prey moves}) = \sum P(\text{Prey in } j) * P(\text{Prey moves from } j \text{ to } i \mid \text{Prey in } j)$$

$$\text{where } j \text{ consists of } i \text{ and all the neighbours of } i$$

Pred Transitional Model

$$P(Predator\ in\ i\ after\ Predator\ moves) =$$

$$\sum 0.6 * P(Predator\ moves\ from\ j\ to\ i \mid Predator\ in\ j) +$$

$$\sum 0.4 * P(Predator\ moves\ from\ j\ to\ i \mid Predator\ in\ j)$$

$$where\ j\ consists\ of\ and\ all\ the\ neighbors\ of\ i$$

The term $0.6 * P(Predator\ moves\ from\ j\ to\ i \mid Predator\ in\ j)$ only gets added when i is the neighbor of j which is closest to the agent, else $P(Predator\ moves\ from\ j\ to\ i \mid Predator\ in\ j)$ becomes 0 . If j has multiple neighbors that are closest to agent then this term gets reduced to $0.6 * P(Predator\ in\ j) * \frac{1}{Number\ of\ close\ neighbors\ of\ j}$.

2.2.3.1.2 Rules

1. If it is not certain about where the predator is, the agent surveys the node with the maximum belief of containing the predator which is closest to the agent.
2. Based on the survey result, the beliefs of the predator get updated.
3. Agent chooses the node with the maximum belief which is closest to the agent as the predator position and follows Agent 1 rules to decide which move to take next.
4. After the agent moves to the next position, we check if the prey/predator is present in the agent position. If the prey/predator is not present in the agent position and the game has not yet ended, we update the predator's beliefs again with the new information that the predator is not present in the agent's position.
5. The prey moves to either of its neighbours or stays in place. We check if the prey is in the agent position, if yes, the agent wins.
6. The predator moves, and we check if the predator catches the agent. If it catches the agent, the game ends as the agent loses.
7. The beliefs of the predator are updated based on the transition model.
8. We repeat the process from step 1 until the game ends.

2.2.3.1.3 Results

Criteria	Agent 5 Results
How often does the Predator catch the Agent?	18.97%
How often does the Agent catch the Prey?	81.03%
How often does the Agent know exactly where the Predator is during a simulation when the information is partial?	49.36%
How often does the simulation hang (no one has caught anything past the 5k step threshold)?	0%

2.2.3.2 Agent 6

2.2.3.2.1 Rules

The sequence of rules and the belief updates that Agent 6 follows, and uses are similar to that of Agent 5. After finding the node that is most likely to contain the predator, Agent 6 follows the logic of Agent 2 instead of Agent 1 to decide its next move. Through this, Agent 6 also focusses on survival rather than catching the prey when there is a high risk of getting caught by the predator. Compared to the previous case where the prey is unknown, as we know the initial location of the

predator, we should ideally be able to predict the predator's location more accurately as we are aware of how the predator chooses to move (predator tries moving closer to agent with 0.6 probability). This greater accuracy and the logic of Agent 2 helps improve the total overall success rate.

2.2.3.2.2 Results

Criteria	Agent 8 Results
How often does the Predator catch the Agent?	11.64%
How often does the Agent catch the Prey?	88.36%
How often does the Agent know exactly where the Predator is during a simulation when the information is partial?	49.82%
How often does the simulation hang (no one has caught anything past the 5k step threshold)?	0%

2.2.3.3 Agent 5 v/s Agent 6

Criteria	Agent 5 Results	Agent 6 Results
How often does the Predator catch the Agent?	18.97%	11.64%
How often does the Agent catch the Prey?	81.03%	88.36%
How often does the Agent know exactly where the Predator is during a simulation when the information is partial?	49.36%	49.82%
How often does the simulation hang (no one has caught anything past the 5k step threshold)?	0%	0%

We can see that there is an improvement in the success rates of Agent 5 and Agent 6 by 7%. As we are aware of the starting position of the predator and it's behaviour of how it moves, we are able to identify the location of the predator with a better accuracy as we have more information. As we are calling Agent 2 in Agent 6, the survivability increases giving it more time to catch the prey.

2.2.4 The Combined Partial Information Setting

In this scenario, the agent is not aware of both the predator and prey location. Initially, the agent is aware of the starting location of the predator. After that, the agent does not know the location of the predator. The agent must estimate the position of the predator/prey through surveying and updating the beliefs.

2.2.4.1 Agent 7

2.2.4.1.1 Belief Updates

The belief updates of Agent 7 remain the same as that of Agent 3 and Agent 5.

2.2.4.1.2 Rules

1. If it is not certain about where the predator is, the agent surveys the node with the maximum belief of containing the predator which is closest to the agent (like Agent 5).
2. If it is certain about where the predator is, if it is not certain about where the prey is, the agent surveys the node with the maximum belief of containing the prey (like Agent 3).
3. Based on the survey result, the beliefs of the predator/prey get updated.
4. Agent chooses the node with the maximum predator belief which is closest to the agent as the predator position, Agent chooses the node with the maximum prey belief as the prey position and follows Agent 1 rules to decide which move to take next.
5. After the agent moves to the next position, we check if the prey/predator is present in the agent position. If the prey/predator is not present in the agent position and the game has not

yet ended, we update the predator/prey beliefs again with the new information that the predator/prey is not present at the agent's position.

6. The prey moves to either of its neighbours or stays in place. We check if the prey is in the agent position, if yes, the agent wins.
7. The beliefs of prey are updated based on the transition model.
8. The predator moves, and we check if the predator catches the agent. If it catches the agent, the game ends as the agent loses.
9. The beliefs of the predator are updated based on the transition model.
10. We repeat the process from step 1 until the game ends.

2.2.4.1.3 Results

Criteria	Agent 7 Results
How often does the Predator catch the Agent?	22.84%
How often does the Agent catch the Prey?	77.16%
How often does the Agent know exactly where the Prey is during a simulation when the information is partial?	1.78%
How often does the Agent know exactly where the Predator is during a simulation when the information is partial?	51.35%
How often does the simulation hang (no one has caught anything past the 5k step threshold)?	0%

2.2.4.2 Agent 8

2.2.4.2.1 Rules

The sequence of rules and the belief updates that Agent 8 follows, and uses are similar to that of Agent 7. After finding the node that is most likely to contain the predator and prey, Agent 8 follows the logic of Agent 2 instead of Agent 1 to decide its next move. Through this, Agent also focusses on survival rather than catching the prey when there is a high risk of getting caught by the predator.

2.2.4.2.2 Results

Criteria	Agent 8 Results
How often does the Predator catch the Agent?	15.37%
How often does the Agent catch the Prey?	84.63%
How often does the Agent know exactly where the Prey is during a simulation when the information is partial?	1.74%
How often does the Agent know exactly where the Predator is during a simulation when the information is partial?	50.96%
How often does the simulation hang (no one has caught anything past the 5k step threshold)?	0%

2.2.4.3 Agent 7 v/s Agent 8

Criteria	Agent 7 Results	Agent 8 Results
How often does the Predator catch the Agent?	22.84%	15.37%
How often does the Agent catch the Prey?	77.16%	84.63%
How often does the Agent know exactly where the Prey is during a simulation when the information is partial?	1.78%	1.74%
How often does the Agent know exactly where the Predator is during a simulation when the information is partial?	51.35%	50.96%
How often does the simulation hang (no one has caught anything past the 5k step threshold)?	0%	0%

We can see that there is an improvement in the success rates of Agent 7 and Agent 8 by 7%. As the movement of the prey is completely random where as the movement of the predator is comparatively more focused, the accuracy with our chances of finding the predator are much

higher than finding the prey. This information makes Agent 8 to perform better than Agent 7 as we call Agent 2.

2.2.5 The Combined Partial Information Setting with Defective Survey Drone

Given that the survey drone is defective, a false negative situation has been introduced. If Prey/Predator is occupying a node that is being survey, then there is a 0.1 probability that the drone returns as unoccupied.

2.2.5.1 Prior to Belief Updates

2.2.5.1.1 Agent 7A

Criteria	Agent 7A Results
How often does the Predator catch the Agent?	42.53%
How often does the Agent catch the Prey?	57.47%
How often does the Agent know exactly where the Prey is during a simulation when the information is partial?	1.86%
How often does the Agent know exactly where the Predator is during a simulation when the information is partial?	33.34%
How often does the simulation hang (no one has caught anything past the 5k step threshold)?	0%

2.2.5.1.2 Agent 8A

Criteria	Agent 8A Results
How often does the Predator catch the Agent?	38.64%
How often does the Agent catch the Prey?	61.36%
How often does the Agent know exactly where the Prey is during a simulation when the information is partial?	1.88%
How often does the Agent know exactly where the Predator is during a simulation when the information is partial?	33.01%
How often does the simulation hang (no one has caught anything past the 5k step threshold)?	0%

2.2.5.2 After Belief Updates

2.2.5.2.1 Belief Updates

Update after the survey is unsuccessful

Update for Prey:

For nodes, each represented by j, where j is not surveyed node:

$$P(\textit{Prey in } j \mid \textit{Prey not in surveyed node}) = \frac{P(\textit{Prey in } j)}{1 - 0.9 * P(\textit{Prey in surveyed node})}$$

$$P(\textit{Prey in surveyed node}) = 0.1 * P(\textit{Prey in surveyed node})$$

Update for Pred:

For nodes, each represented by j, where j is not surveyed node:

$$P(\textit{Pred in } j \mid \textit{Pred not in surveyed node}) = \frac{P(\textit{Pred in } j)}{1 - 0.9 * P(\textit{Pred in surveyed node})}$$

$$P(\textit{Pred in surveyed node}) = 0.1 * P(\textit{Pred in surveyed node})$$

The Initialization, Update when survey is successful, Update after Prey moves, Update after Predator moves, and Update after Agent moves remain same as beliefs stated in the previous Agents. Only, Update after the survey is unsuccessful changes.

2.2.5.2.2 Agent 7B

Criteria	Agent 7B Results
How often does the Predator catch the Agent?	38.28
How often does the Agent catch the Prey?	61.72%
How often does the Agent know exactly where the Prey is during a simulation when the information is partial?	1.91%
How often does the Agent know exactly where the Predator is during a simulation when the information is partial?	34.16%
How often does the simulation hang (no one has caught anything past the 5k step threshold)?	0%

2.2.5.2.3 Agent 8B

Criteria	Agent 8B Results
How often does the Predator catch the Agent?	34.14%
How often does the Agent catch the Prey?	65.86%
How often does the Agent know exactly where the Prey is during a simulation when the information is partial?	1.92%
How often does the Agent know exactly where the Predator is during a simulation when the information is partial?	34.18%
How often does the simulation hang (no one has caught anything past the 5k step threshold)?	0%

2.2.5.3 Comparison of performance of Agents Before and After Updating the Beliefs.

2.2.5.3.1 Agent 7A v/s Agent 7B

Criteria	Agent 7A Results	Agent 7B Results
How often does the Predator catch the Agent?	42.53%	38.28
How often does the Agent catch the Prey?	57.47%	61.72%
How often does the Agent know exactly where the Prey is during a simulation when the information is partial?	1.86%	1.91%
How often does the Agent know exactly where the Predator is during a simulation when the information is partial?	33.34%	34.16%
How often does the simulation hang (no one has caught anything past the 5k step threshold)?	0%	0%

2.2.5.3.2 Agent 8A v/s Agent 8B

Criteria	Agent 8A Results	Agent 8B Results
How often does the Predator catch the Agent?	38.64%	34.14%
How often does the Agent catch the Prey?	61.36%	65.86%
How often does the Agent know exactly where the Prey is during a simulation when the information is partial?	1.88%	1.92%
How often does the Agent know exactly where the Predator is during a simulation when the information is partial?	33.01%	34.18%
How often does the simulation hang (no one has caught anything past the 5k step threshold)?	0%	0%

2.2.5.3.3 Observations

Before updating the belief for the defective drone, we can see that the success rate of Agent 7a and Agent 8a is at 57.47% and 61.72% respectively. After updating the belief, the success rates changed to 61.36% and 65.86% respectively. Approximately we see a 4% improvement, which is not very substantial. As the false negative rate is very low, in most cases the drone returns the correct observations. Hence there isn't a significant improvement between prior and after modification of the belief updates.

2.2.5.4 Agent 9

Agent 9 has to deal with an environment where it is not aware of the position of the prey and the predator (except for the initial position of the predator). It also needs to deal with the defective drone.

2.2.5.4.1 Approach

If we go over the rules of Agent 7/8, we see that the agent only uses the node with the maximum probability of containing the predator/prey to plan its next move according to rules of Agent 1/2. It ignores the information given by other nodes whose probability of containing the predator/prey which are slightly lesser than the max belief node.

This information that other nodes provide can be helpful to identify risk areas within the graph. So even if the drone reported a negative result when the predator is indeed occupying the node, considering the other nodes will reduce the effect of the false information . Once these risk areas are identified, the agent can use it to plan a path towards the prey avoiding these areas. The nodes with low probabilities of containing the predator can be considered as 'safe zones' where the agent can move towards to escape the risk areas and increase its chances of survival.

Algorithm:

1. Surveying for the predator/prey if the agent is not certain about their positions.
2. Updating the predator/prey beliefs
3. Choose top k nodes with high probability of containing the predator. Mark these nodes and their neighbourhood as high-risk areas.
4. Choose top k nodes with lowest probability of containing the predator. Mark these nodes and their neighbourhood as low risk areas.
5. Choose the node with the maximum belief of containing the prey. Find a neighbour of the agent that provides a path to this node through the low-risk area. If a path is not possible through the low-risk area towards the prey, choose a path that avoids the risk area the best.
6. Move to the best neighbour, Update the prey/predator beliefs after the agent moves.
7. Move the prey, update the prey probabilities.
8. Move the pred, update the predator probabilities.
9. Repeat these steps until the game concludes.

2.2.6 Bonus

In the Partial Information Settings - suppose that whenever it is the Agent's turn, the Agent must make a choice to move or to survey, instead of doing both. How should the Agent decide which to do, and once the decision is made, how should the Agent decide what to do (move to take or node to survey)? Implement this and compare its performance to the above.

In the partial information setting, the agent is aware about the starting location of the predator but has no information about the prey. The agent also knows that the movement of the predator will be directed towards the agent with 0.6 probability.

The agent at every timestep can either survey or move. If the agent is sure about the predator's location or at least the area in which the predator is present, it can make its decision based on that information.

We can use the ideas of maintaining belief updates for where the predator and prey are and make the decision to either move or survey based on the below logic. We can prioritize escaping from the predator than finding the prey as we can predict the location and the movement of the predator much better than that of the prey. Then, we can focus on finding the prey using surveys.

*Decision to **move** instead of survey:*

1. *If the agent knows that the predator is very close by, the agent should choose to move away from the predator. The agent must choose a neighbour that is away from the nodes with the highest probabilities of containing the predator.*
2. *If the agent knows with a high probability that the prey is in one of its neighbours and the predator is relatively far away, then the agent can move to one of its neighbours in the hope of finding the prey.*

*Decision to **survey** instead of move:*

1. *If the agent knows that the predator is far from the agent's current location, the agent can focus on finding the prey. So, the agent can choose to survey the node with max probability of containing the prey to get information about the prey's location.*
2. *If the agent has no idea about where either the prey or predator is, it can choose to survey one of the nodes with a likely chance of containing the predator so that it gains more information about where to move next.*

Risks associated with Survey and Move Choices :

1. *The risk associated with surveying is that each time the agent stays in place instead of moving, the predator moves closer to the agent. This increases the risk of being caught by the predator.*
2. *As the prey, predator locations are not constant with time, if we choose to move each time, then it won't give us much information about where the prey and predator are actually present.*
3. *So based on the above rules, the agent should strategically decide to either move or survey.*