Project 2: Space Rats Writeup

Krithish Ayyappan and Lakshya Gour

November 2024

Introduction/Summary

In this project, we designed a bot capable of locating itself in a spaceship and catching a space rat. As per the project description the spaceship was a 30x30 grid and the bot is capabale of sensing, pinging, and moving at any given timestep.

For our data analysis, we conducted 100 simulations for each alpha value, unless otherwise stated. This comprehensive approach allowed us to gather robust data and draw meaningful conclusions about the bot's performance across different scenarios.

We began by developing a baseline bot that operates in two phases. In Phase 1, the bot identifies its location by sensing the number of blocked neighboring cells and moving in the most common open direction. In Phase 2, the bot tracks the space rat by alternating between using the space rat detector and moving towards the cell with the highest probability of containing the rat. The knowledge base was updated using Bayes' theorem based on whether a ping was heard or not.

After establishing the baseline bot, we created an improved bot that enhanced Phase 2 by incorporating utility functions to decide when to ping and when to move. This approach significantly reduced the timesteps required to find the rat compared to the baseline bot, especially for stationary rats.

However, we soon realized that the improved bot struggled when the rat was moving. The bot's performance drastically decreased, requiring significantly more timesteps to find the rat. To address this issue, we conducted further testing and developed a second improved bot that incorporated a transition model. This model accounted for the rat's random movements by redistributing the probabilities across neighboring cells, reflecting the rat's potential new positions.

Through rigorous testing and analysis, we found that the second improved bot performed significantly better than the baseline bot and the initial improved bot in all scenarios for alpha values greater than 0.15. This project demonstrated the effectiveness of combining Bayesian networks, transition models, and utility functions to design a bot capable of efficiently locating and catching a moving space rat.

Check out animations for simulation for each bot in the files we sent

Question 1: Updating the Knowledge Base Based on Ping or No Ping

To update the space rat knowledge base based on whether the bot receives a ping or not, we used Bayes' theorem. The knowledge base consists of the probabilities of the rat being in each cell. Let the probability of the rat being in cell j be $P(R_j)$.

Bayes' Theorem

$$P(R_j|\text{ping}) = \frac{P(\text{ping}|R_j) \cdot P(R_j)}{P(\text{ping})}$$

$$P(R_j|\text{no ping}) = \frac{P(\text{no ping}|R_j) \cdot P(R_j)}{P(\text{no ping})}$$

Where:

- $P(\text{ping}|R_j)$: Likelihood of hearing a ping given the rat is in cell j.
- $P(R_j)$: Prior probability of the rat being in cell j.
- $P(\text{ping})$ and $P(\text{no ping})$: Total probabilities of hearing a ping and not hearing a ping, respectively.

Likelihood Calculation

The likelihood of hearing a ping given the rat is in cell j is:

$$P(\text{ping}|R_j) = e^{-\alpha(d(i,j)-1)}$$

where $d(i,j)$ is the Manhattan distance between the bot's position i and the rat's position j.

The likelihood of not hearing a ping given the rat is in cell j is:

$$P(\text{no ping}|R_j) = 1 - e^{-\alpha(d(i,j)-1)}.$$

Update Formulas

When a ping is heard:

$$P(R_j|\text{ping}) = \frac{e^{-\alpha(d(i,j)-1)} \cdot P(R_j)}{\sum_k e^{-\alpha(d(i,k)-1)} \cdot P(R_k)}.$$

When no ping is heard:

$$P(R_j|\text{no ping}) = \frac{(1 - e^{-\alpha(d(i,j)-1)}) \cdot P(R_j)}{\sum_k (1 - e^{-\alpha(d(i,k)-1)}) \cdot P(R_k)}.$$

Question 2: Design and Algorithm for the Improved Bot

Initialization

- Initialize the knowledge base with uniform probabilities across all open cells.
- Set the initial positions of the bot and the rat.

Phase 1: Locate the Bot

1. Alternate between sensing the number of blocked neighbors and moving in the most common open direction.
2. Update the knowledge base by ruling out positions that do not match the sensed blocked neighbors.
3. Continue until the bot's position is identified.

Phase 2: Track the Rat

1. Alternate between using the space rat detector and moving towards the cell with the highest probability of containing the rat.
2. Use Bayes' theorem to update the knowledge base based on whether a ping is heard or not.
3. Calculate the entropy of the knowledge base to decide whether to listen or move: The entropy of the knowledge base is calculated as:

$$\text{Entropy} = -\sum_i P(R_i) \log(P(R_i) + \epsilon)$$

Where:

- $P(R_i)$ is the probability of the rat being in cell i.
- ϵ is a small constant to prevent taking the logarithm of zero.

The maximum entropy $\text{Entropy}_{\text{max}}$ is calculated as:

$$\text{Entropy}_{\text{max}} = \log(\text{Open Cells})$$

This gives us an entropy ratio of :

$$\text{Entropy}_{\text{ratio}} = \frac{\text{Entropy}}{\text{Entropy}_{\text{max}}}$$

- If the entropy ratio is high (uncertainty is high), listen to reduce uncertainty.
- If the entropy ratio is low, move towards the most probable location.

4. Use BFS to find the shortest path to the cell with the highest probability.

Decision Making

- Calculate the expected utilities of moving and listening.
- Choose the action with the higher utility, considering the costs of moving and listening.
- Penalize excessive listening to encourage movement if no progress is made.

Handling Movement

- Move towards the cell with the highest probability which we find using a BFS shortest path algorithm.

Termination

The bot stops when it finds the rat or reaches a maximum timestep limit (which we arbitraily chose as 20000).

Question 3: Evaluate Base Bot vs. Improved

We focused more on improving Phase 2 of the bot as Phase 1 averaged to 10 timesteps to find itself, leaving little room for improvement (see Figure **??**). However, Phase 2 had a lot of potential with utilizing utility functions to determine when to ping and when to move to find the rat in significantly less timesteps.

Figure 1 shows the number of timesteps required for the bot to find itself during Phase 1 of the localization process. As observed, the bot consistently required approximately 10 timesteps, with little variance. This stability indicates that further optimization was unnecessary for this phase as both are similar.

We also notice that for the timesteps for sensing closed neighbors is relatively the same as we have not improved upon Phase1. Our improved bot and baseline bot have relativelty the same statistics as they have the same approach for locating themselves.

However, our improved significantly reduced the timesteps for finding the rat. Figure 3 shows that baseline bot throughout at alphas, except the initial alphas. The mean of improved across almost all aphas including the standard deviation. We notice that at the end around alph 0.3, the deviation tends to split a lot due to the probability of hearing a ping being low and increasing the timesteps for both.

Let's break this down further as to why by looking at sensing steps and ping steps. Figure 4 shows that our improved bot across all alpha values has a higher success of pinging using the space rat versus the baseline bot. This is because utilize a utility function to estimate our manhattan distance between where we

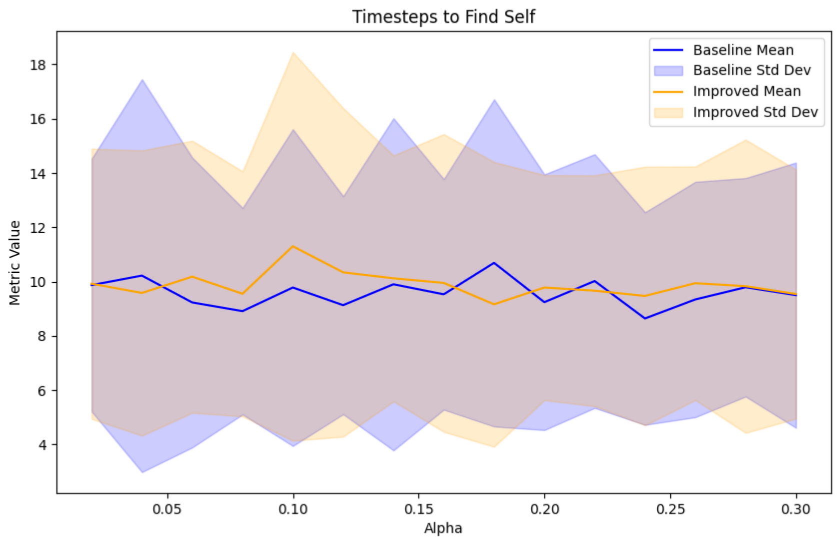


Figure 1: Timesteps to find self for Phase 1.

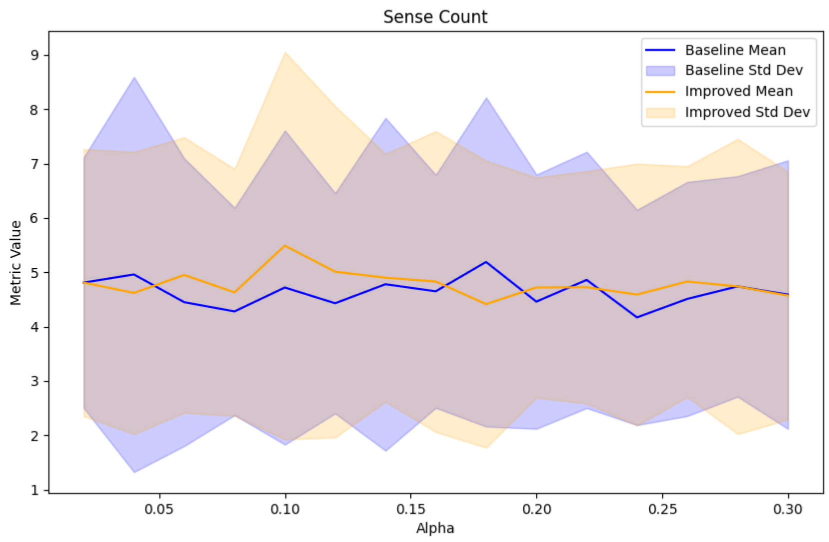


Figure 2: Timesteps for sensing closed neighbors.

are and the target cell to fix our approach based on the probabilites of hearing a ping. Thus, instead of just heading to the maximized cell, we update our utility to move to increase our chance to hear a ping and find the rat faster instead of moving all the way to the highest cell. Thus, across all alpha values, our improved bot hears more pings and tries for ping steps.

For moving in 5, we notice that the baseline bot tends to move more than our improved bot. We believe that the reason for this increased movement is because the baseline bot drastically changes it's location before pings if it does not hear a ping at a certain location to try and test as many areas as it can. Across all alpha values, our improved bot moves less with a condensed standard deviation whereas the baseline bot has a large standard deviation with a positive increase in move steps as alpha increases. Because our improved bot maximizes the chance of hearing a ping, it knows where it is going, varying less and needing

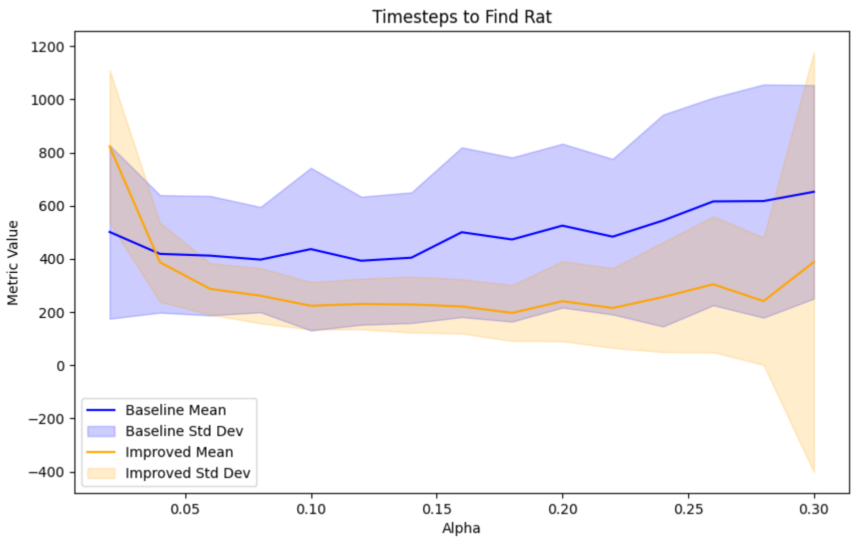


Figure 3: Timesteps to find self for Phase 1.

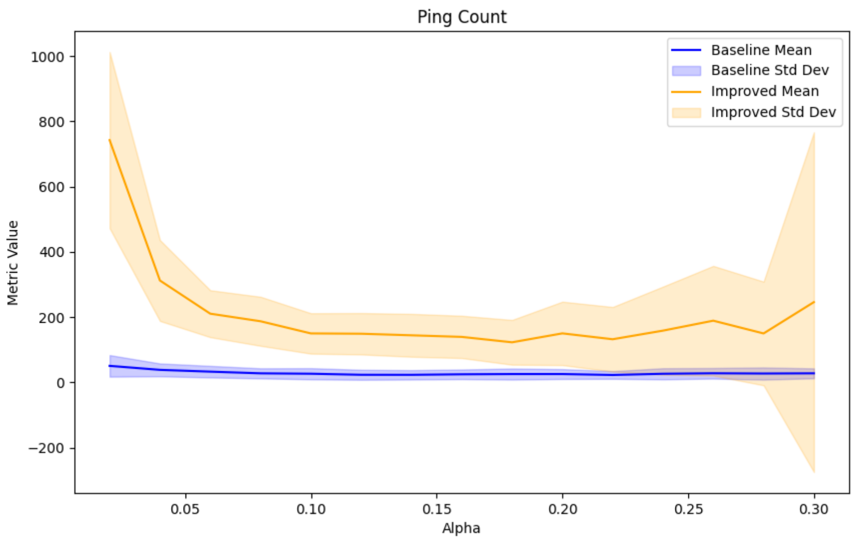


Figure 4: Timesteps when bot pinged using space-rat detector

to move less. We conducted a t-test to see if our improved bot improvement was statistically significant over the baseline bot. Which you can see in figure 6. The t-test ended up giving us a p-value of 0.000264 which shows that our bot was a significant improvement over that of the baseline bot for the stationary rat.

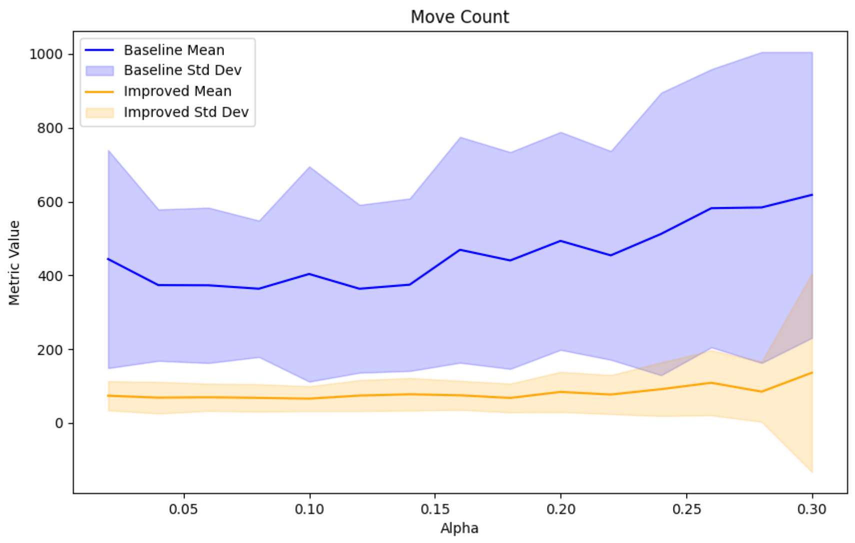
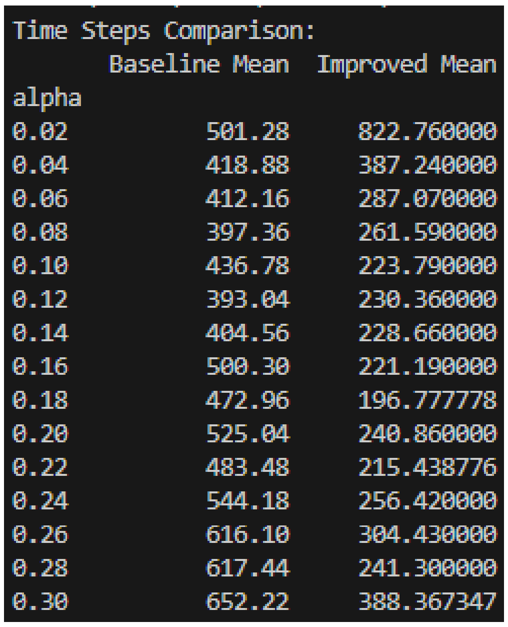
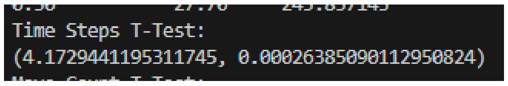


Figure 5: Timesteps when bot moved



```
Time Steps Comparison:
       Baseline Mean  Improved Mean
alpha
0.02          501.28     822.760000
0.04          418.88     387.240000
0.06          412.16     287.070000
0.08          397.36     261.590000
0.10          436.78     223.790000
0.12          393.04     230.360000
0.14          404.56     228.660000
0.16          500.30     221.190000
0.18          472.96     196.777778
0.20          525.04     240.860000
0.22          483.48     215.438776
0.24          544.18     256.420000
0.26          616.10     304.430000
0.28          617.44     241.300000
0.30          652.22     388.367347
```



```
Time Steps T-Test:
(4.1729441195311745, 0.00026385090112950824)
```

Figure 6: Stationary Stats Comparison

Question 4: Updating the Knowledge Base for a Moving Rat

Our original improved bot's performance drastically decreased when it came to finding the moving rat. From our analysis we found that the baseline bot, when compared to itself, for a moving rat required on average 283 time steps more when comparing the mean value across alpha values of 0.02-0.3. Please refer to the Figure 7 to see the difference in timesteps between the moving and stationary rat for the baseline bot.

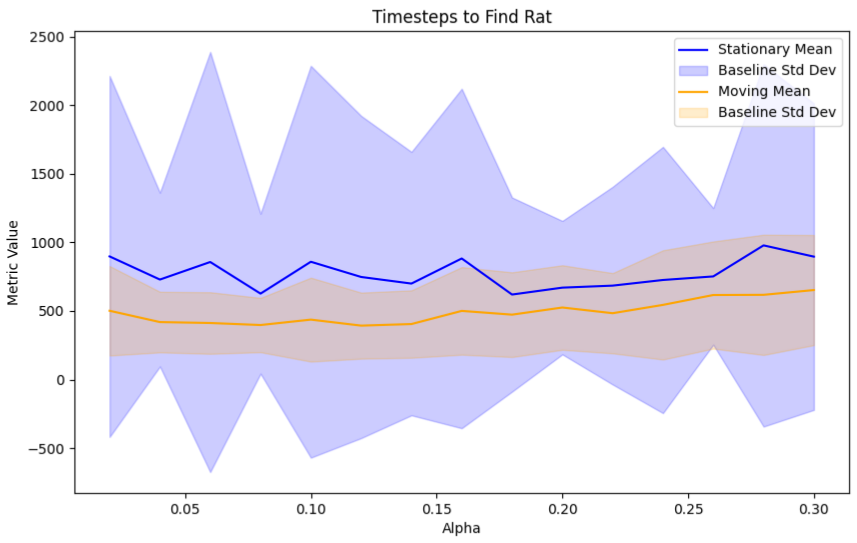


Figure 7: Average Timesteps between Moving and Stationary Rat for Baseline Bot

When comparing the improved bot among alpha values of 0.02-0.3 between stationary and moving values, we found that it required over 4141 timesteps more to find the bot. This drastic difference was caused due to lower alpha values (< 0.1) causing the bot to take upwards to 150000+ timesteps (for our simulation data we limited it to be 20000) to find the rat. If comparing only alpha values from 0.15-0.30 we get that on average, it required 1039 more timesteps to find the bot. Please refer to Figure 8 to see the difference in average timesteps.

This led us to make another version of our improved bot that could better handle a moving bot. The main alteration we had was the way we updated probabilites in our knowledge base through the use of a transition model.

Knowledge Base Updates with a Moving Rat

When the rat moves randomly, the knowledge base update accounts for the rat's movement. The probability of the rat being in a cell j at timestep $t+1$ depends on the probabilities of the rat being in the neighboring cells at timestep t.

Implementing a Transition Model

The transition model greatly improved the bot's ability to find the moving rat. It works by updating the probabilities of where the rat might be, based on its random movements to neighboring cells. This keeps the bot's information accurate and up-to-date. By predicting where the rat could be next, the bot can better track and catch the rat. This method reduces uncertainty, helping the bot make smarter moves. As a result, the bot spends less time pinging and more time moving towards likely rat locations, making the search faster and more efficient. Here is how it worked:
For each cell j:

$$P(R_j^{t+1}) = \sum_k P(R_k^t) \cdot P(R_j|R_k)$$

Where $P(R_j|R_k)$ is the probability of the rat moving from cell k to cell j. We have $n_k + 1$ because there is a chance of the rat staying in the same cell.

Since the rat moves randomly to neighboring cells:

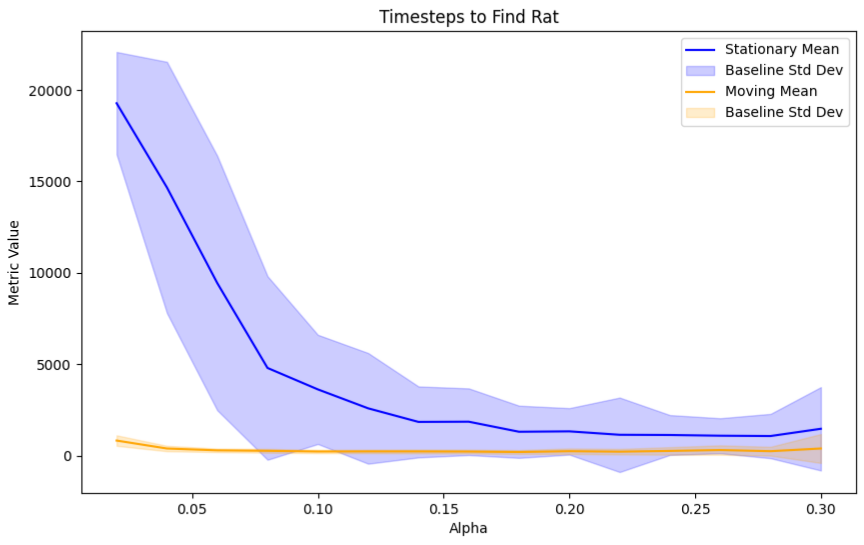


Figure 8: Average Timesteps between Moving and Stationary Rat for Improved Bot

If j is a neighbor of k:

$$P(R_j|R_k) = \frac{1}{n_k + 1}$$

If $j = k$:

$$P(R_j|R_k) = \frac{1}{n_k + 1}$$

Otherwise:

$$P(R_j|R_k) = 0$$

Where n_k is the number of open neighboring cells of cell k.

Update Formula

The updated probability of the rat being in cell j at timestep $t+1$ is:

$$P(R_j^{t+1}) = \sum_{k \in N(j)} \frac{P(R_k^t)}{|N(k)|}$$

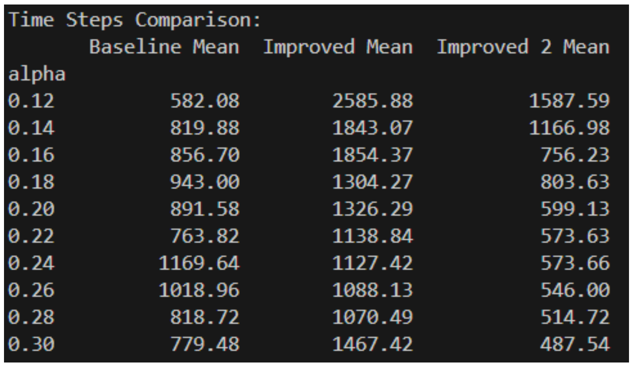
Where $N(k)$ is the number of open neighboring cells of cell k.

Simulation and Comparison

After making our changes to the second improved bot we decided to rerun our simulations. We ran 100 simulations for each alpha value from 0.02 to 0.30 at 0.02 steps. Figure 10 shows the results for each alpha value. The improved bot 2 with the transition model performed significantly better than that of the initial improved bot and even went below the average number of timesteps for alpha values between 0.16 and 0.30 when compared to the baseline bot. In fact, when we conducted a t-test to see if there was a significant improvement between the baseline bot and improved bot across and we got a p-value of 0.4 which indicated no significant improvement. However when we conducted the t-test again for alpha values between 0.16 and

0.30 we got a p-value of 0.0003029878818659705 indicating a significant improvement in number of timesteps for alpha values in the range [0.16, 0.30].

Conducting a statistical analysis between the results gave us what is shown in figure 9



```
Time Steps Comparison:
       Baseline Mean  Improved Mean  Improved 2 Mean
alpha
0.12          582.08        2585.88          1587.59
0.14          819.88        1843.07          1166.98
0.16          856.70        1854.37           756.23
0.18          943.00        1304.27           803.63
0.20          891.58        1326.29           599.13
0.22          763.82        1138.84           573.63
0.24         1169.64        1127.42           573.66
0.26         1018.96        1088.13           546.00
0.28          818.72        1070.49           514.72
0.30          779.48        1467.42           487.54
```

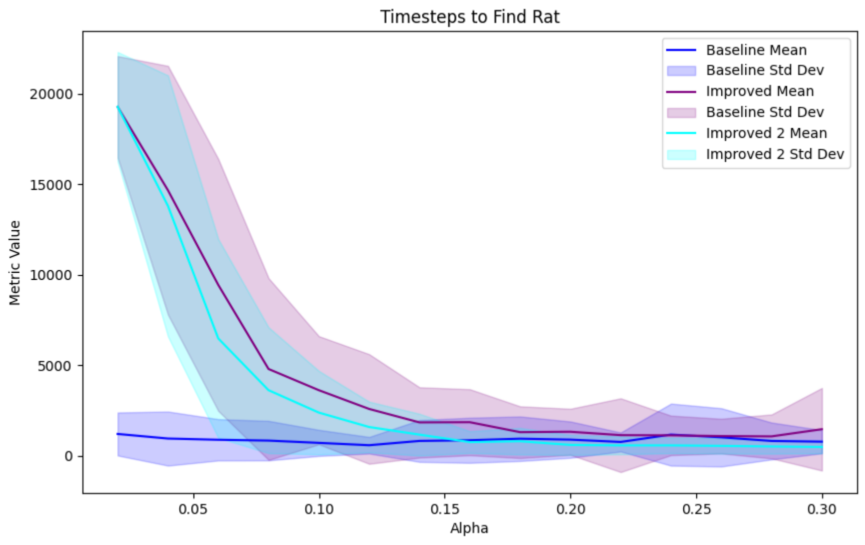
Figure 9: Statistical Analysis of Timesteps between Bots

Improving the Bot Design

- Multi-Step Prediction:
  - Extend the transition model to predict the rat's position over multiple future timesteps rather than just the next step. This could involve simulating several possible future paths for the rat and updating the knowledge base accordingly.
  - By considering multiple future steps, the bot can better anticipate the rat's movements and plan its actions more effectively, increasing the likelihood of intercepting the rat.
- Reinforcement Learning
  - A technique that I am not completely familiar with but has come up in my own research. Implementing reinforcement learning techniques to train the bot to optimize its actions based on rewards would allow the bot to learn from its experiences and improve its decision-making over time, potentially leading to more efficient strategies for finding the rat.
- Dynamic Movement Strategy
  - We could implement a strategy that adjusts how much the bot moves based on its recent successes and failures. If it fails to hear a ping for a long time it could try and move around for multiple steps to explore vs if it does hear a ping it will take small more calculated steps
  - This could also entail using a different traversal method rather than BFS, perhaps using probabilistic path algorithms that would allow the bot to explore more efficiently.

Conclusion

In this project, we tackled the challenge of designing a bot to navigate a spaceship, locate itself, and catch a space rat. We developed three versions of the bot: a baseline bot, an improved bot, and a second improved bot with a transition model to handle a moving rat.



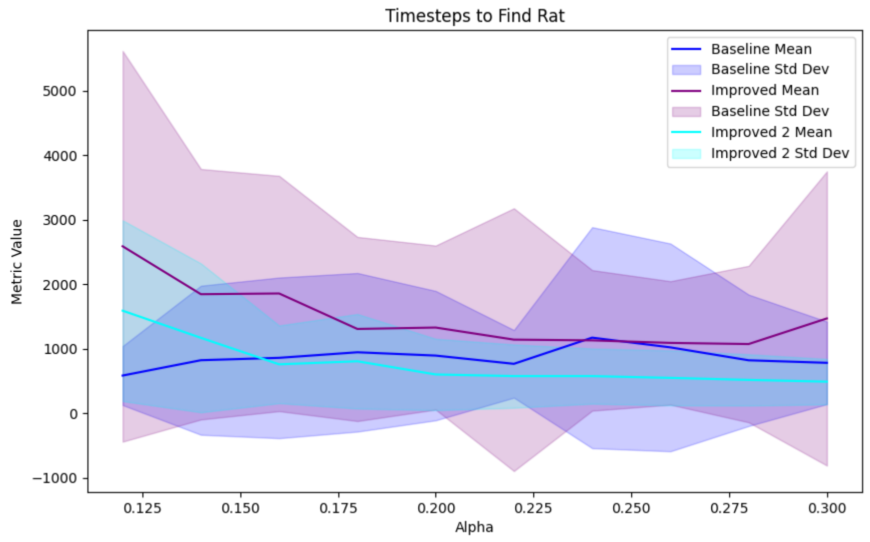


Figure 10: Comparison of mean timesteps to find rat for each bot.

Bots:

Baseline Bot: The baseline bot operated in two phases. In Phase 1, it identified its location by sensing the number of blocked neighboring cells and moving in the most common open direction. In Phase 2, it tracked the space rat by alternating between using the space rat detector and moving towards the cell with the highest probability of containing the rat. The knowledge base was updated using Bayes' theorem based on whether a ping was heard or not.

Improved Bot: The first improved bot enhanced Phase 2 by incorporating utility functions to decide when to ping and when to move. It calculated the entropy of the knowledge base to determine the uncertainty and made decisions accordingly. This approach significantly reduced the timesteps required to find the rat compared to the baseline bot, especially for stationary rats.

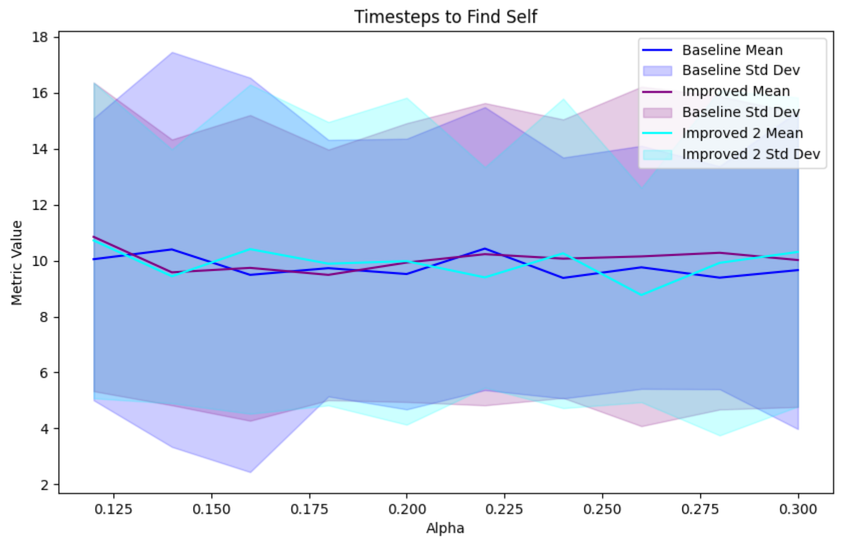


Figure 11: Time for Phase 1 (finding self) for each bot

Second Improved Bot with Transition Model: To address the challenge of a moving rat, we introduced a transition model in the second improved bot. This model accounted for the rat's random movements by redistributing the probabilities across neighboring cells, reflecting the rat's potential new positions. By predicting the rat's future locations and incorporating these predictions into the decision-making process, the bot could more effectively track and intercept the rat. This approach reduced the uncertainty in the bot's knowledge base, allowing it to make more informed and strategic movements. Consequently, the bot spent less time pinging and more time moving towards high-probability areas, leading to a more efficient search and quicker detection of the moving rat.

Achievements:

Bayesian Networks: We utilized Bayesian networks to update the knowledge base based on the bot's observations, ensuring accurate probability distributions.
Transition Model: The implementation of the transition model significantly improved the bot's performance in finding the moving rat by accounting for the rat's random movements.
Utility Functions: The use of utility functions allowed the bot to balance between pinging and moving, optimizing its actions to reduce the time required to find the rat.
Statistical Analysis: We conducted extensive simulations and statistical analyses to compare the performance of the different bots. The results demonstrated that the second improved bot with the transition model performed significantly better than the initial improved bot and the baseline bot, especially for higher alpha values.
Future Improvements: Given more time, we could further enhance the bot's performance by implementing adaptive pinging strategies, multi-step prediction models, reinforcement learning techniques, dynamic movement strategies, and probabilistic path planning algorithms. These improvements would allow the bot to learn from its experiences, anticipate the rat's movements more accurately, and explore the environment more efficiently.

Overall, this project demonstrated the effectiveness of combining Bayesian networks, transition models, and utility functions to design a bot capable of efficiently locating and catching a moving space rat. The iterative development and rigorous testing of the different bot versions highlighted the importance of adaptive strategies and probabilistic reasoning in dynamic environments.