Project 2: Intruder Alert

Ahmed Khan and Elian Deogracia-Brito

June 2024

Abstract

This document presents a comparison of three different algorithms in a grid-based simulation aimed at searching and capturing a hypothetical mouse using probabilistic sensing.

1 Introduction

1.1 Background

It's another day on the deep space vessel *Archaeopteryx* and you are a lonely bot. You're responsible for the safety and security of the ship while the crew is in hibernation. As an example, in the case that the ship picks up a deep space mouse, it is your responsibility to catch it and release it back into the wild (they can survive in the vacuum of space for up to 10,000 years).

1.2 Objectives

- Bot 1 - Move to the location with highest probability, sense.
- Bot 2 - Move toward the location with highest probability, alternate moving and sensing.
- Bot 3 - A bot of your own design.

2 Data and Analysis

2.1 Four Rooms Problem

Suppose the ship had four rooms, A, B, C, D, and the probability the mouse was in each of them was 0.4, 0.3, 0.2, 0.1. If you looked in room B, and the mouse was not there, the probability of it being in A is calculated by:

2.1.1 Solution

Given:

- $P(A) = 0.4$ (Probability the mouse is in room A)
- $P(\neg B) = 1 - P(B) = 0.7$ (Probability the mouse is not in room B)

The probability that the mouse is in room A given that it was not in room B can be found using:

$$P(A \mid \neg B) = \frac{P(A)}{P(\neg B)} = \frac{0.4}{0.7} \approx 0.571$$

2.2 Updating Probabilistic Knowledge Base

The probabilistic knowledge base updates based on many scenarios and different events that occur while in the process of searching for the mouse. These different events result in different changes to the overall probability of cells. We utilize the Bayes' theorem equation and calculate the probabilities using the formula:

$$P(\text{Beep} \mid d) = e^{-\alpha\cdot(d-1)}$$

2.2.1 If the sensor beeps

$$A = \text{The event of receiving a beep.}$$

$$B_{ij} = \text{The event the mouse is at location (i, j).}$$

Let d denote the Manhattan distance of (i, j) to a the bot's current cell

$$P(A \mid B_{ij}) = e^{-\alpha(d_{ij}-1)}$$

$$P(B_{ij} \mid A) = \frac{P(A \mid B_{ij}) \cdot P(B_{ij})}{P(A)} \quad \text{Bayes' Theorem}$$

Given an $N \times N$ grid, where N represents the height & width of a ship, we will assume the probability of a location initially containing the mouse to be $\frac{1}{N^2}$.

$$P(B_{ij}) = \frac{1}{N^2}$$

When we calculate $P(A)$, we want to find the overall probability of receiving a beep (A), regardless of the location of the mouse. This is the sum over all possible locations (k, l). Multiplying $P(A \mid B_{kl})$ by $P(B_{kl})$ provides a joint probability that the mouse is at the location (k, l) (B) AND we receive a beep (A). Summing this product over all possible locations (k, l) gives $P(A)$ across all scenarios where the mouse might be.

$$P(A) = \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} P(A \mid B_{kl}) \cdot P(B_{kl})$$

Since each cell has an equal probability of containing the mouse we can factor out $P(B_{kl})$ from the sum.

$$= \frac{1}{N^2} \cdot \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} P(A \mid B_{kl})$$

$$= \frac{1}{N^2} \cdot \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} e^{-\alpha(d_{kl}-1)}$$

So based on the above expressions:

$$P(B_{ij} \mid A) = \frac{e^{-\alpha(d_{ij}-1)} \cdot \cancel{\frac{1}{N^2}}}{\cancel{\frac{1}{N^2}} \cdot \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} e^{-\alpha(d_{kl}-1)}}$$

$$= \frac{e^{-\alpha(d_{ij}-1)}}{\sum_{k=0}^{N-1} \sum_{l=0}^{N-1} e^{-\alpha(d_{kl}-1)}}$$

Cancellation and the factorization of $P(B)$ only applies in the initial iteration. In reality we have to maintain $P(B)$ in the equation since our probability map updates.

$$P(\text{Mouse at cell } x \mid \text{Beep}) = \frac{P(\text{Beep} \mid \text{Mouse at cell } x) \cdot P(\text{Mouse at cell } x)}{\sum_{all\ cells} P(\text{Beep} \mid \text{Mouse at cell } x) \cdot P(\text{Mouse at cell } x)}$$

The updated probability at a certain cell given a beep is found by multiplying the probability of a beep occurring given the mouse is at the cell by the current probability and normalized(divided) by the summation of the prior.

2.2.2 If the sensor does not beep

$$P(\text{Mouse at cell } x \mid \neg\text{Beep}) = \frac{(1 - P(\text{Beep} \mid \text{Mouse at cell } x)) \cdot P(\text{Mouse at cell } x)}{\sum_{all\, cells}(1 - P(\text{Beep} \mid \text{Mouse at cell } x)) \cdot P(\text{Mouse at cell } x)}$$

The updated probability at a certain cell given a beep is found by multiplying the probability of no beep occurring given the mouse is at the cell by the current probability and normalized(divided) by the summation of the prior.

2.3 Handling Multiple Mice

2.3.1 Two Stationary Mice

When there are two stationary mice, the probability calculations become more complex as we need to account for the presence of two mice on the grid simultaneously.

Initially, the probability of any cell containing a mouse is $\frac{2}{N^2}$, where N is the size of the grid.

If the sensor beeps, indicating at least one mouse is nearby, we update the probability map as follows:

$$P(B_{ij} \mid A) = \frac{P(A \mid B_{ij}) \cdot P(B_{ij})}{P(A)}$$

$$P(A \mid B_{ij}) = 1 - (1 - e^{-\alpha(d_{ij}-1)})^2$$

This formula accounts for the combined probability of at least one of the two mice being within sensor range.

If the sensor does not beep, indicating no mice are nearby, the update is:

$$P(B_{ij} \mid \neg A) = \frac{P(\neg A \mid B_{ij}) \cdot P(B_{ij})}{P(\neg A)}$$

$$P(\neg A \mid B_{ij}) = (1 - e^{-\alpha(d_{ij}-1)})^2$$

This formula represents the probability of neither mouse being within sensor range.

2.3.2 Two Moving Mice

When there are two moving mice, the complexity further increases as we must consider the movements of both mice.

Initially, the probability of any cell containing a mouse is $\frac{2}{N^2}$, similar to the stationary case.

The transition model needs to account for the independent movements of both mice. For each cell, we calculate the probability of it being occupied by each mouse and then combine these probabilities:

$$P(B_{ij} \mid \text{next}) = 1 - 2\left(1 - \sum_{\text{direction}} P(B_{ij} \mid \text{direction}) \cdot P(\text{direction})\right)$$

where $P(\text{direction})$ represents the probability of each possible move.

If the sensor beeps:

$$P(B_{ij} \mid A) = \frac{P(A \mid B_{ij}) \cdot P(B_{ij})}{P(A)}$$

$$P(A \mid B_{ij}) = 1 - (1 - e^{-\alpha(d_{ij}-1)})^2$$

If the sensor does not beep:

$$P(B_{ij} \mid \neg A) = \frac{P(\neg A \mid B_{ij}) \cdot P(B_{ij})}{P(\neg A)}$$

$$P(\neg A \mid B_{ij}) = (1 - e^{-\alpha(d_{ij}-1)})^2$$

In both cases, the key is to calculate the probabilities of each mouse independently and then combine them to update the probability map. While the complexity of these calculations increases, they follow the same fundamental principles as the single mouse case.

2.4 Design and Algorithm for Bot 3

Bot 3 is designed to take core features from Bot 1 and 2 and improve upon them by incorporating predictive and reactive features. The key design choices are as follows:

2.4.1 Algorithm

- **Initialization, Probability Map, Transition Model, and Gradient Map:** The algorithm begins by initializing a probability map that uniformly distributes the probability of the mouse's location across all open cells. It then proceeds to create a transition model that utilizes equal probability to all valid moves (left, right, up, down, and none [not moving]]) from each cell. Alongside this, the initialization of a gradient map is done to track the changes in probability values which will further assist in making more informed decisions on the bot's movements.
- **Predict Step:** Before sensing, Bot 3 predicts the next state of the probability map using the transition model. This step updates the probability of the mouse's location based on its possible movements.
- **Update Step:** When the bot senses, it updates the probability map based on whether the sensor beeped or not. This update uses Bayes' theorem to adjust the probabilities of the mouse's location. To better reflect the changes in probabilities, the gradient map is also updated.
- **Movement Planning:** Bot 3 plans its path to the cell with the highest updated probability of containing the mouse while also considering the gradient values to make more efficient movements.
- **Capture Attempt:** After moving, Bot 3 attempts to capture the mouse. If the mouse is not captured, Bot 3 continues to update the transition model and probability map, smoothing and normalizing probabilities to a probability of 1, to ensure efficiency in further attempts.
- **Max Probability Repeats:** If the max probability location remains the same throughout multiple iterations, Bot3 keeps track of the no. of times a target does not consecutively move. When an arbitrary no. is hit (5), Bot3 will append to its existing path given the new sensor information.
- **Multiple Targets:** When Bot3 finds a given mouse (Stationary or Stochastic) it will adjust its probability map, pretending no beep was given but multiplied with a scale factor of $\frac{1}{1000}$. This is mainly to prevent the removal of nearby important tiles the other mouse/mice may be at.

The algorithm for Bot 3 combines elements of prediction and Bayesian updating to dynamically adjust its search strategy. By continually refining its probability map and transition model, Bot 3 can effectively locate and capture the mouse with fewer moves compared to Bot 1 and Bot 2.

3 Evaluation and Comparison

3.1 Stationary Mouse

The following plot shows the average moves taken by Bots 1, 2, and 3 to catch a stationary mouse as a function of α. The data suggests that Bot 3 consistently performs better across various values of α.

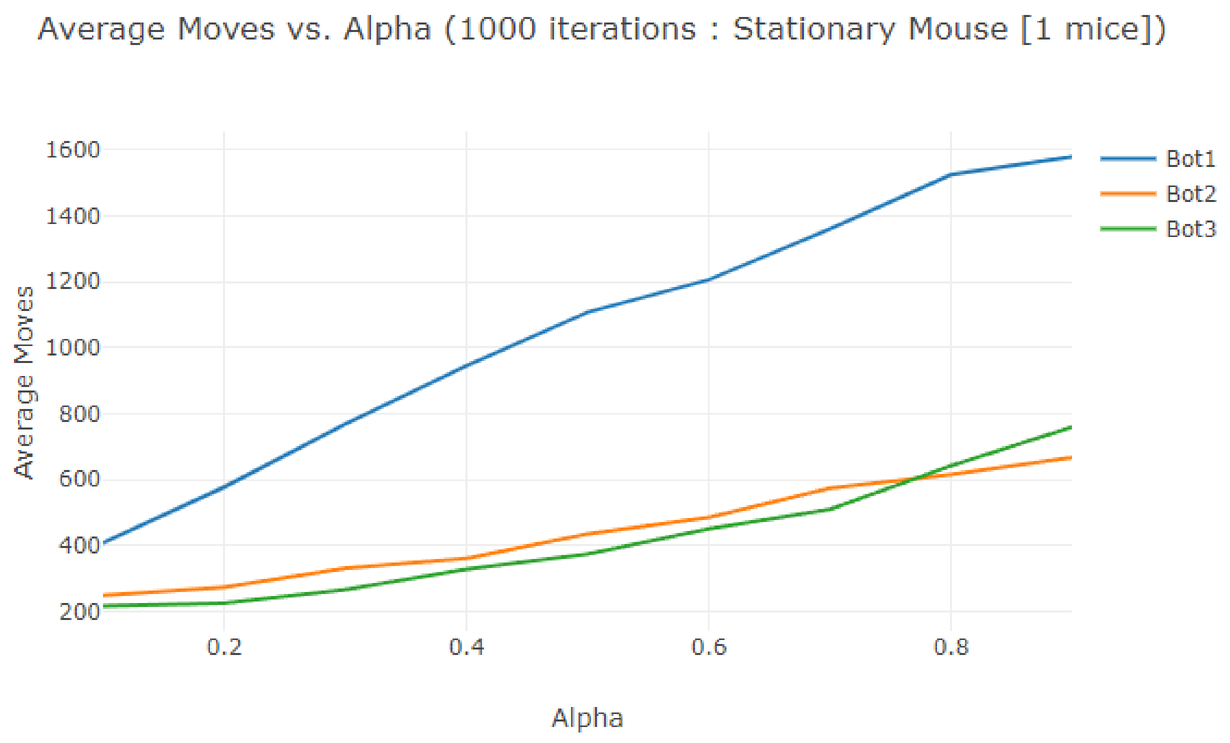

Figure 1: Average Moves vs. Alpha (Stationary Mouse)

3.1.1 Analysis

- **Bot 1:** Demonstrates variability in performance as α changes, as well as including the highest number of moves required out of all the bots. Shows a linear increase relative to around 400 moves for every 0.2 alpha indicating the Bot 1 algorithm is very inefficient in reducing probable areas for the mouse.
- **Bot 2:** Shows a more consistent performance with fewer moves compared to Bot 1, but still shows slight variation depending on α. Does seem to also perform better than Bot 3 at very high levels of alpha but this could be due to testing error. Seemingly has a very slight linear increase indicating a very effective and efficient algorithm albeit just less efficient than Bot 3.
- **Bot 3:** Consistently requires the fewest moves across all values of α until the very late alpha levels, indicating an efficient design capable of performing well amongst all alpha levels on average.

3.2 Stochastic Mouse

The following plot shows the average moves taken by Bots 1, 2, and 3 to catch a stochastic mouse as a function of α. The data suggests that Bot 3 consistently performs better across various values of α.

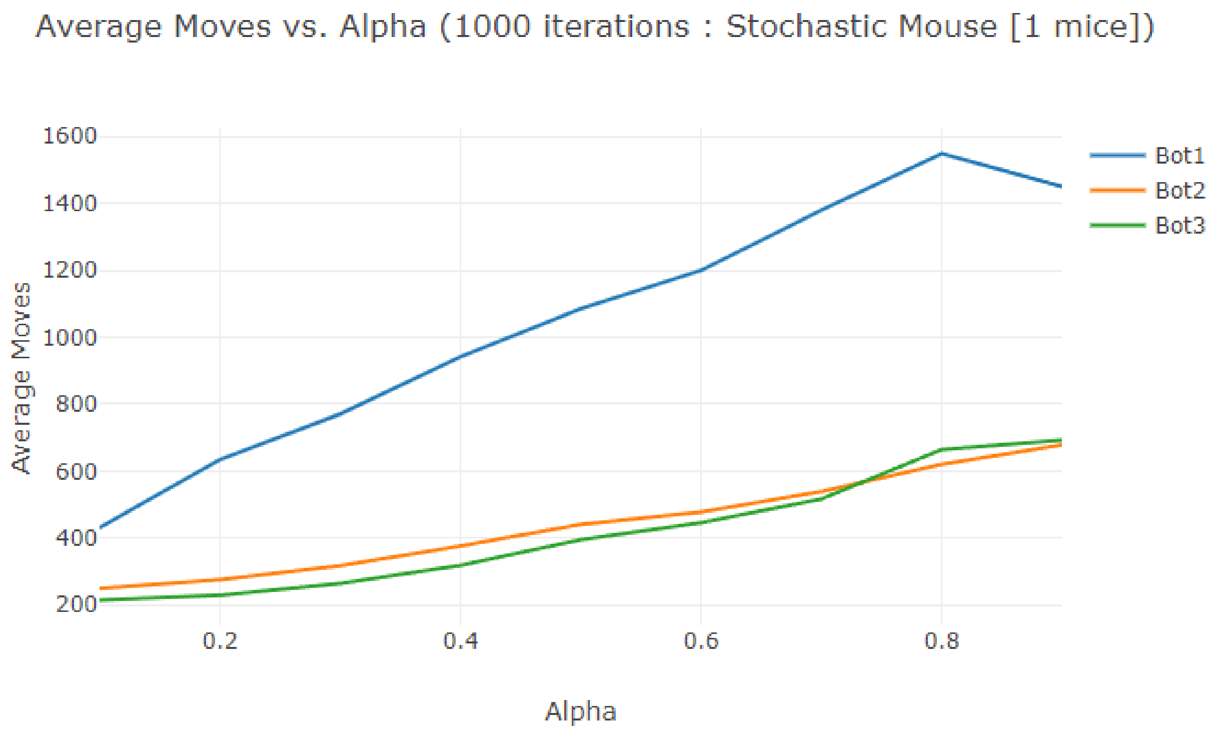

Figure 2: Average Moves vs. Alpha (Stochastic Mouse)

3.2.1 Analysis

- **Bot 1:** Demonstrates variability in performance as α changes, as well as including the highest number of moves required out of all the bots. Performs similar to it's stationary counterpart with a slight increase in number of moves needed.
- **Bot 2:** Shows a more consistent performance with fewer moves compared to Bot 1, but still shows slight variation depending on α. On average, needed more moves to locate the mouse but a very negligible amount. With a very low amount of moves needed, Bot 2 is still a very efficient algorithm showing a slow linear increase in moves per alpha level.
- **Bot 3:** Consistently requires the fewest moves across all values of α until the latter levels, indicating an efficient design capable of performing well amongst all levels on average. Shows signs of similar move amounts for both stationary and stochastic mice with stochastic mice requiring a tiny increase of amount of moves to locate.

3.3 Two Stochastic Mice

The following plot shows the average moves taken by Bots 1, 2, and 3 to catch two stochastic mice as a function of α. The data suggests that Bot 3 consistently performs better across various values of α.

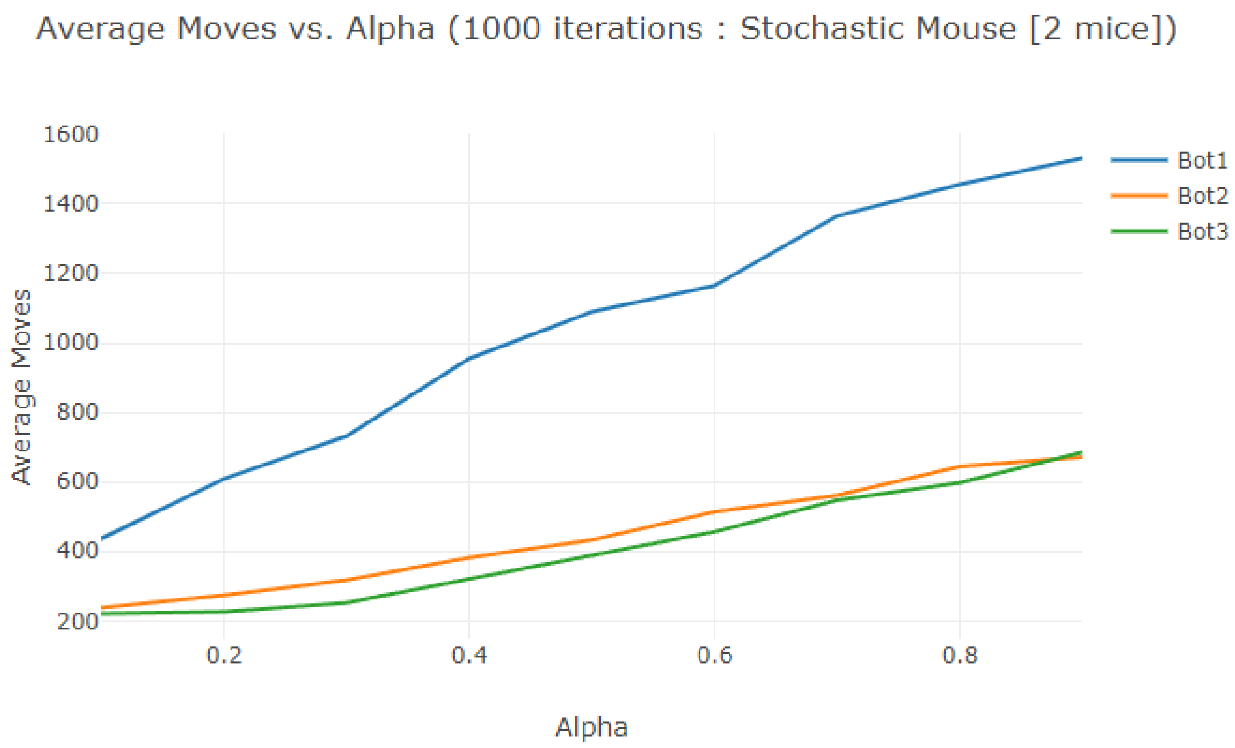

Figure 3: Average Moves vs. Alpha (Two Stochastic Mice)

3.3.1 Analysis

- **Bot 1:** Shows a heavy linear increase in moves required to find both mice especially at higher alpha levels indicating a very inefficient algorithm and pathing. Compared to a single mice, Bot 1 required more moves and showed a steeper increase in moves needed throughout the alpha levels measured.

- **Bot 2:** Bot 2 effectively performs better than Bot 1 but fell just short of Bot 3 in efficiency especially in lower alpha levels. With the amount of moves increasing compared to a single mouse simulation, Bot 2 is still an efficient algorithm. Shows a heavier slope increase compared to prior bot simulations.

- **Bot 3:** Shows to be the most efficient algorithm, requiring the fewest amount of moves across any level of alpha as well as showing less variability in the alpha levels meaning it has the algorithmic potential to be efficient and precise with any sensitivity. Requires more moves than the stationary mouse but similar to the one stochastic mouse, Bot 3 is the overall best choice in it's predictive and reactive assessments.