CS 520 - Project 2

Circle of Life

Produced by: Pranathi Vaddela

Net ID: PV250

Contents

1 Introduction **3**

2 Implementation **3**
- 2.1 Environment . . . 3
- 2.2 Prey . . . 5
- 2.3 Agent . . . 5
- 2.4 Predator . . . 5
  - 2.4.1 Shortest Path . . . 5
- 2.5 Game . . . 5
- 2.6 Assumptions . . . 7

3 Complete Information Setting **7**
- 3.1 Agent 1 . . . 8
  - 3.1.1 Agent 1 results . . . 8
  - 3.1.2 Drawbacks of Agent1 . . . 8
- 3.2 Agent 2 . . . 8
  - 3.2.1 Belief Update Mechanism the Prey . . . 9
  - 3.2.2 Results of Agent 2 . . . 9
- 3.3 Agent 1 vs Agent 2 . . . 10

4 Partial Prey Information Setting **10**
- 4.1 Agent 3 . . . 10
  - 4.1.1 Belief update mechanism at T_i before prey moves . . . 11
  - 4.1.2 TRANSITION MODEL - Belief update mechanism at T_i after prey moves . . . 11
  - 4.1.3 Results of Agent3 . . . 12
  - 4.1.4 Drawbacks of Agent3 . . . 12
- 4.2 Agent 4 . . . 12
  - 4.2.1 Results of Agent 4 . . . 13
- 4.3 Agent 3 vs Agent 4 . . . 13

5 Partial Predator Information Setting **13**
- 5.1 Agent5 . . . 14
  - 5.1.1 Belief update mechanism at T_i before predator moves . . . 15
  - 5.1.2 TRANSITION MODEL - Belief update mechanism at T_i after predator moves . . . 15
  - 5.1.3 Results of Agent5 . . . 16
  - 5.1.4 Drawbacks of Agent5 . . . 16
- 5.2 Agent 6 . . . 16
  - 5.2.1 Results of Agent 6 . . . 16
- 5.3 Agent 5 vs Agent 6 . . . 17

6 Combined Partial Information Setting **17**
- 6.1 Agent 7 . . . 17
  - 6.1.1 Results of Agent7 . . . 18
  - 6.1.2 Drawbacks of Agent7 . . . 18
- 6.2 Agent 8 . . . 18
  - 6.2.1 Results of Agent8 . . . 18
- 6.3 Agent7 vs Agent 8 . . . 19

7 Analysis And Report **19**
7.1 Defective Survey Drone . . . 19
7.1.1 Agent 7: Without the defective drone belief updates . . . 19
7.1.2 Agent 8: Without the defective drone belief updates . . . 20
7.1.3 Agent 7 vs Agent 8 - Without the defective drone belief updates . . . 20
7.1.4 Agent 7: With the defective drone belief updates . . . 21
7.1.5 Agent 8: With the defective drone belief updates . . . 21
7.1.6 Agent 7 vs Agent 8 - With defective drone belief updates . . . 22
7.2 Agent 9 . . . 22

1 Introduction

In project "The Circle of Life", we have three participants in given **environment**.

1. Agent (Wants to catch Prey)
2. Predator (Wants to catch Agent, and knows the agent location)
3. Prey (Moves freely in the Environment)

These three entities—the Predator, the Prey, and the Agent occupy the environment and can travel from node to node along the edges. The Agent and the Predator desire to capture Prey and the Agent, respectively. The Agent succeeds if it shares a node with the Prey. The Agent loses if he shares a node with the Predator. The three players move in rounds, starting with the Agent, then the Prey, and then the Predator.

I am taking **connected graph** of 50 nodes as the environment in this project and will analyze the Agent's success rates in different contexts. We are considering the following contexts for experimentation

1. The Complete Information Setting (Agent always knows exactly where the Predator & Prey is)
2. The Partial Prey Information Setting (Agent is uncertain about Prey location)
3. The Partial Predator Information Setting (Agent is uncertain about Predator location)
4. The Combined Partial Information Setting (Agent is uncertain about Predator & Prey locations)

In each setting we are implementing two types: of agents having their own strategy to catch Prey and avoid being caught by Predator. I built probabilistic models for settings with uncertainty and use it to guide in decision-making.

1. Odd number of Agents : Agents that follow the strategies provided.
2. Even number of Agents : Agents which develop its own strategy to catch Prey and avoid being caught by Predator. I have built probabilistic models for settings with uncertainty which will guide in decision-making.

2 Implementation

2.1 Environment

I am using a connected graph of 50 nodes as my environment. The aim is to create a consistent environment across all the different contexts. So the rules for creating the environment for this project are as follows:

- Create a cycle
- Add random edges to make it more connected
  - Picking a random node with a degree less than 3
  - Add an edge between it and one node within 5 steps forward or backward along the primary loop. (So, node 10 might connect to node 7 or 15, but not node 16.)
  - Do this until no more edges can be added.

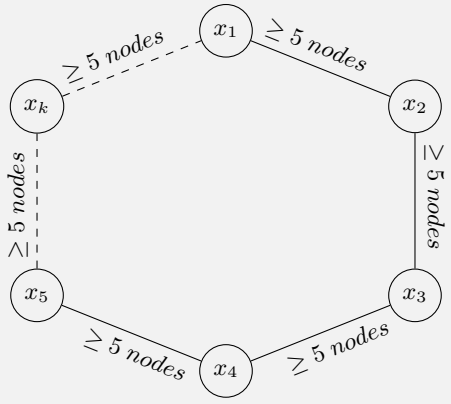
In this way, all nodes in the graph will have a degree of either 2 (because of cycle) or 3 (because of cycle & random edge). I am using **adjacency list** to represent the graph, which will give better performance and semantics in implementation. In the programming context, I have implemented a class for the graph with encapsulating properties and methods related to it.

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?

For 50 nodes, If we generate the graph in given manner, least number of edges that can be formed are 21 and maximum is 25.

Proof: Let's take a graph with N nodes with a cycle formed, so that each node will have degree 2. Now using the above mentioned rule, there will be e random edges added, making $2 * e$ nodes to have degree 3.

So the remaining nodes (having degree 2) are $k = N - 2 * e$, because no two nodes in this k nodes can form an edge. In this way there are minimum of 5 nodes in between any two nodes in these k nodes. Visually this can be represented as



According to the graph above, expression for total number of nodes can be given as

$$(5_1 + 5_2 + 5_3 + \cdots + 5_k)_{(in\_between\_nodes)} + k_{(degree\_2\_nodes)} \leq N$$

$$(5 * k) + k \leq N$$

$$6 * k \leq N$$

$$k \leq \lfloor N/6 \rfloor$$

Here we have N = 50,

$$k \leq 8$$

The maximum value k can have in this scenario is 8. So at most 8 nodes couldn't form random edge, implying minimum of 42 nodes are connected with random edges, which are 21 edges (minimum).

The minimum value k can have in this scenario is 0. So at least 0 nodes couldn't form random edge, implying maximum of 50 nodes are connected with random edges, which are 25 edges (maximum).

2.2 Prey

Prey is participant in 'Circle of life' which moves freely in it's neighborhood in the given environment. Prey either wishes to move to one of its neighbors or choose to remain at its current location in all the settings that we are going to work on. Prey is represented by object of Prey class with encapsulating properties (location) and responsibilities (initialisation and moving).

2.3 Agent

Agent is also a participant in 'Circle of life' trying to catch the Prey and avoiding being caught by Predator. In all the contexts Agent movements are designed in such a way that agent1, agent3, agent5 and agent7 prioritise their neighbors in similar fashion, while it differs for agent2, agent4, agent6 and agent8 which out performs odd numbered agents according to its respective context setting. Agent will be represented as an instances of Agent class encapsulating its properties and methods.

2.4 Predator

Predator will try to catch the agent by always trying to move closer to the agent. So the predator will compute the shortest path to the Agent at every step and will move in that direction.

In Partial Predator Information setting and Combined Partial Information setting, **Distracted Predator** is implemented, which differs with the normal predator - With chance 0.6, the predator will move close to the distance, but with probability 0.4, the predator travels to one of its neighbours uniformly at random. The predator is easily distracted and instead of always travelling close to the distance. Predator is also represented by instance of it's encapsulated class Predator.

2.4.1 Shortest Path

For every game, we have to compute the shortest path between nodes several times. So I have implemented Floyd Marshall algorithm to compute the all pairs shortest path. I have pre-computed the shortest distance between nodes and stored it in a *self.distance* variable in Graph class.

2.5 Game

Game is a class which will co-ordinate the actions among the participants (Agent, Prey, Predator) in the given environment. Co-ordination in the given project will reflect the following:

1. Returning agent location to predator.
2. Returning predator location to agent in first two context settings, in which agent is certain about predator location.
3. Returning prey location to agent in contexts in which agent is certain about prey location.
4. Return neighbors of participant's location when asked by participant (agent, prey or predator).
5. Provide the information regarding the environment to participants when requested, such as degree for a node, shortest path between two specific nodes etc.

These functionalities are achieved by means of storing references of environment and all the participants in Game class. When an object created for environment, agent, prey and predator it will get registered in the Game object, and also store this Game object to request for information required at any given time.
Constructor will look like this for environment and all the participants ...

```
# Constructor for agent
def __init__(self, game: Game):
    self.game = game
    self.game.register_agent(self)
    ...
    ...
```

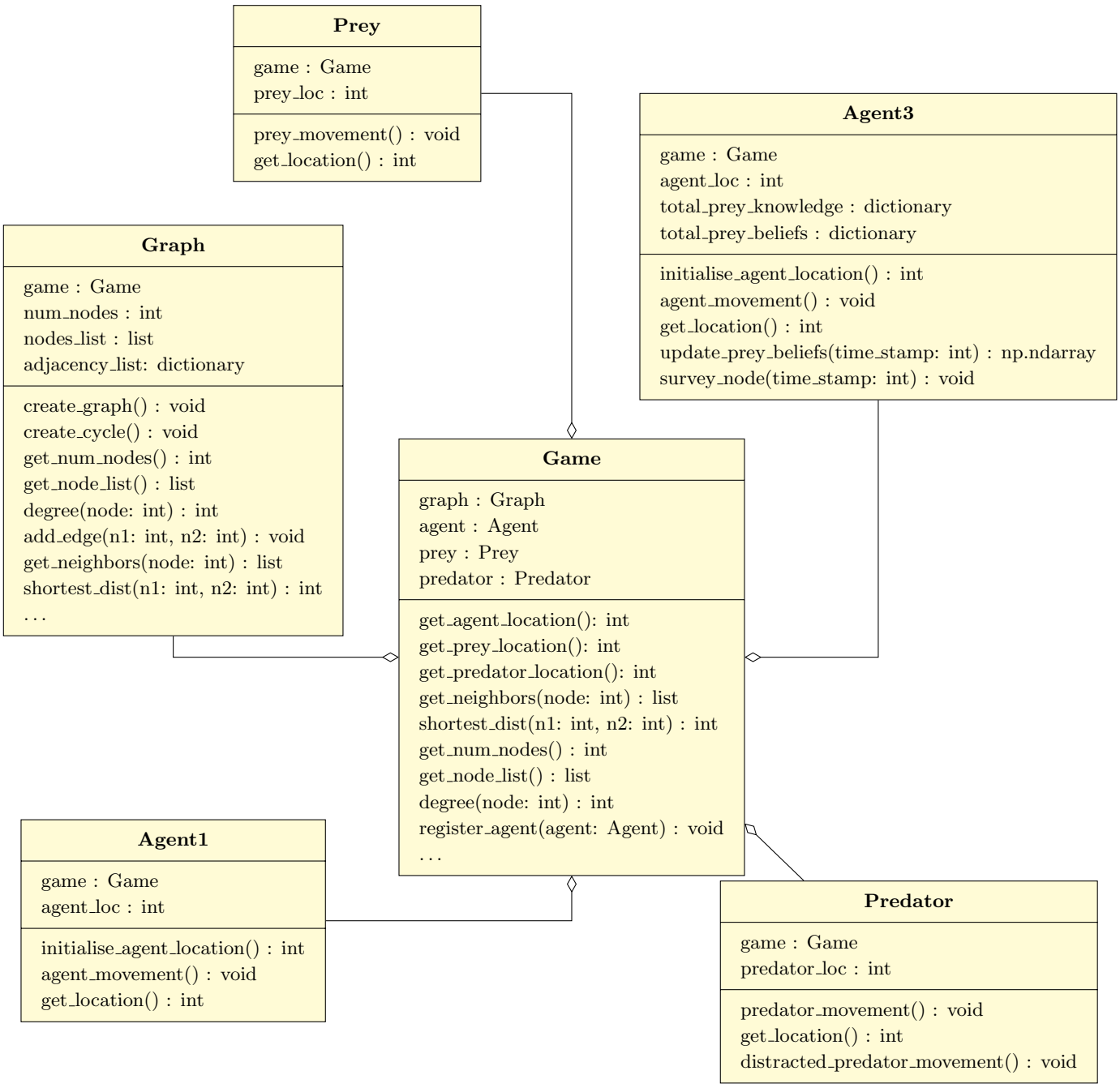
definition for game.register_agent() is given by

```
class Game:
    def register_agent(self, agent: Agent):
        self.agent = agent
```

In the similar fashion constructors for all the participants and environment created along with register methods in Game class.

Note:

So Before creating any object for agent, prey, predator and environment we should create Game object and pass the same Game object to all the participants and environment to get registered in game object.



2.6 Assumptions

- Prey and Predator can reside in the same node.
- Agent should initialise in node other than prey's and predator's initial location.

3 Complete Information Setting

In this setting, the Agent always knows exactly where the Predator is and where the Prey is. In this context I created two agents.

3.1 Agent 1

As Agent is aware of both prey and predator location, it is straight forward implementation. Only concern is how agent is going to catch prey and escape from predator to accomplish its role in 'The Circle of Life'. When this Agent has to move, it will look at all of its neighbours and choose one in the order listed below (breaking ties at random).

- 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.

If any rule is satisfied with neighbors of current location, one of the neighbor will be selected out of all satisfied neighbors of that rule. If no rule is satisfied by the neighbors of current location then agent will stay still.

3.1.1 Agent 1 results

	Agent	no_of_simulations	max_steps_allowed	Success rate	Failure rate	Hangs	% knowing prey location	% knowing predator location
0	Agent1	3000	50	87.53	12.40	2	100.0	100.0
1	Agent1	3000	100	87.87	12.13	0	100.0	100.0
2	Agent1	3000	150	89.00	11.00	0	100.0	100.0
3	Agent1	3000	200	88.30	11.70	0	100.0	100.0
4	Agent1	3000	250	88.67	11.33	0	100.0	100.0
5	Agent1	3000	300	86.87	13.13	0	100.0	100.0

Figure 1: Agent1 results

1. Agent 1 has a good success rate of 88%
2. Has no hangs

3.1.2 Drawbacks of Agent1

Agent 1 follows according to the rules fed to it. Agent 1 fails at few steps where it, either prioritises in catching prey rather that escaping from predator, or else in the situation where Agent is too close to predator and has no other option to move.

3.2 Agent 2

Agent 2 is a modified version of Agent 1 which uses a probabilistic model to determine the future prey locations. The Agent calculates future beliefs regarding the prey for future time step, to locate the prey and move towards it rather than the current prey location.

The main factor here is how many time steps from now do we want the Agent to look at. Considering just the next time step may not help sometimes, and calculating beliefs for a time step way in the

future would be quite tedious. So, a time step of **minimum(5, half of distance of prey from agent)**seemed as a good estimate on how far we can Agent can predict for the future prey locations.

3.2.1 Belief Update Mechanism the Prey

Probabilistic Model for Prey
Agent will follow the following to implement this strategy:

- The Agent will consider a future time step which is **minimum(5, half of the distance of prey from agent)**
- Agent will calculate the future beliefs of the prey for that time step.

  Suppose, the agent needs to calculate the future belief of prey at time step $\{t_4\}$ The agent has to calculate all the future beliefs at time steps: $\{t_1\}$, $\{t_2\}$, $\{t_3\}$ to get beliefs at $\{t_4\}$
- Agent will now move to the prey location having the highest belief (breaking ties according to the shortest distance from agent to prey).

  The Agent will check all the nodes that has the shortest distance from it to the prey (breaking ties at random)

To calculate the beliefs, the Agent here takes into account the possibility of prey either moving to its neighbors or stays in its location for the next time step. A probabilistic model can be developed where we can evaluate how the prey might move to the other nodes in the future time steps.

The Agent will consider the chances that the prey will choose one of its neighbors for the next move or stay at the place. So the probability of a node X having prey can be viewed as the sum of probabilities of prey coming from its neighbors. This logic can be represented as:

$$P(Prey\ at\ X\ at\ T_{i+1}) = \sum_{j=1}^{degree(x)} P(n_j\ at\ time\ T_i) * (1/(degree(n_j)+1)) + P(X|\ at\ time\ T_i) * (1/(degree(X)+1))$$

3.2.2 Results of Agent 2

	Agent	no_of_simulations	max_steps_allowed	Success rate	Failure rate	Hangs	% knowing prey location	% knowing predator location
0	Agent2	3000	50	96.20	3.07	21	100.0	100.0
1	Agent2	3000	100	96.73	3.20	2	100.0	100.0
2	Agent2	3000	150	97.10	2.87	0	100.0	100.0
3	Agent2	3000	200	96.17	3.83	0	100.0	100.0
4	Agent2	3000	250	96.67	3.33	0	100.0	100.0
5	Agent2	3000	300	97.63	2.37	0	100.0	100.0

Figure 2: Agent2 results

1. Agent2 has a high success rate of 96-97%, since the probabilistic model will allow the Agent to have future information about the prey and this helps the Agent to reach prey better.
2. Very few hangs

3.3 Agent 1 vs Agent 2

			Agent1	Agent2
	no_of_simulations	max_steps_allowed	Success rate	Success rate
0	3000	50	87.53	96.20
1	3000	100	87.87	96.73
2	3000	150	89.00	97.10
3	3000	200	88.30	96.17
4	3000	250	88.67	96.67
5	3000	300	86.87	97.63

Figure 3: Agent1 vs Agent2 results

4 Partial Prey Information Setting

In this scenario, the Agent is always aware of the location of the Predator but does not necessarily know the location of the Prey. Each time the Agent advances, it can first select a node to survey (anywhere in the graph) to see if the Prey is present. **Every time the Agent enters a node and the Prey isn't there, the Agent learns where the Prey isn't.** In this situation, the Agent must keep track of a belief state for the Prey's location, which is a set of probabilities for the Prey's presence at each node. **These probability need to be revised every time the Agent gains new information about the Prey. These probability must be updated whenever the Prey is known to move.**

In these cases how should you be updating the probabilities of where the Prey is? Be explicit and clear.

Explained in sections 4.1.1 and 4.1.2

4.1 Agent 3

Since Agent 3 doesn't know the exact location of the prey, it develops a belief system to figure out the exact location of prey eventually. Here, the Agent 3 is designed in such a way that it will built the belief system ***beliefs***, and update it according to the actions taken by the agent described below. Since Agent 3 follows same rules as the Agent 1, I have inherited the Agent1 class to the Agent 3 class to inherit the agent movement and methods.

I am considering below actions will occur at given time T_t sequentially for **Partial Prey Information Setting** :

1. Agent surveys a node according to rules at T_t
2. Agent updates $\boldsymbol{beliefs_t}$ using the result of survey at T_t
3. Agent moves to a new location (one of it's neighbors) at T_t
4. Agent updates $\boldsymbol{beliefs_t}$ using it's new location at T_t
5. Prey moves at T_t
6. Agent updates the beliefs according to the transition model at T_t
7. Predator moves at T_t

The belief updates are performed after every step when it gains new information about the prey location, and the beliefs are done as **before the prey moves at** T_i and **after prey moves at** T_i, which I will be discussing next.

4.1.1 Belief update mechanism at T_i before prey moves

According to the sequences of events occurring, the agent has to update the beliefs after surveying and the agent movement(i.e., Action 2 and 4). Once the agent gains knowledge about the node, on whether it is having prey or not, it will update the belief accordingly, which raises 2 cases:

Let $\boldsymbol{x}$ be the node that has been updated in the $knowledge_t$ (i.e., either it is the node that is being surveyed or the node to which the agent has moved), and $\boldsymbol{n}$ be the node that represents remaining nodes.

1. **If Prey is at the node $\boldsymbol{x}$** : If the node has Prey, then the Agent will make the belief of that node as 1, since it is having highest belief of having the prey, and make the rest of the nodes 0.

$$P(n) = 0$$

$$P(x) = 1.$$

2. **If Prey is not at the node $\boldsymbol{x}$** : Since Prey is not at that node, the belief of that node is 0. So the belief that was earlier assigned to that node has to be distributed to the other nodes, since all the other nodes now have higher belief of having prey than earlier.

$$P(x) = 0.$$

$$P(n \text{ at } T_t \mid \textit{Prey is absent at } x \text{ at } T_t) =$$
$$P(\textit{Prey at } n \text{ at } T_t) * P(\textit{Prey absent at } x \text{ at } T_t \mid \textit{Prey in } N \text{ at } T_t) / P(\textit{Prey absent at } x \; T_t)$$
$$= P(n \text{ at } T_t)) / (1 - P(x \text{ at } T_t))$$

4.1.2 TRANSITION MODEL - Belief update mechanism at T_i after prey moves

After the prey moves, the Agent checks if the prey has moved to agent's current location. If the prey has moved, Game over, Agent Wins. If prey is not at the agent's current location, the agent will update the belief system considering all the possible prey movements.

For assigning belief to the node, we have to consider the possibility of each node to get prey from its neighbors. So the belief will depend on the following factors:

1. Choosing the neighbor from which the prey comes to the node
2. The belief of prey being in that node

So the Transition Model for Prey can be developed as below:

$$P(Prey\ at\ X\ at\ T_i) = \sum_{j=1}^{degree(x)} (P(Prey\ in\ X \mid Prey\ comes\ from\ n_j\ at\ T_{i-1}))$$

$$+$$

$$P(Prey\ at\ X \mid Prey\ stays\ in\ X at\ T_{i-1})$$

$$P(Prey\ in\ X \mid Prey\ comes\ from\ n_j\ at\ T_{i-1}) =$$
$$\text{P(Prey choosing X for its move from node } n_j\ at\ time\ T_{t-1}) * P(Prey\ in\ n_j)$$

4.1.3 Results of Agent3

	Agent	no_of_simulations	max_steps_allowed	Success rate	Failure rate	Hangs	% knowing prey location	% knowing predator location
0	Agent3	3000	50	76.70	18.13	155	5.67	100.0
1	Agent3	3000	100	82.80	17.10	3	5.80	100.0
2	Agent3	3000	150	83.90	16.10	0	5.64	100.0
3	Agent3	3000	200	81.97	18.03	0	5.79	100.0
4	Agent3	3000	250	84.17	15.83	0	5.71	100.0
5	Agent3	3000	300	80.63	19.37	0	5.73	100.0

Figure 4: Agent3 results

Since Agent 3 doesn't have certain information about the prey, it will not be able to achieve Agent 1 success rate. But it has more hangs in less-time steps environment. Agent 3 is able to know the prey location more than most of the agents.

4.1.4 Drawbacks of Agent3

Since Agent 3 inherits the same properties of Agent 1, it will have same drawbacks as Agent 1. Agent 3 cannot properly handle a situation when all its neighbors are very close to the predator.

4.2 Agent 4

Agent 4 is modified version of Agent 3 which is designed to outperform Agent 3. Agent 4 will use the same heuristic as Agent 2, that uses the probabilistic approach to determine the prey locations better. So, I have implemented Agent 4 such that it inherits Agent2 and Agent3 that will make Agent4 to mimic Agent3 along with functionalities of Agent2

Agent 4 will inherit Agent 2 to outperform Agent 3 using the following methods:

1. Estimation of future beliefs

2. Agent's next movement

Agent 4 will inherit Agent 3 for following methods:

1. To update the beliefs before and after prey moves

2. Survey

4.2.1 Results of Agent 4

	Agent	no_of_simulations	max_steps_allowed	Success rate	Failure rate	Hangs	% knowing prey location	% knowing predator location
0	Agent4	3000	50	84.43	7.40	245	5.97	100.0
1	Agent4	3000	100	90.07	9.70	6	6.12	100.0
2	Agent4	3000	150	91.10	8.90	0	5.87	100.0
3	Agent4	3000	200	91.63	8.37	0	5.84	100.0
4	Agent4	3000	250	90.13	9.87	0	6.04	100.0
5	Agent4	3000	300	91.80	8.20	0	5.79	100.0

Figure 5: Agent4 results

Agent 4, unlike Agent 2, has success rate as it has only partial information. Agent 4 hangs at low-step environment and its performance gets better with increasing time steps.

4.3 Agent 3 vs Agent 4

			Agent3	Agent3	Agent4	Agent4
	no_of_simulations	max_steps_allowed	Success rate	% knowing prey location	Success rate	% knowing prey location
0	3000	50	76.70	5.67	84.43	5.97
1	3000	100	82.80	5.80	90.07	6.12
2	3000	150	83.90	5.64	91.10	5.87
3	3000	200	81.97	5.79	91.63	5.84
4	3000	250	84.17	5.71	90.13	6.04
5	3000	300	80.63	5.73	91.80	5.79

Figure 6: Agent3 vs Agent4 results

5 Partial Predator Information Setting

In this scenario, the Agent is always aware of the location of the Prey but may not always be aware of the location of the Predator. The Agent can select a node to survey (anywhere in the graph) before moving in order to check if the Predator is there. Furthermore, Every time the agent visits a node where the Predator isn't, it learns where the Predator isn't. In this scenario, the Agent must keep track of a belief state—a collection of probabilities for each node that the Predator is there—for the Predator's location. These probability must be revised every time the Agent gains new information about the Predator. These probability must be adjusted each time the Predator is known to move. **For this environment, the Agent starts by knowing the location of the Predator.**

Note: In this setting, movement of predator is **distracted predator movement** - With chance 0.6, the predator will move to close the gap, but with probability 0.4, the predator travels to one of its neighbours uniformly at random. **And agents are aware of nature of predator's movements.**

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

Since Prey randomly chooses one of its neighbors, the probability solely depends upon the probability of choosing a particular neighbor and probability of the neighbors of having prey. Unlike prey, predator doesn't give equal probability to its neighbors. It follows shortest path to move towards the agent, and prioritise those neighbors through which the predator can follow shortest path.
This will be explained in detail in sections: 5.1.1 and 5.1.2

5.1 Agent5

Agents in this setting don't know the exact location of predator. So a belief system is used to know the location of predator by agents. Once Agent 5 comes to a conclusion about predator's location using the belief system it has built, it will follow the same process as Agent 1 for moving within the environment.

Unlike prey, predator doesn't give equal probability to it's neighbors to move ahead in the game. Say predator is at node x. It either moves to one of it's neighbors randomly with probability of 0.4 **or** to the neighbors closer to the agent with probability of 0.6, Agent will assign the probability of action performed by predator in previous time-stamp as below.

$$neighbors(x) = \{n_1, n_2, n_3\} \tag{1}$$

$$\begin{aligned} distance(n_1\ toAgent) &= 5 \\ distance(n_2\ toAgent) &= 3 \\ distance(n_3\ toAgent) &= 3 \end{aligned}$$

$$neighbors(x)\ that\ are\ closer\ to\ agent = \{n_2, n_3\} \tag{2}$$

$$P(x \to n_j) = 0.4 * P(choosing\ n_j\ from\ (1)) + 0.6 * P(choosing\ n_j\ from\ (2))$$

$$\begin{aligned} Pr(x \to n_1) &= 0.4 * Pr(choosing\ n_1\ from\ (1)) + 0.6 * Pr(choosing\ n_1\ from\ (2)) = 0.4 * (1/3) + 0.6 * (0) = \mathbf{0.133} \\ Pr(x \to n_2) &= 0.4 * Pr(choosing\ n_2\ from\ (1)) + 0.6 * Pr(choosing\ n_2\ from\ (2)) = 0.4 * (1/3) + 0.6 * (1/2) = \mathbf{0.433} \\ Pr(x \to n_3) &= 0.4 * Pr(choosing\ n_3\ from\ (1)) + 0.6 * Pr(choosing\ n_3\ from\ (2)) = 0.4 * (1/3) + 0.6 * (1/2) = \mathbf{0.433} \end{aligned}$$

As agent knows the initial location of predator, $belief s_0$ will be assigned based on the location of predator. Giving $belief_0$[predator location] = 1 and for all the remaining locations it will be 0.

Sequence of steps of agent, prey and predator at time T_t will be same as discussed in Partial Prey setting.

5.1.1 Belief update mechanism at T_i before predator moves

5.1.2 TRANSITION MODEL - Belief update mechanism at T_i after predator moves

At time T_i, if agent wants to update beliefs after the predator moves, it needs to consider the predator possible actions at time T_{i-1}.

If we are calculating the belief of node x to have predator at time $\boldsymbol{T_i}$, then we need to evaluate the possibilities of predator moving **from any one** of x's neighbors to x at $\boldsymbol{T_{i-1}}$.

say $neighbours(x) = \{n_1, n_2, n_3\}$

To have predator at x at time T_i, the following two events should occur at time T_{i-1} for **any one** of the neighbors of x, i.e $\{n_1, n_2, n_3\}$

1. Predator is present at n_j (neighbor of x)
2. movement of predator is from n_j to x (denoting it as $n_j \rightarrow x$)

$$P(Predator\ at\ X\ at\ time\ T_i) = \sum_{j=1}^{degree(X)} P(Predator\ at\ n_j\ at\ time\ T_i - 1 \mid Predator\ moves\ from\ n_j \rightarrow X)$$

As presence of predator and it's movement are independent for given predator, we can write

$$\sum_{j=1}^{degree(X)} P(Predator\ at\ n_j\ at\ time\ T_i - 1 \mid Predator\ moves\ from\ n_j \rightarrow X)$$
$$= P(Predator\ at\ n_j\ at\ time\ T_{i-1}) * P(Predator\ moves\ from\ n_j \rightarrow X\ at\ time\ T_{i-1})$$

$(Predator\ at\ n_j\ at\ time\ T_{i-1})$ is nothing but $belief s_{i-1}[n_j]$, we need to calculate $P(Predator\ moves\ from\ n_j \rightarrow X\ at\ time\ T_{i-1})$

Since we know how to calculate $Pr(x \rightarrow y)$, we can use this to calculate $P([at\ time\ T_i\ ]\ predator\ at\ X)$

$$P(Predator\ at\ X\ at\ time\ T_i\ ) = \sum_{j=1}^{degree(x)} P(Predator\ at\ n_j \mid n_j \rightarrow x\ at\ time\ T_{i-1}\ )$$
$$= \sum_{j=1}^{degree(x)} P(Predator\ at\ n_j \mid at\ time\ T_{i-1}) * Pr(n_j \rightarrow x\ at\ time\ T_{i-1})$$

In this way, we will calculate $belief s_i[x] = P([at\ time\ T_i\ ]\ predator\ at\ x)$; $\forall\ x \in Graphnodes$, except nodes that are present in $knowledge_i$. beliefs for nodes $\in knowledge_i$ is zero if knowledge is about absence of predator at node $\in knowledge_i$. If knowledge specifies presence of prey, then belief will be 1 for node which has predator and for all remaining 49 nodes beliefs will be 0 at time T_i.

5.1.3 Results of Agent5

	Agent	no_of_simulations	max_steps_allowed	Success rate	Failure rate	Hangs	% knowing prey location	% knowing predator location
0	Agent5	3000	50	77.30	21.57	33	100.0	57.11
1	Agent5	3000	100	78.53	21.47	0	100.0	57.05
2	Agent5	3000	150	76.97	23.03	0	100.0	56.71
3	Agent5	3000	200	77.23	22.77	0	100.0	57.12
4	Agent5	3000	250	80.37	19.63	0	100.0	57.04
5	Agent5	3000	300	80.97	19.03	0	100.0	56.12

Figure 7: Agent5 results

Agent 5 has a good success rate of 81% around. Agents 5, even though doesn't know the exact predator location during the game, is pretty good at knowing the predator's exact location.

5.1.4 Drawbacks of Agent5

Since Agent 5 inherits the same properties of Agent 1, it will have same drawbacks as Agent 1. Since Agent 5 doesn't know the exact location of predator, it cannot properly handle a situation when all its neighbors are very close to the predator.

5.2 Agent 6

Agent 6 is modified version of Agent 5 which is designed to outperform Agent 5. I have implemented Agent 4 such that it inherits Agent2 and Agent5 that will make Agent6 to mimic Agent5 along with functionalities of Agent2.

Agent 6 inherits Agent 2 to beat Agent 5 by using the following methods of Agent 2:

1. Estimation of future beliefs
2. Agents next movement

Agent 6 inherits Agent 5 with the following methods:

1. The belief updates for the predator before and after the predator moves
2. Survey result

5.2.1 Results of Agent 6

	Agent	no_of_simulations	max_steps_allowed	Success rate	Failure rate	Hangs	% knowing prey location	% knowing predator location
0	Agent6	3000	50	82.93	11.43	168	100.0	53.57
1	Agent6	3000	100	87.50	12.27	6	100.0	54.05
2	Agent6	3000	150	86.97	13.03	0	100.0	53.51
3	Agent6	3000	200	88.00	12.00	0	100.0	54.37
4	Agent6	3000	250	87.50	12.50	0	100.0	53.98
5	Agent6	3000	300	87.93	12.07	0	100.0	53.15

Figure 8: Agent 6 results

Agent 6 has a very good success rate of around 88%. It has a few hangs that vanish after 150-200 time steps.

5.3 Agent 5 vs Agent 6

			Agent5	Agent5	Agent6	Agent6
	no_of_simulations	max_steps_allowed	Success rate	% knowing predator location	Success rate	% knowing predator location
0	3000	50	77.30	57.11	82.93	53.57
1	3000	100	78.53	57.05	87.50	54.05
2	3000	150	76.97	56.71	86.97	53.51
3	3000	200	77.23	57.12	88.00	54.37
4	3000	250	80.37	57.04	87.50	53.98
5	3000	300	80.97	56.12	87.93	53.15

Figure 9: Agent5 vs Agent6 results

6 Combined Partial Information Setting

In this scenario, the Agent does not necessarily know where the Predator or Prey are. So the Agent has to keep track of belief states for both the Predator and the Prey, updating them based on information collected and knowledge of the actions of the two players. For this environment, the Agent starts by knowing the location of the Predator.

6.1 Agent 7

Whenever it is this Agent's turn to move, if it is not currently certain where the predator is, it will survey in accordance with Agent 5. If it knows where the Predator is, but not the Prey, it will survey in accordance with Agent 3. So I am modelling Agent 7 that inherits Agent 3 and Agent 5.

I am considering below actions will occur at given time T_t sequentially,

1. Agent surveys a node according to rules at T_t
2. Agent will update the $\boldsymbol{prey - beliefs_t}$ that it gets from **Agent 3**, if it certainly knows where the predator is. If it doesn't know, it will update the $\boldsymbol{predator - beliefs_t}$ which is received from **Agent 5**.
3. Agent moves to a new location (one of it's neighbors) at T_t
4. Agent updates $\boldsymbol{prey - beliefs_t}$ using its new location at T_t
5. Agent updates $\boldsymbol{predator - beliefs_t}$ using its new location
6. Prey moves at T_t
7. Agent updates the $\boldsymbol{prey - beliefs_t}$ according to the transition model of **Agent 3** at T_t
8. Predator moves at T_t
9. Agent updates the $\boldsymbol{predator - beliefs_t}$ according to the transition model of **Agent 5** at T_t

6.1.1 Results of Agent7

	Agent	no_of_simulations	max_steps_allowed	Success rate	Failure rate	Hangs	% knowing prey location	% knowing predator location
0	Agent7	3000	50	65.37	19.87	443	1.77	55.05
1	Agent7	3000	100	72.20	26.73	32	1.82	55.30
2	Agent7	3000	150	75.33	24.63	0	1.77	55.16
3	Agent7	3000	200	74.50	25.50	0	1.62	54.95
4	Agent7	3000	250	74.97	25.03	0	1.83	55.76
5	Agent7	3000	300	74.27	25.73	0	1.73	54.58

Figure 10: Agent7 results

1. Agent 7, even though doesn't have the information about the exact predator and prey location, has a good success rate of 75%
2. It is able to locate the predator more than half of the times even in low time step environments

6.1.2 Drawbacks of Agent7

Since Agent 7 inherits Agent 3 and 5, this agent also faces the same limitation as the former agents.

6.2 Agent 8

Agent 8 is modified version of Agent 7 which is designed to outperform Agent 7.I have implemented Agent8 by inheriting Agent2 and Agent7. Agent8 will have methods used in Combined Partial Information setting by inheriting Agent7 and will have future probabilistic model approach through Agent 2.

Agent 8 inherits Agent 2 to beat Agent 7 by using the following methods of Agent 2:

1. Estimation of future beliefs of prey
2. Agents next movement

Agent 8 inherits Agent 7 for the following methods:

1. The belief updates for the predator before and after the predator moves
2. The belief updates for the prey before and after the prey moves
3. Survey result

6.2.1 Results of Agent8

	Agent	no_of_simulations	max_steps_allowed	Success rate	Failure rate	Hangs	% knowing prey location	% knowing predator location
0	Agent8	3000	50	68.10	12.63	578	1.88	53.96
1	Agent8	3000	100	80.17	17.47	71	1.67	54.09
2	Agent8	3000	150	81.13	18.37	15	1.89	52.53
3	Agent8	3000	200	82.57	17.30	3	1.70	54.13
4	Agent8	3000	250	83.43	16.53	0	1.80	54.15
5	Agent8	3000	300	81.50	18.50	0	1.73	52.92

Figure 11: Agent8 results

6.3 Agent7 vs Agent 8

	no_of_simulations	max_steps_allowed	Agent7 Success rate	Agent7 % knowing prey location	Agent7 % knowing predator location	Agent8 Success rate	Agent8 % knowing prey location	Agent8 % knowing predator location
0	3000	50	65.37	1.77	55.05	68.10	1.88	53.96
1	3000	100	72.20	1.82	55.30	80.17	1.67	54.09
2	3000	150	75.33	1.77	55.16	81.13	1.89	52.53
3	3000	200	74.50	1.62	54.95	82.57	1.70	54.13
4	3000	250	74.97	1.83	55.76	83.43	1.80	54.15
5	3000	300	74.27	1.73	54.58	81.50	1.73	52.92

Figure 12: Agent7 vs Agent8 results

7 Analysis And Report

7.1 Defective Survey Drone

For the Combined Partial Information Setting, the survey drone is defective in such a way that if something actually occupies a node being surveyed, there is a 0.1 probability that it gets reported as unoccupied (a false negative).

7.1.1 Agent 7: Without the defective drone belief updates

The Agent will use the defective drone to survey the nodes instead of the normal survey drone. The survey will always be true when it returns that it has a prey/predator. But when it returns that there is no prey/predator, there is a 0.1 chance that there is a prey/predator.

I am considering below actions will occur at given time T_t sequentially,

1. Agent will use the defective drone only for prey if it certainly knows where the predator is, at T_t. If not, it will use the defective drone for predator.
2. Agent will update the $\boldsymbol{prey - beliefs_t}$ that it gets from **Agent 3**, if it certainly knows where the predator is. If not, it will update the $\boldsymbol{prey - beliefs_t}$ and $\boldsymbol{predator - beliefs_t}$ which is received from **Agent 3** and**Agent 5** respectively.
3. Agent moves to a new location (one of it's neighbors) at T_t
4. Agent updates $\boldsymbol{prey - beliefs_t}$ using it's new location at T_t
5. Prey moves at T_t
6. Agent updates the $\boldsymbol{prey - beliefs_t}$ according to the transition model of **Agent 3** at T_t
7. Predator moves at T_t
8. Agent updates the $\boldsymbol{predator - beliefs_t}$ according to the transition model of **Agent 5** at T_t

Since in this section, we are considering that there is no effect on the beliefs due to the defective survey drone, the beliefs $prey - beliefs_t$ and $predator - beliefs_t$ are updated according to the rules of Agent3 and Agent5 respectively.

Results of Agent 7 - Faulty Survey

	Agent	no_of_simulations	max_steps_allowed	Success rate	Failure rate	Hangs	% knowing prey location	% knowing predator location
0	FaultyAgent7	3000	50	51.77	40.87	221	1.75	37.95
1	FaultyAgent7	3000	100	54.40	45.27	9	1.88	37.43
2	FaultyAgent7	3000	150	58.30	41.67	0	1.77	37.21
3	FaultyAgent7	3000	200	57.07	42.93	0	1.76	38.44
4	FaultyAgent7	3000	250	57.80	42.20	0	1.90	38.16
5	FaultyAgent7	3000	300	56.23	43.77	0	1.87	37.57

Figure 13: Agent7 - Faulty Survey results

So the Agent 7 with defective drone can directly inherit the Normal Agent 7 and implement it using the defective survey drone.

7.1.2 Agent 8: Without the defective drone belief updates

Since Agent8 also follows the rules of Agent 7, Agent 8 with defective drone can directly inherit the Normal Agent 8 and implement it using the defective survey drone.

	Agent	no_of_simulations	max_steps_allowed	Success rate	Failure rate	Hangs	% knowing prey location	% knowing predator location
0	FaultyAgent8	3000	50	52.30	38.17	285	2.06	36.65
1	FaultyAgent8	3000	100	59.97	39.60	12	1.84	36.70
2	FaultyAgent8	3000	150	56.57	43.43	0	1.76	36.87
3	FaultyAgent8	3000	200	58.90	41.10	0	1.76	35.97
4	FaultyAgent8	3000	250	57.87	42.13	0	1.76	37.05
5	FaultyAgent8	3000	300	58.03	41.97	0	1.84	37.06

Figure 14: Agent8 - Faulty Survey results

7.1.3 Agent 7 vs Agent 8 - Without the defective drone belief updates

			FaultyAgent7	FaultyAgent7	FaultyAgent7	FaultyAgent8	FaultyAgent8	FaultyAgent8
	no_of_simulations	max_steps_allowed	Success rate	% knowing prey location	% knowing predator location	Success rate	% knowing prey location	% knowing predator location
0	3000	50	51.77	1.75	37.95	52.30	2.06	36.65
1	3000	100	54.40	1.88	37.43	59.97	1.84	36.70
2	3000	150	58.30	1.77	37.21	56.57	1.76	36.87
3	3000	200	57.07	1.76	38.44	58.90	1.76	35.97
4	3000	250	57.80	1.90	38.16	57.87	1.76	37.05
5	3000	300	56.23	1.87	37.57	58.03	1.84	37.06

Figure 15: Faulty Agent7 vs Agent8

7.1.4 Agent 7: With the defective drone belief updates

When the Agent 7 uses the defective drone, the beliefs of the prey and the predator are effected and needs to be updated accordingly.

We know that when survey gives information that either prey or predator is present, that that is definitely True, since the survey does not give False positives.

The issue arises when the survey gives information of absence of prey or predator at node as we cannot certainly say if that is True or not.(As faulty drone can produce only false negative results).

So we can come up with 2 possibilities that could produce the absence of the participants in the survey:

1. If neither prey nor predator are present in that location (Gained knowledge from survey is True)
2. There is either of the participants in that location, but the survey has returned it as absent(Gained knowledge from survey is False)

Now to calculate the beliefs of each node:

1. When the surveyed node returns absence, then there is still 0.1 possibility that the survey was wrong. This will mean that there is still 0.1*P(Survey Node) probability of having prey there.
2. The remaining probability of the surveyed node is normally distributed to the remaining nodes.

To formulate mathematically, we can represent it as follows:

$$P(Survey\ Node\ at\ time\ T_i\ ) = 0.1 * P(Survey\ Node\ at\ T_{i-1})$$
$$P(Remaining\ Nodes\ at\ time T_i) = 1 - 0.9 * P(Survey\ Node\ at\ time\ T_{i-1})$$

Results of Agent 7 with Belief updates

	Agent	no_of_simulations	max_steps_allowed	Success rate	Failure rate	Hangs	% knowing prey location	% knowing predator location
0	SmartAgent7	3000	50	56.67	33.63	291	1.82	40.93
1	SmartAgent7	3000	100	65.27	34.43	9	1.79	39.68
2	SmartAgent7	3000	150	62.60	37.40	0	1.72	39.48
3	SmartAgent7	3000	200	64.77	35.23	0	1.77	39.88
4	SmartAgent7	3000	250	60.10	39.90	0	1.83	40.71
5	SmartAgent7	3000	300	65.03	34.97	0	1.72	40.30

Figure 16: Agent7 -with Faulty Survey Beliefs results

7.1.5 Agent 8: With the defective drone belief updates

Agent 8 is just the modified version of Agent7 in this defective survey setting. So I have implemented Agent 8 by inheriting Agent 7 of this setting (with the belief updates) and Agent 2 to inherit the probabilistic model.

	Agent	no_of_simulations	max_steps_allowed	Success rate	Failure rate	Hangs	% knowing prey location	% knowing predator location
0	SmartAgent8	3000	50	59.53	28.27	366	1.79	39.16
1	SmartAgent8	3000	100	65.97	32.90	33	1.65	38.03
2	SmartAgent8	3000	150	69.97	30.03	0	1.80	38.00
3	SmartAgent8	3000	200	69.23	30.73	0	1.71	38.14
4	SmartAgent8	3000	250	67.83	32.17	0	1.83	38.15
5	SmartAgent8	3000	300	67.33	32.67	0	1.70	37.89

Figure 17: Agent8 -with Faulty Survey Beliefs results

7.1.6 Agent 7 vs Agent 8 - With defective drone belief updates

			SmartAgent7	SmartAgent7	SmartAgent7	SmartAgent8	SmartAgent8	SmartAgent8
	no_of_simulations	max_steps_allowed	Success rate	% knowing prey location	% knowing predator location	Success rate	% knowing prey location	% knowing predator location
0	3000	50	56.67	1.82	40.93	59.53	1.79	39.16
1	3000	100	65.27	1.79	39.68	65.97	1.65	38.03
2	3000	150	62.60	1.72	39.48	69.97	1.80	38.00
3	3000	200	64.77	1.77	39.88	69.23	1.71	38.14
4	3000	250	60.10	1.83	40.71	67.83	1.83	38.15
5	3000	300	65.03	1.72	40.30	67.33	1.70	37.89

Figure 18: Smart Agent 7 vs Agent 8

7.2 Agent 9

I have few ideas which we can use to implement Agent 9, to make it perform better in Combined Partial Information setting with the defective survey.

1. Implement a probabilistic model to determine both future prey and future predator locations.

   I have already tried to implement probabilistic model in Agent 2. If we are able to develop similar one with the future predator locations as well, then this would create a great model which will help to gain more success rate in Combined Partial environments.

2. Create the agent movement system in such a way that agent ensures that there is no predator within a particular radius distance from the agent.

   To compute distance and ensure to be within a radius distance away from agent needs a lot of computations. We can use Floyd Warshall, which I have used to compute all pairs shortest distance.