CS 520: Probabilities

Project 2: The Circle of Life

Prepared by:

dm1639: Darshee Machhar
Rutgers University

pn266: Pankti Nanavati
Rutgers University

16th November, 2022

Contents

1 Introduction **3**
- 1.1 Problem Statement- The Game . . . 3
- 1.2 Problem Environment . . . 3
- 1.3 The Circular Forest . . . 3
- 1.4 The Prey . . . 4
- 1.5 The Predator . . . 4
- 1.6 The Distracted Predator . . . 4

2 The Complete Information Setting **5**
- 2.1 Agent 1 . . . 5
  - 2.1.1 Analysis . . . 6
- 2.2 Agent 2 . . . 7
  - 2.2.1 Analysis . . . 7
  - 2.2.2 Comparison . . . 8

3 The Partial Prey Information Setting **9**
- 3.1 Agent 3 . . . 9
  - 3.1.1 Analysis . . . 12
- 3.2 Agent 4 . . . 13
  - 3.2.1 Analysis . . . 13
  - 3.2.2 Comparison . . . 14

4 The Partial Predator Information Setting **15**
- 4.1 Agent 5 . . . 15
  - 4.1.1 Analysis . . . 17
- 4.2 Agent 6 . . . 18
  - 4.2.1 Analysis . . . 18
  - 4.2.2 Comparison . . . 19

5 The Combined Partial Information Setting **20**
- 5.1 Agent 7 . . . 20
  - 5.1.1 Analysis . . . 20
- 5.2 Agent 8 . . . 22
  - 5.2.1 Analysis . . . 22
  - 5.2.2 Comparison . . . 23

6 The Faulty Survey Drone **24**
- 6.1 Re-implementation of Agent 7 without updating belief states . . 24
- 6.2 Re-implementation of Agent 8 without updating belief states . . 24
- 6.3 Re-implementation of Agent 7 updating the belief states . . . 25
- 6.4 Re-implementation of Agent 8 updating the belief states . . . 26
- 6.5 Agent 9 . . . 27
- 6.6 Analysis . . . 28

7 Bonus **30**
7.1 Implementation of Agent 10 30
7.2 Analysis 31
7.3 Comparison of Performance 32

1 Introduction

1.1 Problem Statement- The Game

Within a circular graph, some nodes of which are occupied by a prey or a predator, an agent will try to capture the prey and at the same time evade the predator. If the agent can catch the prey, it wins but if the predator get the agent instead, the agent dies. The agent doesn't always know where the prey and predator reside within the graph, we have to build and maintain probabilistic models to help the agent deliberate their positions and make informed decisions and win the game.

1.2 Problem Environment

The environment consists, of a circular graph in which we add edges to make the graph more connected. The prey and predator are two separate entities, that are completely independent of each other. They can reside in the same spot and not harm each other.

1.3 The Circular Forest

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? We ran multiple simulations and the maximum number of nodes we could add was 25 nodes and the minimum number of nodes that were added was 18. The reason for the maximum of nodes is, there are 50 nodes, so if meticulously chosen, we can up to 50/2= 25 nodes.

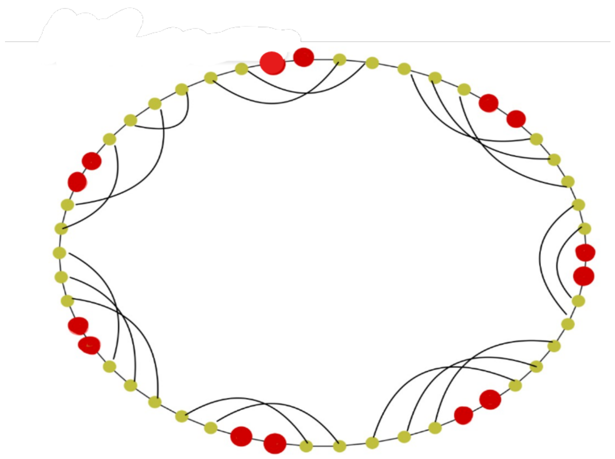


Figure 1: Minimum edges in the graph

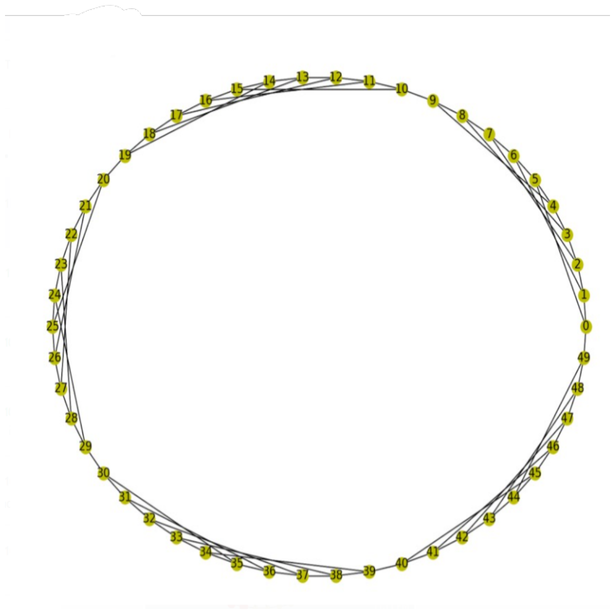


Figure 2: Maximum edges in the graph

1.4 The Prey

The prey moves randomly. It can move to any of its neighbours or continue to inhabit the same node with equal probability.

1.5 The Predator

The predator is more determined. It always moves in the direction that gets it closer to the agent. It finds the shortest path to the agent and takes the next step along that path, if there are multiple shortest paths, it chooses uniformly at random from amongst them. Agents 1,2,3 and 4 try to evade this predator.

1.6 The Distracted Predator

Agents 5 onwards, we have to evade a distracted predator. This predator doesn't always move along the shortest path to the agent but instead it can go to any of its neighbours uniformly at random with 40% chance or go to the shortest path node with 60% chance.

2 The Complete Information Setting

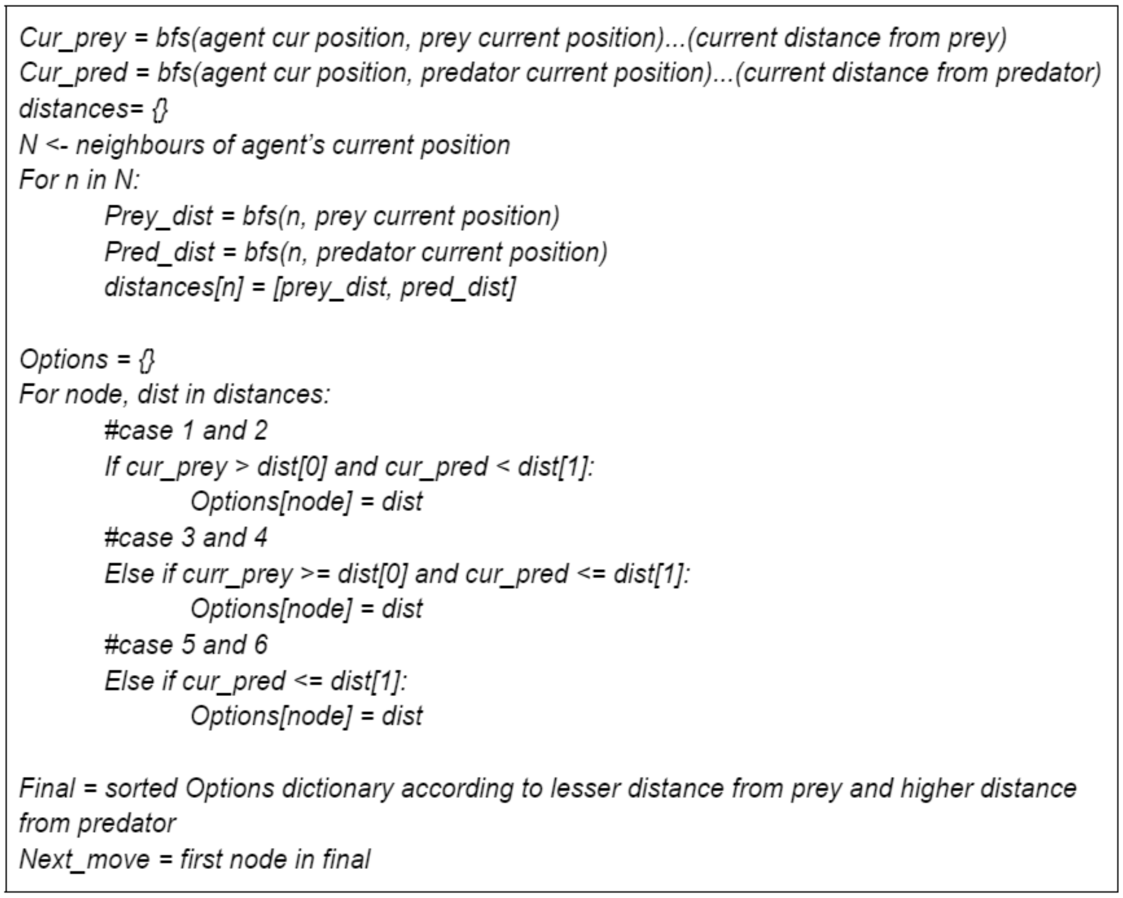
In this setting, the agent is completely aware of where the prey and predator are at all times. It makes a move by trying to stay as far away from the predator as possible and try to chase the prey as quickly as possible.

2.1 Agent 1

We built an agent that calculates its neighbour's (next possible node) distance from the prey and predator at every timestep and compares it to its current distance from the prey and predator. We now try to minimize the distance between the prey and the agent and maximise that between the agent and predator.
We have been given a set of conditions to follow in to order our agent's priorities while it makes it move viz.

- Neighbors that are closer to the Prey and farther from the Predator.
- Neighbors that are closer to the Prey and not closer to the Predator.
- Neighbors that are not farther from the Prey and farther from the Predator.
- Neighbors that are not farther from the Prey and not closer to the Predator.
- Neighbors that are farther from the Predator.
- Neighbors that are not closer to the Predator.
- Sit still and pray.

This is the psuedocode that mirrors these conditions that we have implemented:



```
Cur_prey = bfs(agent cur position, prey current position)...(current distance from prey)
Cur_pred = bfs(agent cur position, predator current position)...(current distance from predator)
distances= {}
N <- neighbours of agent's current position
For n in N:
        Prey_dist = bfs(n, prey current position)
        Pred_dist = bfs(n, predator current position)
        distances[n] = [prey_dist, pred_dist]

Options = {}
For node, dist in distances:
        #case 1 and 2
        If cur_prey > dist[0] and cur_pred < dist[1]:
                Options[node] = dist
        #case 3 and 4
        Else if curr_prey >= dist[0] and cur_pred <= dist[1]:
                Options[node] = dist
        #case 5 and 6
        Else if cur_pred <= dist[1]:
                Options[node] = dist

Final = sorted Options dictionary according to lesser distance from prey and higher distance
from predator
Next_move = first node in final
```

2.1.1 Analysis

The simulations are run 100 times for 30 different graphs and these are the results:

(a) How often the Predator catches the Agent?
Answer- 313 times out of 3000 times (10.43%) the predator catches the agent i.e., the number of times the agent loses the game

(b) How often the Agent catches the Prey?
Answer- 2687 times out of 3000 times (89.57%) the agent catches the prey i.e., the number of times the agent wins the game

(c) How often the simulation hangs (no one has caught anything past a certain large time threshold)?
Answer- The simulation hangs 0 times. (Threshold=5000 as mentioned by TA)

2.2 Agent 2

In order to improve the survivability than that of Agent 1, we made a Agent 2 that prioritizes its movements based on staying away from the predator more than getting nearer to the prey.
Especially if the predator is very close to the agent, the agent will only focus on getting a safe distance away from the predator, without paying attention to the prey's position.

2.2.1 Analysis

The simulations are run 100 times for 30 different graphs and these are the results:

(a) How often the Predator catches the Agent?
Answer- 147 times out of 3000 times (4.90%) the predator catches the agent i.e., the number of times the agent loses the game

(b) How often the Agent catches the Prey?
Answer- 2853 times out of 3000 times (95.10%) the agent catches the prey i.e., the number of times the agent wins the game

(c) How often the simulation hangs (no one has caught anything past a certain large time threshold)?
Answer- The simulation hangs 0 times. (Threshold=5000 as mentioned by TA)

2.2.2 Comparison

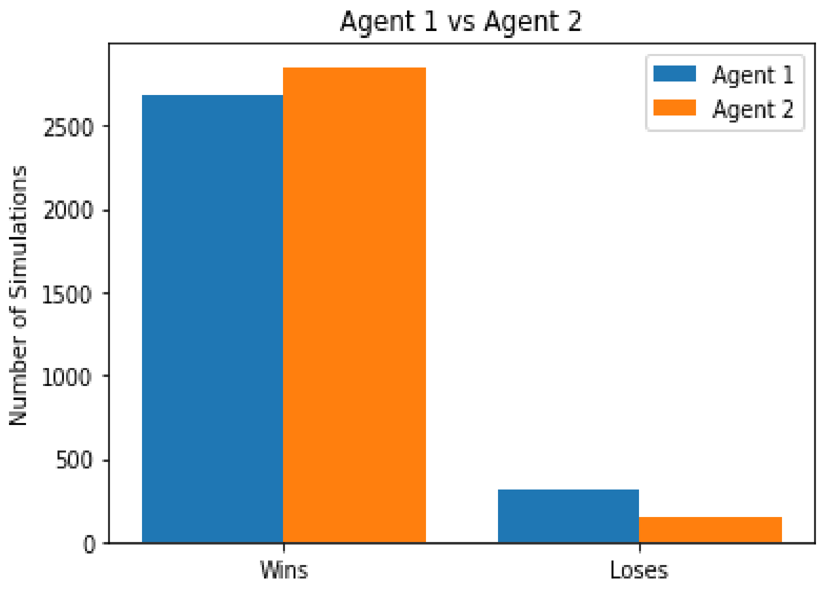


Figure 3: Agent 1 vs Agent 2

As seen from the figure 3, Agent 2 performs better in comparison to Agent 1 since Agent 2 prioritizes its movements based on staying away from the predator when the predator is one step away from the agent. This leads to less frequent times predator catching the agent.

3 The Partial Prey Information Setting

In this setting, the agent is not completely aware of his environment. While he does always know where the predator is, the prey's position is unknown.

3.1 Agent 3

The agent tries to track down the prey by surveying a node in the graph randomly and trying to build a belief system of where the prey could be. In subsequent attempts, the agent surveys the node with the highest probability of containing the prey.
Once the agent does find the prey through surveillance, we need to propagate the new probabilities according to how the prey moves.
Beyond this, agent 3 will employ the strategies of agent 1 to move towards its best guess for the prey and stay away from the predator.

Q. In these cases, how should you be updating the probabilities of where the Prey is? Be explicit and clear. There is a right answer to this, and there is definitely a wrong answer that every year proves very popular.

Logic Behind building the Belief System:
We are maintaining two belief states, at timestep and timestep t+1 or a current belief state and a future (one step into the future) belief state. When the agent or prey moves or when we survey a node, we gain some additional information about the environment which helps us modify and update the probabilities in our belief state.
At the outset of the game, we initialise our current belief state with equal probability for all nodes to contain the prey except the node being occupied by the agent, because the prey and agent can't spawn at the same position.
Say we have 5 nodes only, and the agent is at node 1, our current belief state will be this: [1: 0, 2: $\frac{1}{4}$, 3: $\frac{1}{4}$, 4: $\frac{1}{4}$, 5: $\frac{1}{4}$]
As the game goes on we can gain some information about the location of the prey by-

(a) Surveying a node

(b) Moving the agent to a new node

When we Survey:
For Current Belief State-
When we survey a node and find the prey, the probability of that node becomes 1 and the rest of the nodes have a probability of 0. Obviously because the prey can not be in more than one node at any particular timestep.
Say we survey node 2 and find the prey: Current Belief State = [1: 0, 2: 1 , 3: 0, 4: 0 , 5: 0]
Upon surveying, if we don't find the prey, we have to make the probability of

occupying the prey 0 and the rest of the probabilities will get updated.
We update the probabilities using conditional probability.
Let A be the probability of the prey being at node A
Let B be the probability of the prey being at node B
Therefore the probability of the prey being at node a, given that we surveyed b and did not find the prey there, can be represented as: P(A | not B)
According to the conditional probability rule:
P(x | y) = [P (x) . P (y | x)]/ P (y)
We get: P(A | not B) = P(A).P(not B|A)/P(not B) =P(A).P(not B—A)/(1-P(B))
Also P(not B|A) is always going to be 1, because the prey can only be at one node at a time.
Therefore the probability of it not being at B, given that it is at A, is always true, always 1.
Therefore; **P(A | not B) = P(A)/ (1-P(B))**

For Example:
4 Nodes: A, B, C, D
P(in A) = 0.4
P(in B) = 0.3
P(in C) = 0.2
P(in D) = 0.1

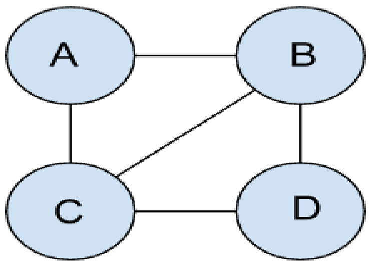
We survey B, prey isn't there
P(in A | prey not in B)
= P(in A) P(prey not in B | in A) / P(prey not in B)
= 0.4 * 1 / (1 - P(in B))
= 0.4/ (1- 0.3)
= 0.4 / 0.7

P(in A | prey not in B) = 0.4 / 0.7
P(in B | prey not in B) = 0
P(in C | prey not in B) = 0.2 / 0.7
P(in D | prey not in B) = 0.1 / 0.7

Similarly, when our agent moves to its new position and the game doesn't end, i.e. there is no prey at that node the agent moved to. We can make similar changes again to the current belief state.
Now we must observe that all of these calculations were for the current state, but when the agent must choose the most probable prey position to move towards it, he can not use the current state as is, because the prey will move in the next timestep. Hence we maintain the future belief state. For this we modify probabilities according to the prey's transitions as will be explained in the next section.

When we transition:

The prey can move to any of its neighboring nodes or continue occupying the current node with equal probability. That needs to be reflected when we propagate the probabilities of the current belief state to the future belief state.
We need to find the P(A at T= t+1) depending on the probabilities of its neighbours at T= t
Say B,C,D are the neighbours of A in this way:



And these are their probabilities as we'd calculated above:
P(in A) = 4/7
P(in B) = 0
P(in C) = 2/7
P(in D) = 1/7

We make use of Marginalization here -
P(Prey in A Next) = P(Prey in A Next, Prey in A Now)
+P(Prey in A Next, Prey in B Now)
+P(Prey in A Next, Prey in C Now)
+P(Prey in A Next, Prey in D Now)

Therefore we can apply conditional factoring to this and get:
P(Prey in A Next) = P(Prey in A Now) * P(Prey in A Next — Prey in A Now)
P(Prey in B Now) * P(Prey in A Next — Prey in B Now)
P(Prey in C Now) * P(Prey in A Next — Prey in C Now)
P(Prey in D Now) * P(Prey in A Next — Prey in D Now)

Where P(Prey in A Next — Prey in A Now) is basically referring to the one divided by degree of node A plus one [1/(degree + 1)], if A has 2 neighbours, the prey is (100/3) = 33% of its current probability likely to be at its neighbouring node or at the same position in the next timestep.

If a node is not connected to the node in question, like node D in the example above:
P(Prey in A Next — Prey in D Now) will be 0. Because node D's probabilities

won't propagate to node A.
therefore , P(Prey in A Next) = (4/7) (1/3) + (0) (1/4) + (2/7) (1/4) + (1/7)
= 4/21 * 1/14
= 0.261

This is how we update our current belief state and future belief state, to help the agent have a comprehensive understanding of its environment.

3.1.1 Analysis

(a) How often the Predator catches the Agent?
Answer- 639 times out of 3000 times (21.30%) the predator catches the agent i.e., the number of times the agent loses the game

(b) How often the Agent catches the Prey?
Answer- 2361 times out of 3000 times (78.70%) the agent catches the prey i.e., the number of times the agent wins the game

(c) How often the simulation hangs (no one has caught anything past a certain large time threshold)?
Answer- The simulation hangs 0 times. (Threshold=5000 as mentioned by TA)

(d) How often the Agent knows exactly where the Prey is during a simulation where that information is partial?
Answer- 10.6712% of times the agent knows exactly where the Prey is during a simulation.

3.2 Agent 4

In order to improve the performance of agent 3 with respect to surveying the prey more successfully, we implemented a tweak in how to choose the best node to survey.
Instead of just going for the node with the highest probability, we go for one whose probability has increased the most in one timestep (which could also be the node with the highest probability)

For example:
Current Belief State = A: 0.4, B: 0.3, C: 0.2, D: 0.1
Future Belief State = A: 0.6, B: 0, C: 0, D: 0.4

Here even though node A has the highest probability of having the prey, we scout the D node, since its probability increased the most from the previous timestep.
This logic proves to give better results, since it intuitively is veering towards the possible latest movement of the prey.

To improve the survivability of the agent we have also prioritised keeping away from the predator instead of prioritized following the prey.

3.2.1 Analysis

(a) How often the Predator catches the Agent?
Answer- 561 times out of 3000 times (18.70%) the predator catches the agent i.e., the number of times the agent loses the game

(b) How often the Agent catches the Prey?
Answer- 2439 times out of 3000 times (81.30%) the agent catches the prey i.e., the number of times the agent wins the game

(c) How often the simulation hangs (no one has caught anything past a certain large time threshold)?
Answer- The simulation hangs 0 times. (Threshold=5000 as mentioned by TA)

(d) How often the Agent knows exactly where the Prey is during a simulation where that information is partial?
Answer- 11.236% of times the agent knows exactly where the Prey is during a simulation.

3.2.2 Comparison

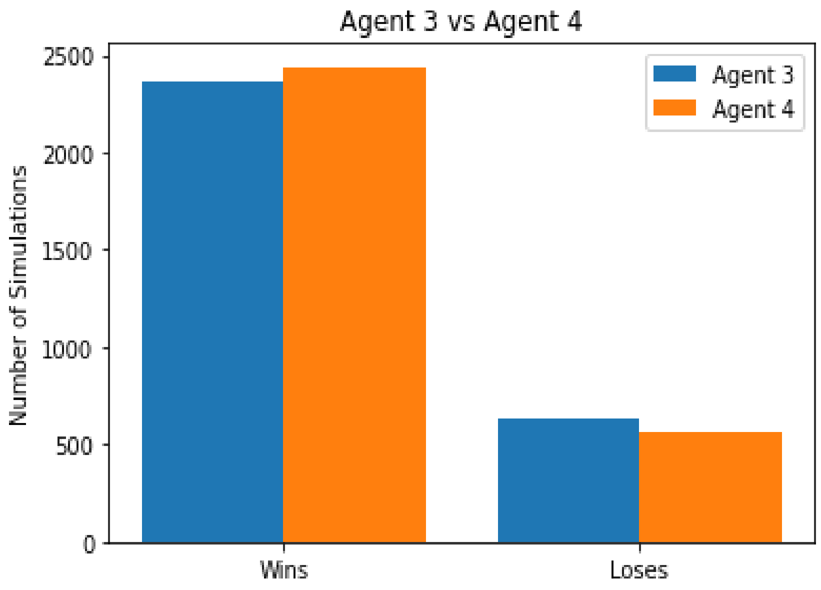


Figure 4: Agent 3 vs Agent 4

As seen from the figure 4, Agent 4 performs better in comparison to Agent 3 since Agent 4 chooses the node to survey from the belief state based on whose probability has increased the most in the last time stamp in addition to moving away from the predator if predator is one step away from the agent.

4 The Partial Predator Information Setting

In this partial information setting, instead of not knowing the position of the prey, the agent is unaware of the predator. The agent always knows where the prey is and it also knows the predator's position in the beginning of the game. But here the predator that the agent is up against is the distracted predator, whose movements are not as predictable as the original predator.

4.1 Agent 5

The agent tries to locate the predator by surveying a node in the graph with the highest probability of containing the predator. Once the agent does find the predator through surveillance, we need to propagate the new probabilities according to how the distracted predator moves.

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

Logic Behind building the Belief System:
Similar to Agent 3 with the prey, we are maintaining two belief states, current belief state and a future belief state. We gain some additional information about the environment to modify and update the probabilities in our belief state by surveying the nodes and by default when the agent moves.

The Differences between our approach:

(a) **How we initialise the belief state:**
At the outset of the game, unlike the prey, we know where the predator is. And it is the distracted predator. Hence we initialise our current belief state with probability = 1 for the predator node and zero for all the other positions.
Say we have 5 nodes only, and the predator is at node 1, our current belief state will be this: [1: 1, 2: 0 , 3: 0, 4: 0 , 5: 0]

(b) **How we change the probabilities during transistion**
When we survey the the graph for the predator, we update the belief states with the same logic as we have seen earlier with agent 3 and the prey. Just to reiterate:
When we survey a node and find the predator, the probability of that node becomes 1 and the rest of the nodes have a probability of 0.
Upon surveying, if we don't find the predator, we have to make the probability of occupying the prey 0 and the rest of the probabilities will get updated.

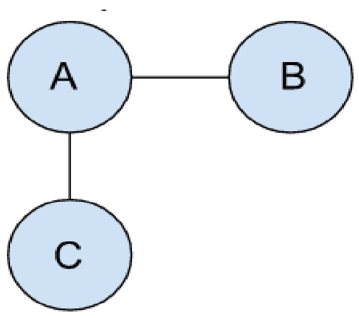
We use this equation for the update operations: **$(A \mid \text{not } B) = P(A)/(1\text{-}P(B))$** Where A is the probability of the predator being at node A, And B is the probability of the predator being at node B

Similarly, when our agent moves to its new position and the game doesn't end, i.e. there is no predator at that node the agent moved to. We can make similar changes again to the current belief state.

The differences come into the picture when we have to modify probabilities according to the predator's movements which is different from that of the prey.

When we transition:
The distracted predator doesn't stay put. It can move to any of its neighboring nodes uniformly randomly with a 0.4 probability. And it can move to the node nearest to the agent with 0.6 probability. That needs to be reflected when we propagate the probabilities of the current belief state to the future belief state.

We need to find the P(A at T= t+1) depending on the probabilities of its neighbours at T= t Say B,C are the neighbours of A in this way:



We make use of Marginalization here -
*Unlike the prey, the predator can't continue to occupy the same node, so the P(prey in A now) won't factor in here.

P(Prey in A Next) = 0.4 * P(Prey in A Next, Prey in B Now) +0.4* P(Prey in A Next, Prey in C Now) + 0.6* P(Prey in B now) * SP(A, B) + 0.6* P(Prey in C now) * SP(A, C)

Where SP(X, Y) is the number of shortest paths from Y to the agent that passes through X divided by the total number of shortest paths from Y. (where SP stands for shortest path)

Therefore we can apply conditional factoring to this and get:
P(Prey in A Next) = 0.4* P(Prey in B Now) * P(Prey in A Next | Prey in

B Now) + 0.4* P(Prey in C Now) * P(Prey in A Next | Prey in C Now) +0.6* P(Prey in B now) * SP(A, B) + 0.6* P(Prey in C now) * SP(A, C)

Where P(Prey in A Next | Prey in A Now) is basically referring to the degree of node A, if A has 3 neighbours, the prey is 33% of its current probability likely to be at its neighbouring node or at the same position in the next timestep.

This is how we differ from agent 3 while we update our current belief state and future belief state, to help the agent have a comprehensive understanding of its environment.

(c) **Breaking ties**
We break ties for predator by choosing the node that is the closest to the agent unlike the prey where we break ties randomly.

4.1.1 Analysis

(a) How often the Predator catches the Agent?
Answer- 519 times out of 3000 times (17.30%) the predator catches the agent i.e., the number of times the agent loses the game

(b) How often the Agent catches the Prey?
Answer- 2481 times out of 3000 times (82.70%) the agent catches the prey i.e., the number of times the agent wins the game

(c) How often the simulation hangs (no one has caught anything past a certain large time threshold)?
Answer- The simulation hangs 0 times. (Threshold=5000 as mentioned by TA)

(d) How often the Agent knows exactly where the Predator is during a simulation where that information is partial?
Answer- 29.343% of times the agent knows exactly where the Prey is during a simulation.

4.2 Agent 6

Our Agent 6 mirrors the behaviour of Agent 4.
Similar to Agent 4, we improve the performance of agent 5 with respect to surveying the predator more successfully, we implemented a tweak in how to choose the best node to survey.
Instead of just going for the node with the highest probability, we go for one whose probability has increased the most in one timestep (which could also be the node with the highest probability).
And in order to increase the survivability, we prioritise staying away from the predator over getting closer to the prey.

4.2.1 Analysis

(a) How often the Predator catches the Agent?
Answer- 391 times out of 3000 times (13.03%) the predator catches the agent i.e., the number of times the agent loses the game

(b) How often the Agent catches the Prey?
Answer- 2609 times out of 3000 times (86.97%) the agent catches the prey i.e., the number of times the agent wins the game

(c) How often the simulation hangs (no one has caught anything past a certain large time threshold)?
Answer- The simulation hangs 0 times. (Threshold=5000 as mentioned by TA)

(d) How often the Agent knows exactly where the Predator is during a simulation where that information is partial?
Answer- 31.07% of times the agent knows exactly where the Prey is during a simulation.

4.2.2 Comparison

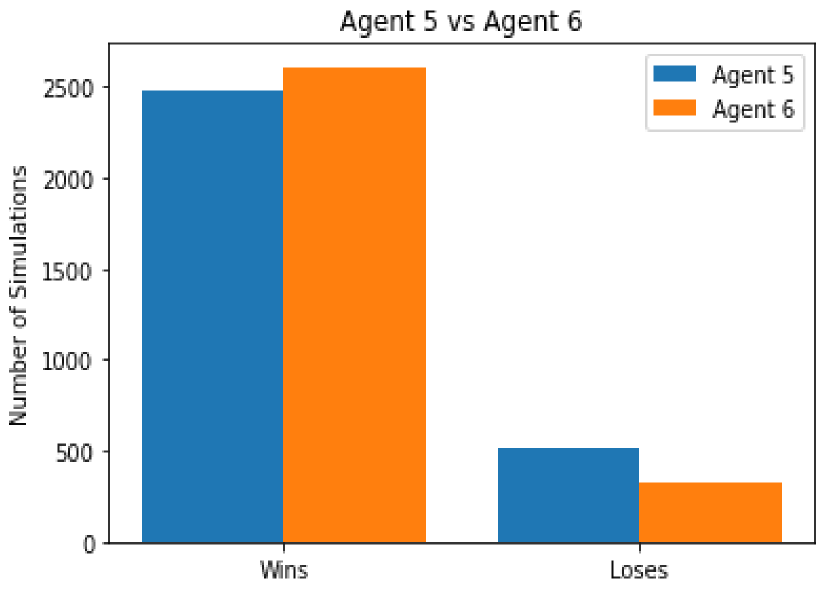


Figure 5: Agent 5 vs Agent 6

As seen from the figure 5, Agent 6 performs better in comparison to Agent 5 since Agent 6 chooses the node to survey from the belief state based on whose probability has increased the most in the last time stamp in addition to moving away from the predator if predator is one step away from the agent.

5 The Combined Partial Information Setting

So far we knew either the location of the prey, predator or both. But now we are presented with an information setting, where we know neither. To make it fair for the agents though, we do know where the predator is in the beginning of the game, but its the distracted predator.

5.1 Agent 7

To build agent 7, we borrowed from agent 3 and agent 5. Since we don't know where either the prey or predator lies, we need to maintain two separate belief systems for the prey and the predator.

So we maintain a total of four belief states:
Current Belief State for Prey
Future Belief State for Prey
Current Belief State for Predator
Future Belief State for Predator

Surveying:
Since we can survey only once, i.e. we can't survey two separate nodes, one for the prey and one for the predator. So what we decided to do was, survey with respect to the belief state of the predator only. But if we know where the predator is, we can survey for the prey.
Our reasons for this are as follows:

1. The belief state of predator is more dense than that of the prey (which generally speaking is more sparse before the prey is found via surveying the first time)
2. Hence, surveying would be more successful, if we were looking for predator rather than looking for prey.
3. Intuitively, we can claim that as long as we stay away from the predator, we will always find the prey.
4. Once we know for sure where the predator is, i.e. P(Predator at Node X) = 1, we don't need to survey for predator anymore, we can scout for the prey.

After surveying, we then follow the rules of agent 3 to update belief state for prey and we follow the rules of agent 5 for updating the belief states of the predator.

5.1.1 Analysis

(a) How often the Predator catches the Agent?
Answer- 864 times out of 3000 times (28.80%) the predator catches the agent i.e., the number of times the agent loses the game

(b) How often the Agent catches the Prey?
Answer- 2136 times out of 3000 times (71.20%) the agent catches the prey i.e., the number of times the agent wins the game

(c) How often the simulation hangs (no one has caught anything past a certain large time threshold)?
Answer- The simulation hangs 0 times. (Threshold=5000 as mentioned by TA)

(d) How often the Agent knows exactly where the Prey is during a simulation where that information is partial?
Answer- 3.031% of times the agent knows exactly where the Prey is during a simulation.

(e) How often the Agent knows exactly where the Predator is during a simulation where that information is partial?
Answer- 26.14% of times the agent knows exactly where the Predator is during a simulation.

5.2 Agent 8

In the same vein, agent 8 borrows from agent 6.
To reiterate, we improve the performance of agent 7 with respect to surveying the predator more successfully, we implemented a tweak in how to choose the best node to survey.
Instead of just going for the node with the highest probability, we go for one whose probability has increased the most in one timestep (which could also be the node with the highest probability)

5.2.1 Analysis

(a) How often the Predator catches the Agent?
Answer- 748 times out of 3000 times (24.93%) the predator catches the agent i.e., the number of times the agent loses the game

(b) How often the Agent catches the Prey?
Answer- 2252 times out of 3000 times (74.07%) the agent catches the prey i.e., the number of times the agent wins the game

(c) How often the simulation hangs (no one has caught anything past a certain large time threshold)?
Answer- The simulation hangs 0 times. (Threshold=5000 as mentioned by TA)

(d) How often the Agent knows exactly where the Prey is during a simulation where that information is partial?
Answer- 3.11% of times the agent knows exactly where the Prey is during a simulation.

(e) How often the Agent knows exactly where the Predator is during a simulation where that information is partial?
Answer- 27.82% of times the agent knows exactly where the Predator is during a simulation.

5.2.2 Comparison

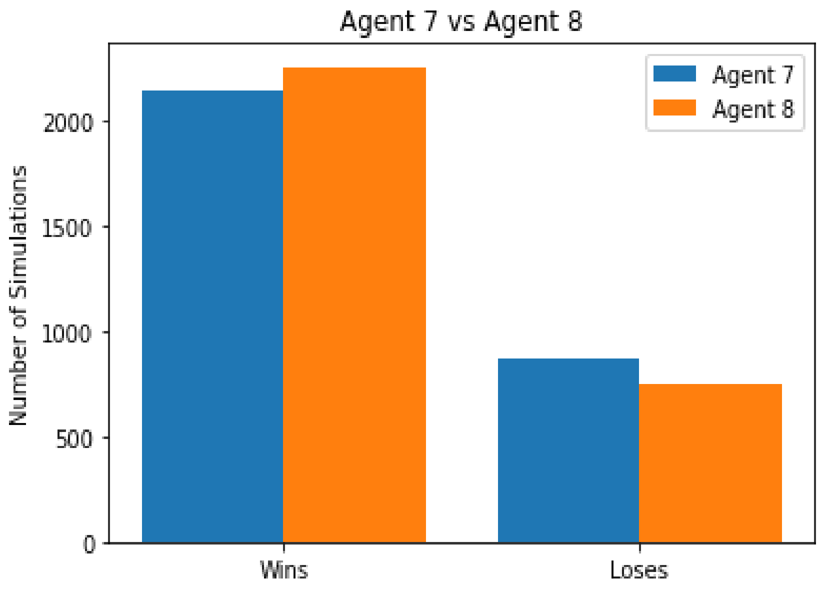


Figure 6: Agent 7 vs Agent 8

As seen from the figure 6, Agent 8 performs better in comparison to Agent 7 since Agent 8 chooses the node to survey from the belief state based on whose probability has increased the most in the last time stamp in addition to moving away from the predator if predator is one step away from the agent.

6 The Faulty Survey Drone

When an agent surveys a node to check if it is being occupied by a prey or a predator, we rely completely on the surveillance that it is giving us the correct information.
But what if it wasn't?
In this scenario, we are told that everytime the survey drone spots that a node is occupied, there is a 90% chance that it reports it correctly and a 10% chance that it falsely reports that it isn't occupied.

6.1 Re-implementation of Agent 7 without updating belief states

Analysis

(a) How often the Predator catches the Agent?
Answer- 1105 times out of 3000 times (36.83%) the predator catches the agent i.e., the number of times the agent loses the game

(b) How often the Agent catches the Prey?
Answer- 1895 times out of 3000 times (63.17%) the agent catches the prey i.e., the number of times the agent wins the game

(c) How often the simulation hangs (no one has caught anything past a certain large time threshold)?
Answer- The simulation hangs 0 times. (Threshold=5000 as mentioned by TA)

(d) How often the Agent knows exactly where the Prey is during a simulation where that information is partial?
Answer- 1.54% of times the agent knows exactly where the Prey is during a simulation.

(e) How often the Agent knows exactly where the Predator is during a simulation where that information is partial?
Answer- 9.01% of times the agent knows exactly where the Predator is during a simulation.

6.2 Re-implementation of Agent 8 without updating belief states

Analysis

(a) How often the Predator catches the Agent?
Answer- 1099 times out of 3000 times (36.63%) the predator catches the agent i.e., the number of times the agent loses the game

(b) How often the Agent catches the Prey?
Answer- 1901 times out of 3000 times (63.37%) the agent catches the prey i.e., the number of times the agent wins the game

(c) How often the simulation hangs (no one has caught anything past a certain large time threshold)?
Answer- The simulation hangs 0 times. (Threshold=5000 as mentioned by TA)

(d) How often the Agent knows exactly where the Prey is during a simulation where that information is partial?
Answer- 1.55% of times the agent knows exactly where the Prey is during a simulation.

(e) How often the Agent knows exactly where the Predator is during a simulation where that information is partial?
Answer- 9.58% of times the agent knows exactly where the Predator is during a simulation.

6.3 Re-implementation of Agent 7 updating the belief states

We first run agent 7 by implementing the faulty drone but we don't account for it in our belief states. Then we re-implement agent 7 with updates to the belief states, accounting for the faulty drone situation.

We had to change the way the belief systems are calculated. And this is how we tackled it:

(a) Everytime we survey a node and it comes up False, we can not consider the probability for that node as 0 anymore. We need to consider it 0.1, since it could be a false negative with 10% chance.

(b) But everytime, the survey drone does come up as True, we can continue considering the probability of that node as 1, because the survey node only gives out false negatives with a certain probability, but never false positives.

Apart from this, the rest of the logic works just like the older version of agent 7.
Analysis

(a) How often the Predator catches the Agent?
Answer- 1020 times out of 3000 times (34%) the predator catches the agent i.e., the number of times the agent loses the game

(b) How often the Agent catches the Prey?
Answer- 1989 times out of 3000 times (66%) the agent catches the prey i.e., the number of times the agent wins the game

(c) How often the simulation hangs (no one has caught anything past a certain large time threshold)?
Answer- The simulation hangs 0 times. (Threshold=5000 as mentioned by TA)

(d) How often the Agent knows exactly where the Prey is during a simulation where that information is partial?
Answer- 1.68% of times the agent knows exactly where the Prey is during a simulation.

(e) How often the Agent knows exactly where the Predator is during a simulation where that information is partial?
Answer- 10.891% of times the agent knows exactly where the Predator is during a simulation.

6.4 Re-implementation of Agent 8 updating the belief states

Similarly we run agent 8 with the faulty drone scenario, once by updating the belief state accordingly and once without.
Analysis

(a) How often the Predator catches the Agent?
Answer- 999 times out of 3000 times (33.30%) the predator catches the agent i.e., the number of times the agent loses the game

(b) How often the Agent catches the Prey?
Answer- 2001 times out of 3000 times (66.70%) the agent catches the prey i.e., the number of times the agent wins the game

(c) How often the simulation hangs (no one has caught anything past a certain large time threshold)?
Answer- The simulation hangs 0 times. (Threshold=5000 as mentioned by TA)

(d) How often the Agent knows exactly where the Prey is during a simulation where that information is partial?
Answer- 1.68% of times the agent knows exactly where the Prey is during a simulation.

(e) How often the Agent knows exactly where the Predator is during a simulation where that information is partial?
Answer- 10.891% of times the agent knows exactly where the Predator is during a simulation.

The comparisons for these versions is given below:

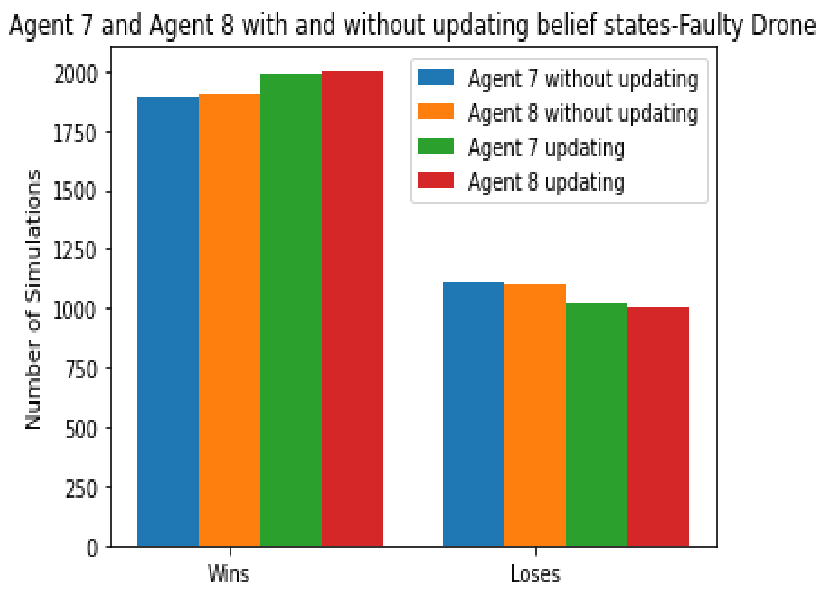


Figure 7: Agent 7 and Agent 8 with and without updating belief states-Faulty Drone

Agent 7 and Agent 8 without accounting for the changes in the belief state perform poorly in comparison to the Agent 7 and Agent 8 which account for the changes in the belief state.

6.5 Agent 9

We had to design an agent 9 that surpassed the performance of agent 7 and 8. We thought of changing the way we survey.

In agent 7 and 8, we were surveying to find the predator, not the prey, thinking that if our agent survives long enough, he will eventually spot the prey.

For agent 9 we tried to find both the predator and the prey. We go for the node with the highest probability across the prey and predator belief states. This helped our agent spot the prey and move towards it, while also greatly escaping from the predator.

6.6 Analysis

(a) How often the Predator catches the Agent?
Answer- 970 times out of 3000 times (32.33%) the predator catches the agent i.e., the number of times the agent loses the game

(b) How often the Agent catches the Prey?
Answer- 2030 times out of 3000 times (67.67%) the agent catches the prey i.e., the number of times the agent wins the game

(c) How often the simulation hangs (no one has caught anything past a certain large time threshold)?
Answer- The simulation hangs 0 times. (Threshold=5000 as mentioned by TA)

(d) How often the Agent knows exactly where the Prey is during a simulation where that information is partial?
Answer- 1.6798% of times the agent knows exactly where the Prey is during a simulation.

(e) How often the Agent knows exactly where the Predator is during a simulation where that information is partial?
Answer- 10.91% of times the agent knows exactly where the Predator is during a simulation.

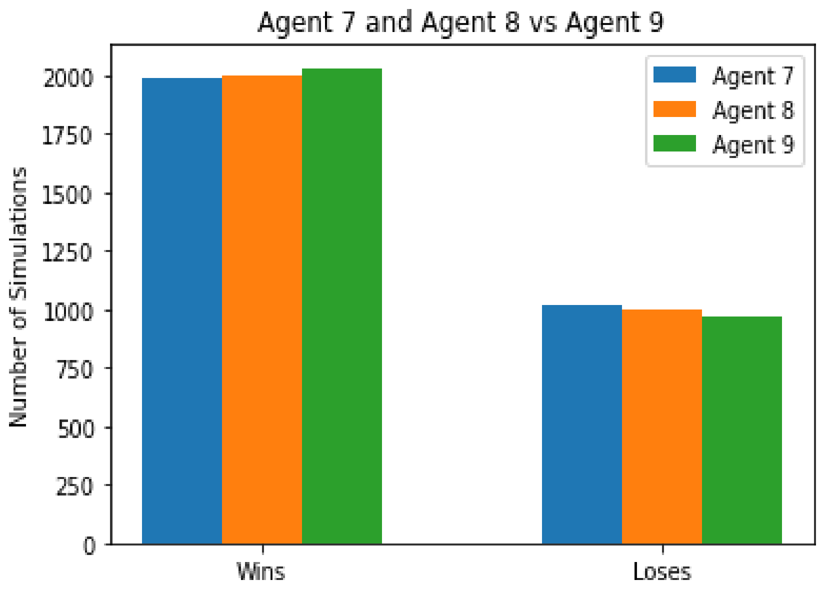


Figure 8: Agent 7 and Agent 8 vs Agent 9

The figure 8 shows that Agent 9 performs better compared to Agent 7 and Agent 8 since it account for the survey node from the highest probability amongst the belief states of prey and predator both.

7 Bonus

In the Partial Information Settings, we were able to survey a node and move to a meticulously chosen location on the basis of the probabilities updated with the survey results.

But in the bonus scenario, we are given a constraint that our agents can only either survey a node or move, not both.

7.1 Implementation of Agent 10

With the above constraint in mind, we designed an agent 10.
We made agent 10 keeping the combined partial information setting in mind, where we are unaware about the whereabouts of both the prey and the predator.

Logic Behind implementation:
Since we know where the predator is at the beginning of the game, we can calculate the distance from the predator to the agent in the beginning.

This distance, say its k steps, tell us that if the agent doesn't move at all, the predator will take atleast k timesteps to get to the agent to kill it. Since it is the distracted predator, it can take more than k timesteps but never less than that.

So we decided that it is safe for the agent to not move and just survey for the prey till the predator doesn't get too close. Upon some trials we decided that the best results were being fetched when we kept the threshold at k-5 timesteps. (So if the predator comes within 5 steps on the agent, its time for it to move to escape it)

For those k-5 timesteps, the agent can keep looking for the prey.
While surveying, we can be met with two scenarios:
Scenario 1:
We don't find the prey in any of those survey attempts. Then we just eventually start moving away from the predator when it gets close.
Scenario 2:
If we do find the prey, then we start moving towards the prey.
We can't only move because the predator can catch up to the agent, since the graph is interconnected within the circular structure.
Hence when we start moving towards the prey, we alternatively survey for the predator and prey as well.
So we follow the actions like: move-survey-move-survey-move

The belief states for both the prey and predator will be updated at every timestep. When we move or when we survey, either action will give us new information, which can be used to modify the belief states.

7.2 Analysis

(a) How often the Predator catches the Agent?
Answer- 1221 times out of 3000 times (32.33%) the predator catches the agent i.e., the number of times the agent loses the game

(b) How often the Agent catches the Prey?
Answer- 1779 times out of 3000 times (67.67%) the agent catches the prey i.e., the number of times the agent wins the game

(c) How often the simulation hangs (no one has caught anything past a certain large time threshold)?
Answer- The simulation hangs 0 times. (Threshold=5000 as mentioned by TA)

(d) How often the Agent knows exactly where the Prey is during a simulation where that information is partial?
Answer- 6.521% of times the agent knows exactly where the Prey is during a simulation.

(e) How often the Agent knows exactly where the Predator is during a simulation where that information is partial?
Answer- 5.1096% of times the agent knows exactly where the Predator is during a simulation.

7.3 Comparison of Performance

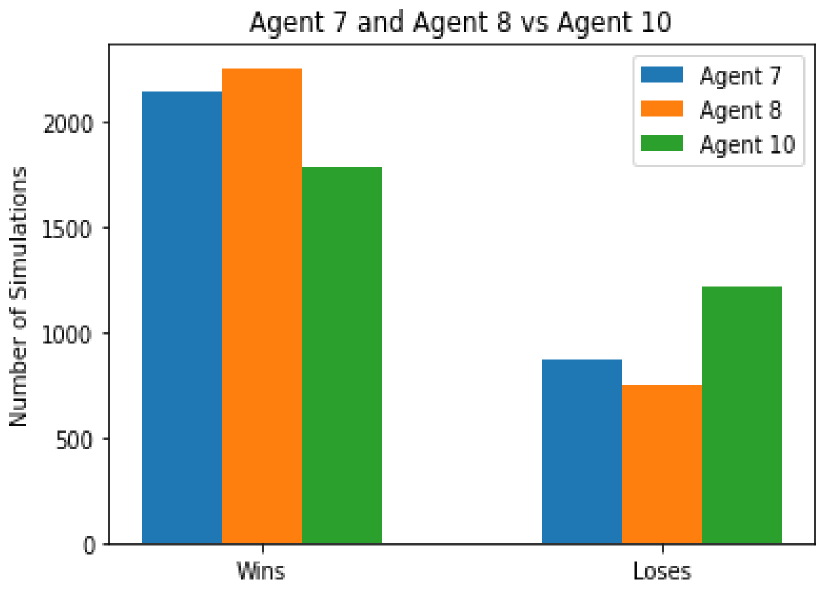


Figure 9: Agent 7 and Agent 8 vs Agent 10

As seen from figure 9, Agent 8 performs the best followed by Agent 7. Agent 10 performs the worst because agent 10 can only survey or move at each timestep while agent 7 and agent 8 survey and move at each timestep which increases the probability of finding the prey and the predator.