Space Rats Lab Report

Jeeva Ramasamy
Rutgers University
jeeva.ramasamy@rutgers.edu
Course: 16:198:520
Instructor: Professor Cowan

Abstract

In this project, we explore the design and evaluation of bot strategies to address the new challenges onboard the deep-space vessel *Archaeopteryx*. Following a blast of cosmic radiation, the bot has lost knowledge of its exact location within the ship, though it retains access to a map of the ship. Simultaneously, a space rat has infiltrated the vessel, requiring the bot's attention for its capture. To locate both itself and the target, the bot has access to three actions at each timestep: it can sense the number of blocked neighboring cells, use a space rat detector that pings with a probability proportional to the rat's distance, or attempt movement in a cardinal direction, learning when it encounters walls. The ship's layout is represented as a 30 x 30 square grid with blocked outer edge cells, along with the same interior structure used in the Fire Extinguisher Project. Two bot strategies are implemented, each leveraging a utility-based approach to handle uncertain localization and tracking. Bot 1 employs a static utility approach, adjusting its position estimates based on immediate sensor data but relying on consistent movement patterns to explore the grid. Bot 2 adapts dynamically, evaluating expected utility as a combination of both immediate and future rewards. Performance is assessed through simulations at varying detector sensitivities (α) to gauge effectiveness of the bots. Results indicate that Bot 2 outperforms Bot 1 in most cases by capturing the space rat in less turns.

1 Introduction

The bot operates in a simulated 30 x 30 grid environment, designed to replicate the complex layout of the deep space vessel *Archaeopteryx*. This grid is dynamically generated with a mix of open paths and blocked cells, creating a complex navigation challenge. At the start of each simulation, the bot and space rat are placed in random cells. Initially, it is assumed that the space rat is stationary and thus, any cell visited cannot contain the space rat at a later time. This project also explores the challenges and changes that arise when the movement of the space rat is taken into consideration.

2 Methodology

This section outlines the processes for creating the simulated environment, defining bot behaviors, and evaluating their performance in navigating toward the space rat.

2.1 Environment Simulation

The first step in setting up the environment involves generating a 30 x 30 grid to represent the ship's layout. The process for populating this grid with blocked and open cells is designed to create a navigable maze-like environment for the bots, while maintaining some randomness and flexibility in the ship's layout. The steps are as follows:

- **Grid Initialization**: The grid begins as a completely blocked space, where all cells are marked as closed. A random starting cell within the inner 28 x 28 grid is selected and opened to initiate the process of creating pathways. Note that none of the cells in the outer layer can be opened during this full process.
- **Opening Cells**: The 28 x 28 grid is progressively populated with open cells. Each new open cell is chosen based on having exactly one neighboring cell that is already open. This constraint ensures that the paths formed are narrow and interconnected, resulting in a maze-like structure. The process continues until no additional cells can be opened using this rule, preventing the creation of isolated open cells and ensuring connectivity.
- **Opening Some Dead Ends**: Approximately half of the dead-end cells (cells with only one open neighbor) are randomly opened once the initial maze is generated. This step enhances navigability by reducing bottlenecks and providing more potential routes, preventing overly restrictive pathways that could trap bots.
- **Bot and Space Rat Placement**: After the grid layout is established, distinct open cells are randomly selected for the bot's starting position and the space rat location. Since all pathways are connected, the bot will always be able to reach the space rat.

2.2 Bot Capabilities

Each bot is allowed to perform one of three moves at each timestep:

- **Blocked Cell Detection**: The bot can sense how many of the neighboring eight cells are currently blocked. This does not provide any information about the specific cells which are blocked.
- **Space Rat Detector**: The bot can use its space rat detector which will ping based on how close the space rat is. The space rat detector will ping definitively if the bot is in the same cell as the space rat. If the bot is in cell i and the space rat is in cell j, then the probability of hearing a ping is given by $e^{-\alpha(d(i,j)-1)}$, where $d(i, j)$ is the manhattan distance between cell i and cell j, and $\alpha > 0$ is a constant specifying the sensitivity of the detector.

- **Attempt Move**: The bot can try to move in a cardinal direction. The bot is aware if it hits a wall, in which case it will remain in place. Due to the nature of the grid generation, it is guaranteed that at least one of the four cardinal directions will always be open.

2.3 Bot Design and Behavior

The bots are designed to navigate the grid and respond to the space rat based on the following strategies:

- **Bot 1**: This bot uses a two-phase strategy to locate the space rat.
    - **Phase 1**: Bot 1 starts at an unknown position and begins by sensing the number of blocked cells around it to narrow down possible locations. Using this information, it systematically explores the most common open direction and updates the list of potential positions. For every move, Bot 1 tracks its new position and updates a set of explored and closed cells to ensure it does not move around in circles or try to move to a closed cell from a different side. This process continues until Bot 1 narrows down its possible location to a single cell, allowing it to locate itself on the map.
    - **Phase 2**: Bot 1 switches focus to tracking down the space rat. It uses a detector that provides probabilistic pings based on proximity to the rat, using this feedback to adjust its utility map for high-likelihood target cells. The bot performs a breadth-first search (BFS) from its current location, prioritizing moves toward cells with higher probabilities of containing the rat as tracked by the utilities in its knowledge base.
- **Bot 2**: This bot also uses a two-phase strategy, but uses the space rat detector in Phase 1 to start locating the rat while figuring out its own position on the map. Bot 2 balances exploration with exploitation using a utility based pathfinding algorithm along with discounts for previously explored cells.
    - **Phase 1**: Bot 2 performs the same actions as Bot 1's Phase 1 to locate its position on the grid. In addition, Bot 2 uses the rat detector at every new position to keep track of ping responses and utilities. Since Bot 2 cannot access the game's grid before finding its own location, it generates a dynamic grid relative to its starting position for its movements and utility tracking. The starting cell is (0, 0), and each movement is relative to this coordinate. At the end of Phase 1 when Bot 2 has identified its location on the map, it converts all tracked relative positions to the actual locations on the grid and transfers its data to the utility grid.
    - **Phase 2**: In the second phase, Bot 2 uses its space rat detector each time it visits an unexplored cell, recording pings and adjusting utilities for neighboring cells accordingly. In order to assess the best move, Bot 2 takes into consideration both the immediate reward of the move as well as the expected future reward. The calculation for cell utility is as follows:

$cell\ utility = multiplier * (immediate\ reward + total\ future\ reward)$
where the **immediate reward** is the utility of the cell in the direction taken and
total future reward = discount * Σ utilities of depth 1 from move
+ discount^2 * Σ utilities of depth 2 from move
+ discount^3 * Σ utilities of depth 3 from move
+ discount^4 * Σ utilities of depth 4 from move
After extensive testing, the best **discount** factor was determined to be **0.6**, which allows Bot 2 to give attention to nearby utilities without overly favoring distant moves. To ensure that the algorithm does not take too much time, the **depth** has been limited to **4**, which corresponds to a discount of ~ 0.13 at the deepest layer of cells visited. **Multiplier** adjusts the priority based on whether a cell has been visited before. For cells that have already been visited, the multiplier is **0.3**, reducing their attractiveness; for new cells, it remains **1**, making them more appealing for exploration. By dynamically calculating this total utility score for each possible move, Bot 2 can select the path that maximizes its likelihood of success in tracking the space rat.

- **Backtracking Plan**: It is possible for Bot 2 to go down a path with a dead end and get stuck since all utilities of depth 4 are zero due to prior exploration. In such cases, the bot will use Breadth-First Search to the current highest utility cell until a positive utility is detected. Once the bot detects a viable path with positive utility, it resumes its regular pathfinding algorithm.

2.4 Knowledge Base

The knowledge base for the bots is implemented as described below:

- **Knowledge Base Implementation for both bots**: The knowledge base is represented as a utility grid where each cell's value reflects the probability that the space rat might be located there. All open cells have an initial value of 1 while closed cells have 0. At every new position visited by the bot, the utility is set to 0 unless the space rat is found. When there is a space rat detection, the cells immediately located in the four cardinal directions have their utilities increased by 1. When there is no detection, the four cells will have their utilities set to 0. This is because the space rat detector is guaranteed to ping at a Manhattan Distance of 1, regardless of α. Any cell previously set to 0 will never have its utility increase at any step. This helps the bots quickly track down the space rat if the space rat is not known to move.
- **Bot 2 Special Behavior**: Bot 2 is more likely to explore the surrounding region of a ping due to the calculation of expected future rewards. This means that it would underperform when compared to Bot 1 in cases where there are too many pings far away from the space rat. In order to account for this issue, Bot 2 will not use the utilities for tracking if over 90% of ping attempts are successful after at least 10 turns. This adjustment helps Bot 2

avoid being misled by an excessive number of false-positive pings, allowing it to prioritize more promising locations and improve its efficiency in locating the target.

- **Knowledge Base Update**: Given N open cells and manhattan distance D from space rat,
  Prior: $P(space\ rat\ at\ cell\ C) = \frac{1}{N}$
  Posterior: $P(space\ rat\ at\ cell\ C \mid ping) = \frac{P(ping \mid space\ rat\ at\ cell\ C) * P(space\ rat\ at\ cell\ C)}{P(ping)}$
  $= \frac{e^{-\alpha(D-1)} * \frac{1}{N}}{\Sigma\, P(ping \mid space\ rat\ at\ cell\ C')\ P(space\ rat\ at\ cell\ C')}$

  The posterior probability changes after each time the space rat detector is used, due to confirming that the space rat cannot be in the current bot position, given that the game does not end. Though the posterior probability can be calculated as shown above, it cannot be applied to the algorithm since it requires knowledge of the space rat detector sensitivity parameter α. Hence, Bot 2 uses a similar update mechanism to Bot 1, but accounts for cases in which there are many pings far from the space rat's location.

2.5 Performance Evaluation

To assess the effectiveness of each bot, the following steps are undertaken:

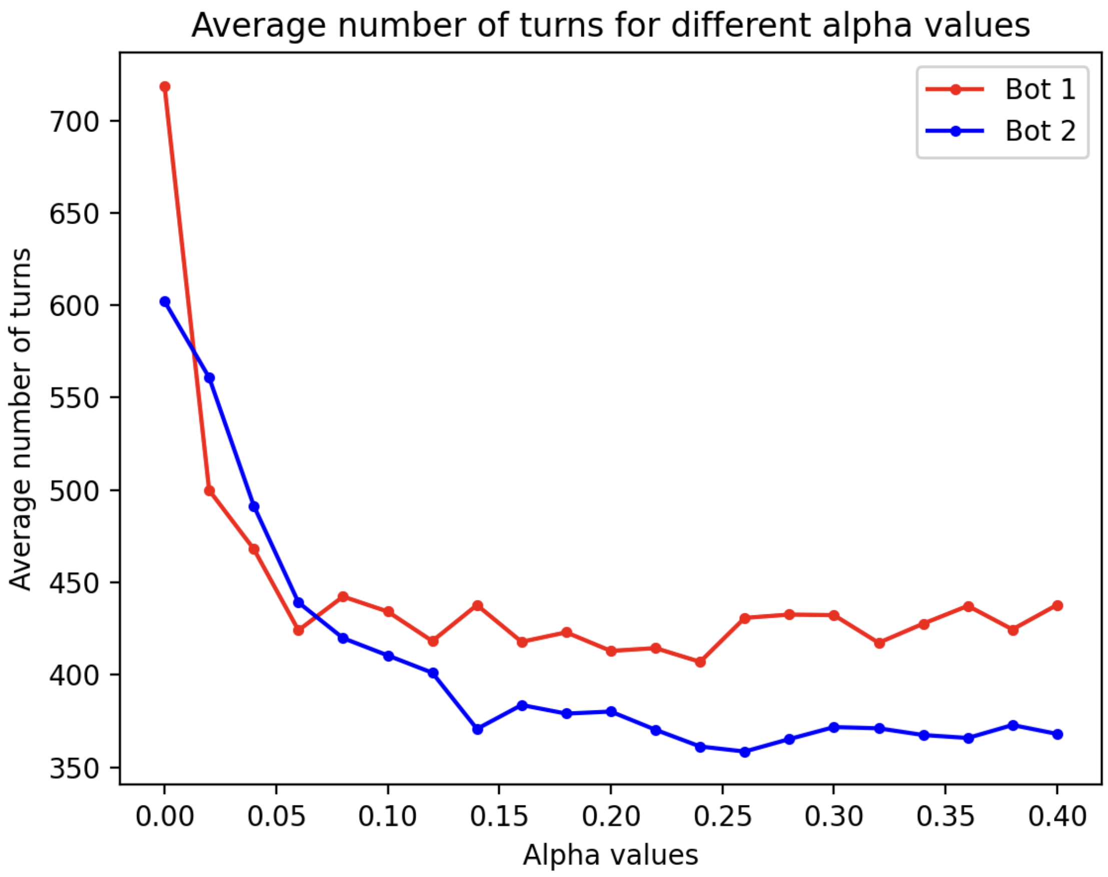
- **Experimental Design and Repetition**: Each bot is tested in numerous environments with varying values of the space rat detector sensitivity parameter α, ranging from 0 to 0.4 in increments of 0.02. For each value of α, 400 independent simulations were conducted to ensure robust statistical evaluation. The random generation of grid layouts, bot positions, and space rat positions allowed the bots to be tested for adaptability to diverse conditions.
- **Success Metrics**: The primary measure of success is the number of turns taken by the bot to reach the space rat. The bot's ability to locate itself on the map and reach the goal was monitored, and the number of turns taken was recorded for each bot in every simulation. There are no cases where success is not possible for the bot, though there are simulations in which the bot takes an excessive number of steps before success.
- **Visualization of Results**: The number of turns taken by each bot across different values of α were plotted for a visual comparison. See Section 3.1 for a visual comparison of the bots' performances across different α values.

3 Results

This section presents the findings from implementing and evaluating the two bots across various simulated environments on the ship.

3.1 Bot Performance Across Varying α Values

The bots were evaluated in numerous simulations while varying the space rat detector sensitivity parameter α from 0 to 0.4 in increments of 0.02. The average number of turns taken for each bot were recorded and graphed to visualize the trends. In order to ensure a fair comparison, both bots were tested on the same grid configurations. Furthermore, consistent grid generation across varying α values was achieved by setting the random number generator's seed to '520,' eliminating discrepancies in starting positions.



The figure above shows the average number of turns taken by each bot across various α values (space rat is stationary)

For each α value, 400 simulations were conducted to evaluate the performance and effectiveness of the bots accurately. There is a clear decrease in the average number of turns taken by each bot until around α value 0.2. Both bots follow a similar exponential decay trend, but Bot 1 has a much greater decrease at α value 0.02. Increasing α beyond 0.4 does not decrease the average number of turns further.

3.2 Comparative Analysis of Bot Strategies

A comparative analysis reveals distinct patterns in the bots' performances, especially regarding their responses to varying numbers of space rat detections:

- **Bot 1** took fewer turns than Bot 2 to find its position on the map in most cases. For low α values, except for 0, Bot 1 also finds the space rat slightly faster.
    - In layouts where the bot reaches the space rat by chance while locating itself, Bot 2 ends the game early. On the other hand, Bot 1 continues until Phase 1 is over to start looking for the space rat.
- **Bot 2** outperformed Bot 1 at nearly all α values, demonstrating that its unique strategy effectively combined pathfinding with real-time adaptations to the space rat detections. However, at lower α values, Bot 2 performs slightly worse due to its focus on utilizing space rat detections to track the location of the target.

At $\alpha = 0$, the bots will receive a successful ping at every detector use, regardless of distance. Conversely, at α values higher than 0.2, the number of successful pings drastically decreases. This means that the bots will not gain much information about the space rat. Although the space rat detector is mostly useless in such cases, it is still required to confirm whether the bot reached the space rat's location.

4 Discussion

The results of this comparative analysis highlight the complex interplay between bot strategies and the varying degrees of space rat detector sensitivity parameter α.

4.1 Adaptability and Strategy

Bot 1, which relied on static utility and BFS showed great performance at lower α values but faltered as information was decreased. This highlights an important limitation: a lack of foresight due to only taking the immediate reward into consideration. In contrast, Bot 2, which altered its path at every timestep based on expected utility, demonstrated lower average turns at most α values.

4.2 Tracking a Moving Space Rat

Previously, the space rat was assumed to be stationary. This greatly simplified the process of finding the space rat for both bots. However, it is not very likely that an intruding space rat will remain in one place after entering the space vessel. Therefore, the bot's pathfinding algorithm needs to take into consideration that the space rat continues to move at every timestep. In this section, the space rat is assumed to move in a random open adjacent cell after each move taken by the bot. To accommodate for this change in space rat behavior, the handling of cell utilities needed to be modified. Even if a cell is visited and has a utility of 0, it can later increase based on space rat detections. This is because it is possible for the space rat to have moved to a location previously visited by the bot.

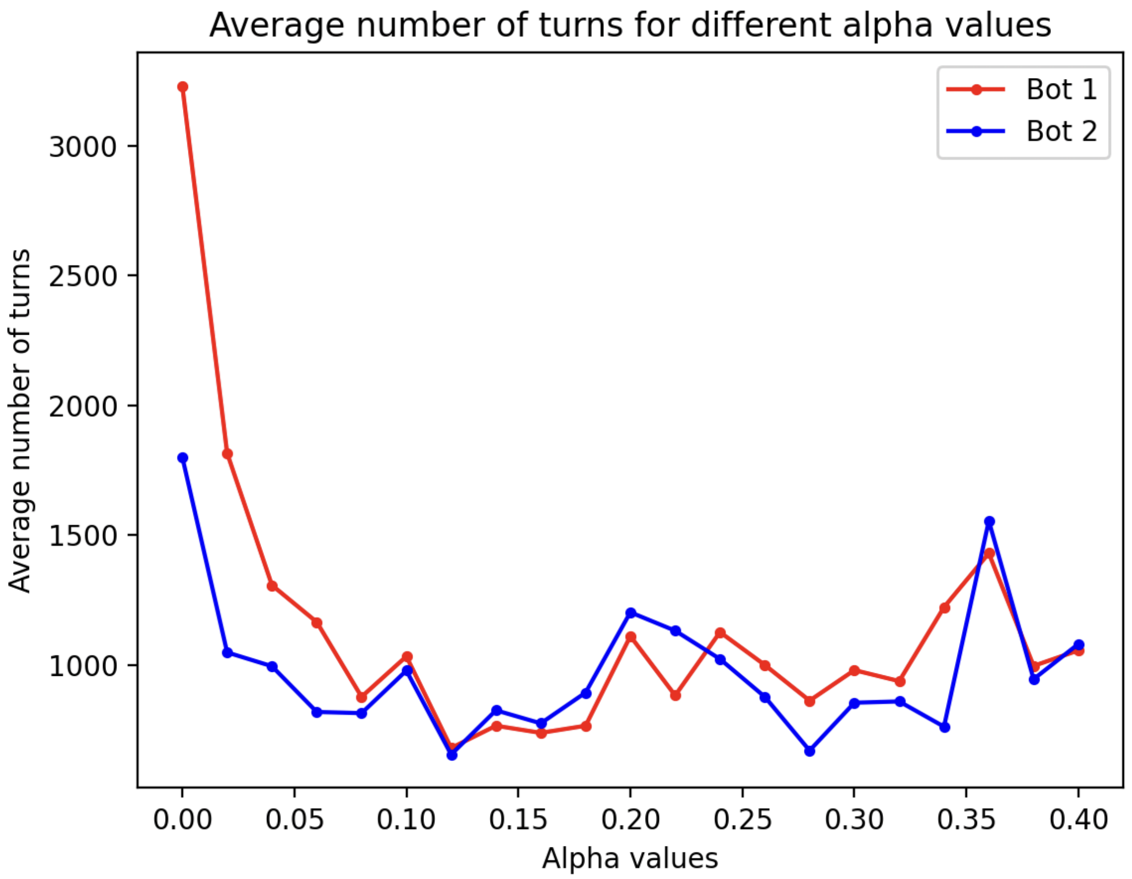
Knowledge Base Update: Given N open cells and manhattan distance D from space rat,

Prior: $P(space\ rat\ at\ cell\ C) = \frac{1}{N}$

Posterior: $P(space\ rat\ at\ cell\ C \mid ping) = \frac{P(ping \mid space\ rat\ at\ cell\ C) * P(space\ rat\ at\ cell\ C)}{P(ping)}$

$$= \frac{e^{-\alpha(D-1)} * \frac{1}{N}}{\Sigma\, P(ping \mid space\ rat\ at\ cell\ C')\, P(space\ rat\ at\ cell\ C')}$$

The knowledge base has the same prior and posterior probabilities, regardless of whether the space rat is stationary or mobile. The only difference is that this probability remains constant throughout the entire game, since knowledge of the space rat's absence in a cell does not indicate future absence.



The figure above shows the average number of turns taken by each bot across various α values (space rat is not stationary)

The bots were evaluated in numerous simulations while varying the space rat detector sensitivity parameter α from 0 to 0.4 in increments of 0.02. The bots were evaluated under the same conditions described in Section 3.1. For each α value, 100 simulations were conducted to evaluate the performance and effectiveness of the bots accurately. There is a clear decrease in the average number of turns taken by each bot until around α value 0.18, after which it starts to increase for both bots. Both bots take approximately 500 more turns to catch the space rat on average. However, Bot 2 shows slightly better performance at most α values and is also more stable, which suggests that Bot 2 is more adept at adapting to varying levels of detector sensitivity.

5 Conclusion

The exploration of bot performance aboard the deep-space vessel *Archaeopteryx* has provided valuable insights into the complexities of tracking and locating a target, mobile or stationary. The comparative analysis of Bot 1 and Bot 2 demonstrated distinct performance patterns, with each bot employing different strategies to address the challenge of detecting the space rat. Bot 1 focused on immediate utility updates, while Bot 2's usage of future expected reward allowed it to explore better paths. The results revealed that the success of the bots was significantly influenced by their ability to account for the variable pings from the space rat detector. Bots demonstrated varied levels of adaptability, with Bot 2 slightly outperforming Bot 1 in most scenarios.