Circle of Life Lab Report

Tej Shah
Rutgers University
tej.shah@rutgers.edu

Srinandini Marpaka
Rutgers University
sm2237@rutgers.edu

Abstract

We develop Agents to interact in the circle-of-life game, where a Predator tries to catch the Agent (loss) before the Agent catches the Prey (win) in various environments. In the complete information and partial prey environments, the predator is optimal, always selecting actions that maximally decrease the distance to the Agent from its action space; in all other environments, the predator is distracted, choosing optimal actions with 0.6 probability and a random neighbor with 0.4 probability. The prey always selects any one of its neighbors or its current node randomly. We develop $A1, A2, \ldots, A10$ in various environments: complete, partial prey, partial predator, complete partial, and complete partial defective.

1 Introduction

Suppose that there are 3 entities in an environment: the prey, the predator, and the agent. These 3 entities interact with each other on a graph environment with $\mid V \mid = 50$ nodes. The game's objective is for the agent to capture the prey before being killed by the predator. Until the game concludes, the prey moves with equal probability to either its current location or any one of its immediate neighbors from the current node. For Agents 1 - 4, the predator always moves to its neighbor, which minimizes the distance to the agent, breaking ties randomly. For all other agents, the predator is easily distracted and has a 0.6 probability it will act optimally and a 0.4 probability that it will move to one of its neighbors uniformly at random. The agent moves first, then the prey, and then the predator. If the agent and prey occupy the same node, then the agent wins. If the agent and the predator occupy the same node, the predator wins. The game can terminate early after t time steps.

2 Agents

The game terminates when either of the following conditions is met: (1) the agent and the prey occupy the same node (agent wins); (2) the agent and the predator occupy the same node (predator wins); (3) the agent took $\geq t = 1000$ steps, resulting in an early termination hung simulation.

2.1 Complete Information Setting

In Complete Information, the agent knows exactly where the prey and predator are. In $A1$ and $A2$, both strategies are variants of minimizing distance to prey and maximizing distance to the predator.

2.1.1 Agent 1

We select a neighbor at random for the first non-empty set intersection in the following order of priority : (1) $A \cup B$, (2) $A \cup D$, (3) $C \cup B$, (4) $C \cup D$, (5) $B \cup B$, (6) $D \cup D$, (7) $E \cup E$.

a is the current location of the agent. `d(u,v)` returns the minimum distance from two locations using Breadth First Search (BFS). x `= d(a, prey.location)` and y `= d(a, pred.location)`.

$$A = \{n_i \mid d(n_i, prey) < x\} \quad = \text{set of neighbors that are closer to prey}$$
$$B = \{n_i \mid d(n_i, prey) > y\} \quad = \text{set of neighbors that are farther from predator}$$
$$C = \{n_i \mid d(n_i, prey) \leq x\} \quad = \text{set of neighbors that are not farther from prey}$$
$$D = \{n_i \mid d(n_i, prey) \geq y\} \quad = \text{set of neighbors that are not closer to predator}$$
$$E = \{a\} \quad = \text{set only including the agent's current location}$$

2.1.2 Agent 2

$A2$ greedily selects actions that bring it closer to the prey until the predator is too close by, at which point it maximally runs away until the game ends. While the Agent has the shortest distance of > 2 turns away from the Predator, use Breadth First Search (BFS) to compute the minimum distance to the Prey from each of the Agent's current neighbors' locations and greedily select the neighbor that minimizes the distance to the prey. When the Predator is a distance of ≤ 2 from the Agent, the Agent computes BFS on each of its neighbors to compute its' minimum distance from the Predator; then, the Agent greedily selects the neighbor that maximizes the minimum distance to the predator.

2.2 Partial Prey Information Setting

In this setting, we don't know where the prey is initially, and we do know where the predator is. At each time step, we can survey a node to see whether the prey exists at that location.

2.2.1 Agent 3

Agent 3 has a probabilistic model of the world. Initially, it believes that the chance that a prey exists at n_i is equally likely at all locations besides the agent's current location. At each time step, $A3$ selects the highest probability node, breaking ties randomly. If the prey does not exist at that location and the actual location of the prey has not been determined even once, then we update the probabilities of all nodes, excluding the agent's current location or current surveyed node, to be of equal likelihood (the agent's current location or surveyed node would have 0 probability). If the surveyed node contains the prey, then all nodes have a 0 probability of containing the prey except for the surveyed node, which has a probability of 1 of containing the prey. If the surveyed node does not contain the prey, but we have seen the prey before, then we distribute the probability of a node to any one of its neighbors depending on the frequency of visiting that node from the current frontier. For a more detailed and explicit discussion of belief updates, please refer to section **3.2**, where we specify the details.

After we update the probabilities of the belief of where the prey is, we predict the highest probability node to contain the prey. Then, we move $A3$ according to the logic of $A1$ based on the Agent's current location, the Predator's current location, and the predicted Prey's current location.

2.2.2 Agent 4

Agent 4 has the same probabilistic model of the world as Agent 3. The first optimization is as follows: when we select a node of the highest probability to be the predicted location of the prey, we select a node not only of highest belief but also of farthest distance to the predator and then break ties at random. This heuristic of selecting nodes as the potential prey to plan a path to that are farthest from the predator, while we are uncertain of where the prey is, maximizes the odds of survivability since it is more likely that the Agent will survive and not die by running into the predator. The second optimization is to move $A4$ according to the core logic of $A2$ based on the Agent's current location, the Predator's current location, and the predicted Prey's current location.

2.3 Partial Predator Information Setting

In this setting, the predator is easily distracted; with $p = 0.6$ it will move optimally, and $p = 0.4$, it will randomly select a neighbor. We are given where the predator is initially. We do know where the prey is at all times. At each time step, we can survey a node to see whether the predator exists.

2.3.1 Agent 5

Agent 5 has a probabilistic model of the world. Initially, it knows which node contains the predator and assigns a probability belief of 1 to that node and 0 everywhere else. From then on, until termination, it surveys the highest probability node, breaking ties randomly. If the surveyed node contains the highest probability, then it redistributes the probability mass of the current surveyed node to the most optimal (least distance to the agent) neighbors with a probability of 0.6 and to all neighbors of the node with a probability of 0.4. Otherwise, if the surveyed node does not contain the predator, then we redistribute the probability mass to all the neighbors of the current node states that the agent could possibly be in the frontier $\mathcal{F}$. For a more detailed and explicit discussion of belief updates, please refer to section **3.3**. Then, assuming that the Predator is now located at the highest probability node, we now move $A5$ according to the rules and logic specified by $A1$.

2.3.2 Agent 6

Agent 6 has a similar probabilistic model of the world as Agent 5. The main difference is that we assume that the predator will always move to a neighbor with the shortest distance remaining from the Agent. As indicated in **3.3**, the update is now as follows to accommodate for the optimal movement.

$$B(i) = \begin{cases} \frac{\texttt{opruned[i]}}{\sum_{s \in \texttt{opruned}} \texttt{opruned[s]}} & i \in \texttt{opruned} \\ 0 & i \notin \texttt{opruned} \end{cases}$$

To outperform Agent 5, we assume, given the probabilistic model of the world, the predator is located at a node with the highest belief. From there, we modify the core logic of $A2$ as follows, since we have more uncertainty about the true location of the predator: we select the action that moves us closest to the prey so long as the predicted predator distance is ≥ 3 away and the belief of the action has ≤ 0.1 probability of containing the predator. If this condition is not satisfied, we choose the neighbor that maximally moves away from the location of the Predicted Predator. In other words, we want to more greedily select outcomes that take us closer to the prey than before; since we have more uncertainty about the location of the predator, the heuristic to move away maximally from the optimal predator is still useful but not as useful of an inductive bias as it is in the case of a complete information setting as in $A2$. Essentially, we assume the worst-case location for the predator by assuming the predator moves optimally - this heuristic enables us to maximize survivability.

2.4 Combined Partial Information Setting

In this setting, we do not know where the predator and the prey are. We do know where the predator starts off at. Consider Agents 7 and 8, which act in this partially observable environment. At each time step, we can survey a node to see what occupies that node and if anything is there.

2.4.1 Agent 7

Agent 7 has a probabilistic model of the world, with separate beliefs about where the predator and where the prey are. If the agent is not currently certain about the location of the predator (in scenarios where multiple potential actions have the highest probability of containing the predator), we survey one of the nodes with the highest probability of containing the predator. Otherwise, we survey the node with the highest probability of containing the prey. Based on whether the predator or prey exists at the surveyed node, we update the beliefs for `predbeliefs` and `preybeliefs` according to the logic of $A5$ and $A3$ respectively, so we have 2 different belief states.

Then, we assume the `potentialpred` and `potentialprey` exist at the node of highest probability for their respective belief states, breaking ties randomly. Then, we move $A7$ according to $A1$ logic, assuming that `potentialpred` and `potentialprey` are the true locations of the predator and prey.

2.4.2 Agent 8

Agent 8 has a similar probabilistic world model as Agent 7. The `preybeliefs` are updated according to the logic of $A3$ while the `predbeliefs` are updated according to the logic of $A6$. The main difference between the belief updates between $A7$ and $A8$ is that we assume that for $A8$, the predator will move all the time optimally rather than with a probability of 0.6. Also, for $A8$, we do not

move the according to the logic of $A1$ based on `potentialpred` and `potentialprey`. Instead, we use the following variant of $A2$, $A4$, $A6$: since we are always more certain of the predator location (since it moves optimally and we know where it initializes), we always choose the action that maximizes the distance away in the agent's current action space that maximizes the distance to the location of the predicted predator; at any point, we have queried a node that does contain a prey, we move the agent to the node that minimizes the distance to the predicted prey based on the agent's current action space if the `potentialpredatordistance` ≥ 4, `predbeliefs[action]` ≤ 0.4, and `preybeliefs[action]` ≥ 0.2; otherwise, we continue with the logic of moving to the action that maximizes distance away from the predicted predator.

2.5 Bonus

2.6 Agent 9

The Kalman Filter is an algorithm that uses a series of measurements observed over time and produces estimates of unknown variables that tend to be more precise than those based on a single measurement alone. It is a recursive filter, meaning it obtains a more precise posterior belief by combining previous prior beliefs and current information. Each node's posterior belief is updated according to the Kalman Filter equation using a measurement matrix and an identity matrix. As the agent is unaware of the noise, it assigns an equal probability of error (0.1) to each node, which is used in the noise covariance matrix. Agent 9 then moves according to A8 logic. If there is high confidence in either location of the prey or predator, it will move away from the predator and towards the prey. This is done by computing the distance to each node that contains a prey or predator and moving to the node with the highest probability for prey and the smallest distance for the predator. We didn't implement $A9$.

2.6.1 Agent 10

Suppose now that the Agent can only choose to move or to survey at each time step in the complete partial information environment. To account for this, we move only if we have more than 5% confidence of a predator existing in any one of the nodes: in this situation, we move according to the logic of $A8$, assuming the Prey and Predator to be at the nodes of maximal probability; to update the belief states of the Prey and Predator, we simply propagate the probability mass to the relevant neighbor of neighbors as defined earlier for the belief updates. Otherwise, if we have less than 5% confidence of a predator existing at any particular node, then we survey a node and update the beliefs according to $A8$. Surprisingly, this combination yields us *better* performance than $A8$; we hypothesize this to be due to the fact that we more often than not just distribute the probability mass, which allows our agents to be more "bold" and "risky" in choosing actions. We implemented $A10$.

3 Lab Report Questions

3.1 With the given specifications, you can add at most 25 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 initialize the graph with 50 nodes. At each of the nodes, we have a max degree of 3. Hence, the max number of edges to add is $150 \cdot 3 = 150$. But, the initial circle has $50 \cdot 2 = 100$ edges.There are possible edges = $150 - 100 = 50$ edges. But, each edge needs two nodes, so $50/2 = 25$ max edges.

We're always able to add at least 20 edges. We empirically verify this by generating $100,000$ graphs and counting the number of edges added. In $100,000$ graphs, with the key being the number of edges added and the values being their frequency, we get `{24: 45819, 23: 42199, 22: 7412, 25: 4254, 21: 314, 20: 2}`. Though it's rare, we can always add at least 20 edges to the graph according to the assignment specifications.

3.2 In the partial prey information setting, how should you update your probabilities of where the Prey is? Be explicit and clear in showing how the beliefs are updated.

a is agent's node location, s is the surveyed node, and $\mid V \mid = 50$ is the number of nodes.

(1) Initial belief is that all nodes are equally probable to contain the prey. The prey is certainly not in the agent's current location, so there is equal likelihood among $\mid V \mid -1 = 49$ nodes.

$$B(i) = \begin{cases} 0 & i \in \{a\} \\ \frac{1}{49} & i \in \{1, 2, \ldots, 50\} \text{ and } i \notin \{a\} \end{cases}$$

(2) If we survey a node and the prey is not there, then we know that the surveyed node and the agent's location definitely do not contain the prey, so there is equal likelihood among $\mid V \mid -2 = 48$ nodes. We only run this condition if we've never seen the prey before. Otherwise, we move to (3) or (4).

$$B(i) = \begin{cases} 0 & i \in \{a, s\} \\ \frac{1}{48} & i \in \{1, 2, \ldots, 50\} \text{ and } i \notin \{a, s\} \end{cases}$$

(3) If we survey a node, and a prey is there, we know that the prey is located at that node and nowhere else. There is a 100% chance that prey is at s and a 0% chance everywhere else. We also initialize the frontier $\mathcal{F} = \{s\}$ to contain the surveyed node to distribute probability for future turns.

$$B(i) = \begin{cases} 1 & i \in \{s\} \\ 0 & i \in \{1, 2, \ldots, 50\} \text{ and } i \notin \{s\} \end{cases}$$

(4) For $i \in \mathcal{F}$, we create a hashmap `count{node: frequency}` counting the frequency each node can be visited from the given frontier. `count[node]` is updated with $+1$ anytime a node already exists in the frontier or it is one of the graph neighbors of an existing node in the frontier. After we have updated the `count`, we globally set $\mathcal{F} =$ `count.keys()`. We set `p = deepcopy(count)` and set `p[a] = p[s] = 0`. Now, we have `p`, which stores the frequency of all potential prey locations.

$$B(i) = \begin{cases} \frac{\texttt{p[i]}}{\sum_{s \in \texttt{p}} \texttt{p[s]}} & i \in \texttt{p} \\ 0 & i \notin \texttt{p} \end{cases}$$

3.3 In the partial predator information setting, how do the probability updates for the Predator differ from the probability updates for the Prey? Be explicit and clear.

The belief updates differ since, for this environment, the Agent starts by knowing the location of the easily-distracted predator instead of not knowing the information about the unknown target (i.e., the prey in $A3, A4$) from the start. At every turn, the predator moves optimally 60% of the time and randomly 40% of the time. Hence, we distribute the probability mass to both optimal and random actions. Suppose the predator starts off at location k. We update beliefs as follows:

(1) In this setting, we start off by knowing the location of the predator, so all nodes not containing the predator are initialized to 0 probability. The node of the predator has 1 probability.

$$B(i) = \begin{cases} 1 & i \in \{k\} \\ 0 & i \in \{1, 2, \ldots, 50\} \text{ and } i \notin \{k\} \end{cases}$$

(2) After initializing the initial probability distribution, until termination, we survey the highest probability node of containing the predator. If the predator exists at that location, we initialize or re-write an existing frontier as $\mathcal{F} = \{s\}$. If the predator doesn't exist at surveyed location, we keep our existing frontier from the previous time step. For $i \in \mathcal{F}$, we create a hashmap `count{node: frequency}` counting the frequency at each node that can be visited from the given frontier. `count[node]` is updated with $+1$ anytime a node already exists in the frontier, or it is one of the graph neighbors of an existing node in the frontier. For $i \in \mathcal{F}$, we create a hashmap `distances{node: distance}`, which stores the minimum distance from each node in the frontier and its neighbors to the agent.

We then create a hashmap `opruned` that stores `opruned{node: frequency}` for any possible optimal action of min distance for each of the current states in the hashmap. We additionally create a hashmap `rpruned{node: frequency}` that counts the frequency at each node that can be visited from the given frontier through its neighbors randomly. For the final belief, we multiply `opruned` values by 0.6 and `rpruned` values by 0.4 to account for the predator's respective optimal and random movements. In the end, we set $\mathcal{F} =$ `opruned.keys()` $\cup$ `rpruned.keys()`.

$$B(i) = \begin{cases} \frac{\texttt{opruned[i] * 0.6 + rpruned[i] * 0.4}}{\sum_{s \in \texttt{opruned}} \texttt{opruned[s]} + \sum_{s \in \texttt{rpruned}} \texttt{rpruned[s]}} & i \in \texttt{opruned} \cup i \in \texttt{rpruned} \\ 0 & i \notin \texttt{opruned} \cap i \notin \texttt{rpruned} \end{cases}$$

Explicitly, we will highlight the differences in belief updates between agents in the Partial Prey Information setting and Partial Predator setting. (1) In the Partial Predator Information setting, we already know where the predator starts off, whereas, in the Partial Prey Information setting, we don't; hence, we do not need to survey nodes until we find a surveyed node that matches a predator repeatedly. From each iteration from the beginning, in the case we do not survey a node that actually has the predator, we can redistribute the existing probability masses to the most optimal neighbors with a 0.6 probability and to all neighbors with a 0.4 probability for each item in the frontier. (2) In the partial prey-information setting, the probability mass is distributed to all of the neighbors in the frontier based on the frequency each neighbor can be visited from the given frontier. In the partial predator-information setting, the probability mass is also distributed to all of the neighbors for each item in the frontier but gives greater weight to an optimal neighbor than a random, sub-optimal one.

3.4 When do your Agents outperform the specified Agents in the above, and why? Do you think that your Agents use the available info as effectively as possible? Why or why not?

Our Agents $A2, A4, A6, A8$ all outperform the specified agents $A1, A3, A5, A7$, respectively. In general, our agents greedily search for the prey and maximize survivability by always being a distance k away from the predicted predator; if an action is of high likelihood to be a prey and low likelihood to be a predator, then we select that action based on hand-crafted heuristics using our belief states for the prey and the predator. Given that we use the beliefs to guide the even-numbered agents' strategies, we believe that we use the available info well. If we had more time, we would look into a deep reinforcement learning approach to learn optimal policies from data observations.

3.5 For the Combined Partial Information Setting, imagine that your survey drone is defective and that if something is actually occupying a node being surveyed, there is a 0.1 probability that it gets reported as unoccupied (a false negative).

3.5.1 How do $A7$ and $A8$ compare if they don't take into account the defective signal

Per Table 5, $A7$ and $A8$ have a survival rate of 29.77% and 41.73% respectively.

3.5.2 How should you update your belief update rules to account for this?

Since the signals are defective, we update the beliefs at each time step as follows:

$$P(\text{obj@ } n_i \mid \text{sig=}F) = \frac{P(\text{sig=}F \mid \text{obj@ } n_i) \cdot P(\text{obj@ } n_i)}{P(\text{sig=}F \mid \text{obj@ } n_i) \cdot P(\text{obj@ } n_i) + P(\text{sig=}F \mid \text{obj not@ } n_i) \cdot P(\text{obj not@ } n_i)}$$
$$= \frac{0.1 \cdot P(\text{obj@ } n_i)}{0.1 \cdot P(\text{obj@ } n_i) + 1 \cdot (1 - P(\text{obj@ } n_i))}$$

3.5.3 How do $A7$ and $A8$ compare in this setting, once beliefs are updated?

Per Table 6, $A7$ now has a 29.87% average success rate, and $A8$ has a 43.37% success rate, improvements of 0.10% and 1.64% respectively from $A7$ and $A8$ without taking into account the signal being defective for a false negative with $p = 0.1$. Updating beliefs with Bayesian Updating enabled us to get better performance for $A7$ and $A8$ as a result. Refer to the experiments section for more details.

3.5.4 Can you build an Agent 9 to do better?

Yes. The measurements from the surveyed node are noisy, and we have prior belief states about where the predator and prey are. To converge to a good estimator of the true predator and prey locations, use a Kalman Filter to localize the predator and prey. More details are under $A9$ in the Agents section.

3.6 Bonus: In the partial info 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 (move take/node survey)? Implement in combined partial info setting.

The agent moves if at least one node has a greater than a 5% chance of containing the predator: in this situation, it assumes the predator and prey to be at the node of highest probability, breaking ties at random; the probability mass of predator and prey are distributed to the relevant neighbors within the

frontier, as outlined in the elaboration of belief update in **3.3** and **3.4**. Otherwise, the agent decides to survey nodes and update its belief states about the predator and prey per $A8$ until the condition that at least one node has a greater than 5% chance of containing the predator is met. Please refer to the discussion about $A10$ in the Agents section for more details on the implementation.

4 Experiments

4.1 Overview

We analyzed the agents' performance by averaging the results of 30 100-simulation trials. Figures 1 - 7 has the agents' survival rates plotted against the agents of the respective information setting. We have also plotted the error bars with a 95% confidence interval in these figures. For each setting, we consider the agent with the highest average success rate to have the best performance.

Tables 1 - 7 summarize the overall statistics of the agents of the respective information setting. The Average Success Rate represents the percentage of simulations in which the agent and prey occupied the same node, causing the agent to win. The losses are split into when the agent and predator occupied the same node (Average Losses) and when the agent took $\geq t = 1000$ steps (Average Timeouts). In simulations where the information is partial, Average-Found Prey and Average Found-Predator indicate the average number of times the agent knows exactly where the prey and predator are, respectively, as a percentage of the total moves of each simulation.

4.2 Complete Information Setting

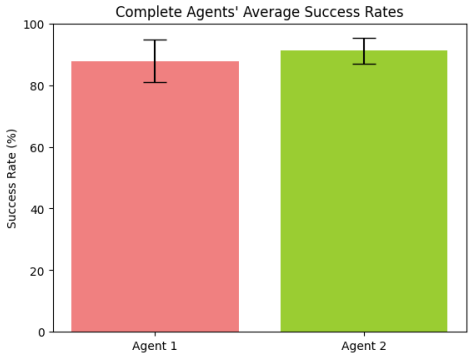

Figure 1: Success Rates

Table 1: Simulation Statistics

	Agent 1	Agent 2
Average Success Rate	87.77 %	91.20 %
Average Losses	12.23 %	7.83 %
Average Timeouts	0.00 %	0.97 %
Average Found-Prey	N/A	N/A
Average Found-Predator	N/A	N/A

Agent 1 and Agent 2 are in the complete information setting and have an average success rate of 87.77% and 91.20%, respectively. Agent 2's better performance can be attributed to its use of a modified greedy algorithm that maximizes survivability rather than Agent 1's predefined approach. Agent 2's 0.97% timeout exceeds Agent 1's 0.00% timeout because it takes more steps as it primarily uses the prey to direct its movements and only considers the predator if it is an immediate threat, while Agent 1 considers both the predator and prey almost equally from the beginning. However, even when considering the timeout as a loss for Agent 2, Agent 2 outperforms Agent 1.

4.3 Partial Prey Information Setting

Agent 3 and Agent 4 are in the partial prey information setting and have an average success rate of 61.13% and 89.47%, respectively. Agent 4's better performance and 0.93% greater timeouts can be attributed to the fact that it is bootstrapped on Agent 2 rather than Agent 1. The prey was found at 3.99% and 3.80% of total steps for Agent 3 and Agent 4, respectively. The decrease in value is because Agent 4 selects a random node of the highest belief and farthest distance from the predator to be the predicted location of the prey, while Agent 3 selects a random node of only the highest belief. The additional filter in Agent 4, when determining the prey's predicted location, is not representative of the prey's actual movement and, therefore, makes it more difficult to find the prey. However, the constraint helps maximize survivability, resulting in Agent 4 significantly outperforming Agent 3.

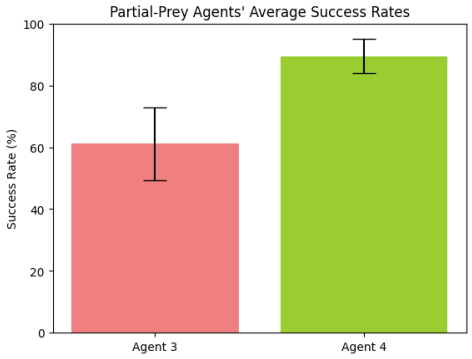

Figure 2: Success Rates

Table 2: Simulation Statistics

	Agent 3	Agent 4
Average Success Rate	61.13 %	89.47 %
Average Losses	38.87 %	9.60 %
Average Timeouts	0.00 %	0.93 %
Average Found-Prey	3.99 %	3.80 %
Average Found-Predator	N/A	N/A

4.4 Partial Predator Information Setting

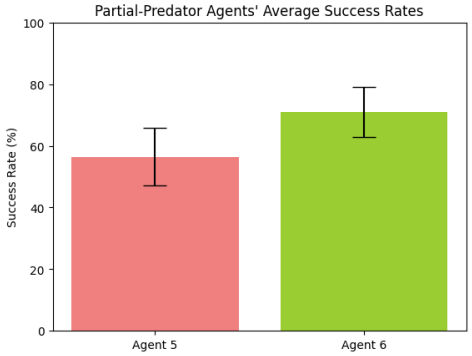

Figure 3: Success Rates

Table 3: Simulation Statistics

	Agent 5	Agent 6
Average Success Rate	56.47 %	71.00 %
Average Losses	43.53 %	29.00 %
Average Timeouts	0.00 %	0.00 %
Average Found-Prey	N/A	N/A
Average Found-Predator	49.18 %	53.81 %

Agent 5 and Agent 6 are in the partial predator information setting and have an average success rate of 56.47% and 71.00%, respectively. Agent 6's better performance can be attributed to the fact that it assumes the optimal movement of the predator rather than following Agent 5's beliefs which reflect the predator's actual 60.00% optimal and 40.00% random movement. The predator was found at 49.18% and 53.81% of total steps for Agent 5 and Agent 6, respectively. This is not the expected behavior as Agent 5 has a more accurate belief system; however, a possible explanation of Agent 5's lower value is that even if the predator moves at random in the immediate, it converges to the optimal action and that convergence results in the need of extra steps before the game concludes. Agent 6, in contrast, assumes that the predator always moves in the direction with the shortest distance remaining to the agent and, therefore, takes fewer steps and significantly outperforms Agent 5.

4.5 Combined Partial Information Setting

Agent 7 and Agent 8 are in the combined partial information setting and have an average success rate of 30.73% and 45.33%, respectively. Agent 8's better performance can be attributed to the fact that, similar to Agent 6, it assumes the optimal movement of the predator rather than following the predator's actual optimal movement 60.00% of the time and random movement 40.00% of the time as Agent 7 and Agent 5 do. The prey was found at 2.30% and 3.49% of total steps for Agent 7 and Agent 8, respectively. The reason for the increase in Found-Prey can be explained by the fact that rather than following Agent 1's predefined method as Agent 7 does, Agent 8 uses a greedy algorithm that maximizes the distance from the predator until the prey is found. Agent 8, in turn, is able to survive longer and, therefore, has a greater likelihood of finding the prey once it finds it once. The predator was found at 22.72% and 3.96% of total steps for Agent 7 and Agent 8, respectively. The decrease in Found-Predator can be explained by the fact that Agent 7's predator beliefs reflect the predator's actual movement, while Agent 8's predator beliefs reflect the predator's optimal movement.

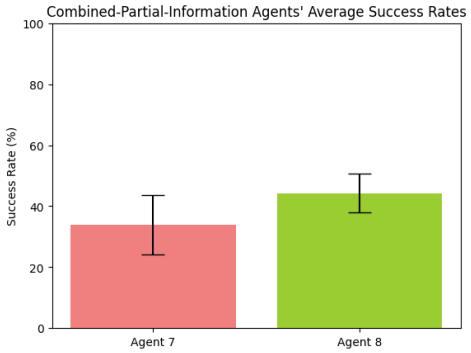

Figure 4: Success Rates

Table 4: Simulation Statistics

	Agent 7	Agent 8
Average Success Rate	30.73 %	45.33 %
Average Losses	69.27 %	54.67 %
Average Timeouts	0.00 %	0.00 %
Average Found-Prey	2.30 %	3.49 %
Average Found-Predator	22.72 %	3.96 %

With this, Agent 7 has a greater likelihood of knowing exactly where the predator is, while Agent 8 is able to maximize survivability and, therefore, outperform Agent 7.

4.6 Combined Partial Information Setting, Defective Readings

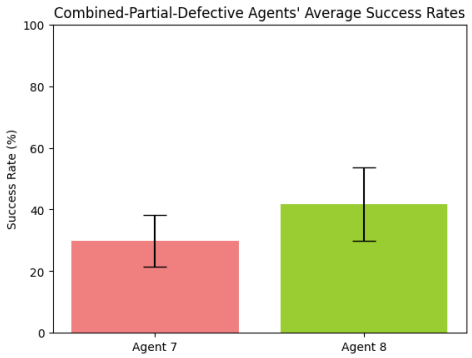

Figure 5: Success Rates

Table 5: Simulation Statistics

	Agent 7B	Agent 8B
Average Success Rate	29.77 %	41.73 %
Average Losses	70.23 %	58.27 %
Average Timeouts	0.00 %	0.00 %
Average Found-Prey	2.03 %	3.08 %
Average Found-Predator	17.74 %	4.12 %

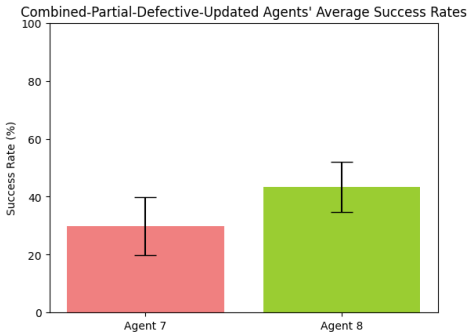

Figure 6: Success Rates

Table 6: Simulation Statistics

	Agent 7C	Agent 8C
Average Success Rate	29.87 %	43.37 %
Average Losses	70.13 %	56.63 %
Average Timeouts	0.00 %	0.00 %
Average Found-Prey	1.82 %	2.99 %
Average Found-Predator	17.14 %	4.30 %

Agents 7B, 8B, 7C, and 8C are in the combined partial information setting with defective readings. There is a 0.1 probability that a node that is occupied will be reported as unoccupied when surveyed.

Agents 7B and 8B are in the setting with the original probabilistic models and have an average success rate of 29.77% and 41.73%, respectively. These agents are bootstrapped off of the original Agent 7 and Agent 8. Therefore, these agents are similar to the original agents in that Agent 8B performs better, finds the prey more often, and finds the predator less often when compared to Agent

7B. However, Agents 7B and 8B do perform worse than Agents 7 and 8 as the new agents use the original agents' belief updates, and those do not reflect the surveyed nodes' potential false negatives.

Agents 7C and 8C are in the setting with the updated probabilistic models and have an average success rate of 29.87% and 43.37%, respectively. The agents are bootstrapped off of Agent 7B and Agent 8B. Therefore, these agents are similar to the other agents in a combined partial information setting in that Agent 8C performs better, finds the prey more often, and finds the predator less often when compared to Agent 7C. Agents 7C and 8C do perform better than Agents 7B and 8B as their beliefs have been updated to reflect the surveyed nodes' potential false negatives.

4.7 Bonus

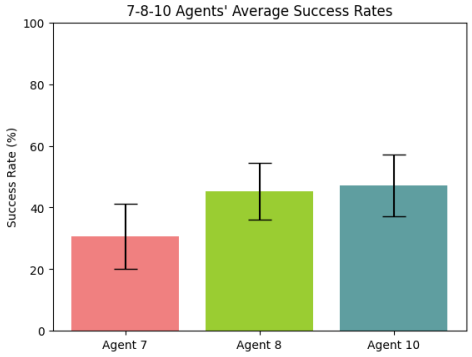

Figure 7: Success Rates

Table 7: Simulation Statistics

	A7	A8	A10
Avg Success Rate	30.73 %	45.33 %	47.20 %
Avg Losses	69.27 %	54.67 %	52.80 %
Avg Timeouts	0.00 %	0.00 %	0.00 %
Avg Found-Prey	2.30 %	3.49 %	0.00 %
Avg Found-Pred	22.72 %	3.96 %	0.00 %

Agent 10 is in the combined partial information setting and has an average success rate of 47.20%. Agent 10's better performance than Agent 7 and Agent 8 can be attributed to its more greedy approach that only surveys a node if it has less than 5% confidence of the predator's location. Agent 8's 0.00% Found-Prey and 0.00% Found-Predator can be explained by the fact that we only survey the node if we have minimal confidence on where the predator is located; therefore, in the cases, we are more than 5% certain, we just moving according to our assumption rather than surveying for confirmation.

5 Conclusion

We develop a variety of agents to interact in a variety of environments: the complete information environment, the partial prey information environment, the partial predator information environment, the complete partial information environment, and the defective signal complete partial information environment. $A1$ uses a methodical procedure. $A3, A5$, and $A7$ move according to $A1$ based on the belief of where the prey and/or predator are. $A2, A4, A6$, and $A8$ were developed by us to maximize survivability while simultaneously being greedy toward finding the prey. In the case of the defective signal, we use Bayesian Updating to update our beliefs in the false negative case. Our developed agents outperform all the specified agents $A1, A3, A5, A7$. $A9$ is outlined to use a Kalman Filter and $A10$ has to tradeoff between moving and surveying a node. 'In short, we were able to use our probabilistic world model of where the prey and predator are to guide our Agents' policy.