Project 2
The Power of Knowledge

Submitted by
Will Chandler (wcc4)
Vignesh Krishnan (vk505)
Rohan Tripathi (rt767)

Introduction

In this project, we implement bot designs that make more informed decisions based on the information they have available from the environment. As provided in the project description, the information is as follows:

- A crewmate detection sensor which uses distance from crewmate as a probabilistic measure of likelihood of receiving a beep
- An alien detection sensor which beeps when an alien enters a designated perimeter of the bot
- The movement related restrictions involving aliens and the constraints on the bot.

Based on this knowledge base, we create appropriate data structures to process the knowledge in an organized manner and enable our bots to make logical decisions while approaching the crewmate.

Structure of Existing knowledge

The knowledge base for our project is maintained through the following data structures:

- hashmaps to store crewmate and alien probabilities
- A hashmap for storing shortest paths from every open cell in the ship to every other open cell (calculated using dijkstra)

General Bots Overview

Bots 1, 3, 4, 6, and 7 all follow the same logic and strategy, only differing on which probability functions are used and what scenario they are simulated in. Each of these bots will find the shortest path toward the cell with the current highest probability of containing a crewmate, using values of distances calculated using dijkstra's algorithm. These bots will only move if the next cell in their path has a probability of containing an alien below a near zero threshold, i.e. the next cell in their path has almost no chance of harboring an alien. If this is not the case, they will remain still until the alien moves outside the bots alien detection zone.

Bots 2, 5 and 8 on the other hand, do not simply wait out the threat from aliens. Each of these bots uses three unique strategies, to increase the information gathered on the crewmate location, decrease the time steps needed to rescue the crewmates and decrease the likelihood of being caught by the alien.

Probability Modeling and Updation

$$P(\text{Crewmate in cell j}) = \begin{cases} \begin{cases} \frac{P(\text{crew in cell } j) * e^{-\alpha(d_j - 1)}}{\sum_k^{\text{all cells}} P(\text{crew in cell } k) * e^{-\alpha(d_k - 1)}} \\ \frac{P(\text{crew in cell } j) * (1 - e^{-\alpha(d_j - 1)})}{\sum_k^{\text{all cells}} P(\text{crew in cell } k) * (1 - e^{-\alpha(d_k - 1)})} \end{cases} & \begin{array}{l} \text{if crew-beep is true} \\ \text{if crew-beep is false} \end{array} \end{cases}$$

First the probability of getting a beep is calculated using the distance from the bots current cell to every open cell. Then if a beep was received this value is multiplied against the prior probability of each cell containing an alien then all cell probabilities are normalized. If no beep is received then we take 1 - the probability of receiving a beep, i.e. the probability of not receiving a beep and multiply this against all prior probabilities and again normalize.

$$P(\text{Alien in cell j}) = \begin{cases} \begin{cases} 0 & \text{if cell outside bot detector zone} \\ \frac{P(\text{alien in cell } j)}{\sum_{i=0}^{\text{all cells}} P(\text{alien in cell } j_i)} & \text{if cell inside bot detector zone} \end{cases} & \text{if alien-beep is true} \\ \begin{cases} \frac{P(\text{alien in cell } j)}{\sum_{i=0}^{\text{all cells}} P(\text{alien in cell } j_i)} & \text{if cell outside bot detector zone} \\ 0 & \text{if cell inside bot detector zone} \end{cases} & \text{if alien-beep is false} \end{cases}$$

First values are set based on the location of the alien, either within the detection square or outside the detection square. If the alien is within the detection square the probability of containing an alien is set to 0 for all cells outside the detection square. Then the cells inside the detection square are normalized.

$$P(Alien\ in\ cell\ j)_{t+1} = \beta_t \sum_{i=0}^{all\ neighbor\ cells\ j} P(Alien\ in\ cell\ i)_t * P(Alien\ in\ cell\ j | Alien\ in\ cell\ i)_{t+1}$$

where all cells i are neighbors of j

$$\beta_t = \frac{1}{\sum_k^{all\ cells} \sum_{i=0}^{all\ neighbor\ cells\ k} P(Alien\ in\ cell\ i)_t * P(Alien\ in\ cell\ k | Alien\ in\ cell\ i)_{t+1}}$$

where all cells i are neighbors of k

The probability an alien is in a cell after the alien has moved is calculated by taking the probability that the alien is in a neighbor cell multiplied by the probability it moves out of that neighbor cell. These values are then summed for each neighbor cell, normalized and set as the new probability of the initial cell containing an alien.

$$P(\text{crew in cell j} \wedge \text{crew in cell k}) = \frac{P(\text{crew in cell j} \wedge \text{crew in cell k}) \cdot P(\text{Beep}|\text{crew in cell j} \wedge \text{crew in cell k})}{\sum P(\text{crew in cell j} \wedge \text{crew in cell k}) \cdot P(\text{Beep}|\text{crew in cell j} \wedge \text{crew in cell k})}$$

$$P(\text{crew in cell j}) = \sum_k P(\text{crew in cell j} \wedge \text{crew in cell k})$$

Instead of calculating the probability a crew member is in a cell we calculate the probability there are crew members in a pair of cells. First a hashmap of every possible combination of two cells is created, excluding cases where the two cells are equal. Then these cell pairs are given an initial uniform probability. Whenever a beep is received these cell pair probabilities are updated by multiplying the cell pair probability by the probability of getting a beep from those two cells. This is calculated by taking 1 minus the probability of not receiving a beep. When a beep is not received these cell pair probabilities are updated by multiplying them times the probability of not receiving a beep from both cells in the cell pair. These updated cell pair probabilities are then normalized. Once we have an updated cell pair probability of containing a crew member we calculate the probability of an individual cell containing a crew member by summing every cell pair probability containing that cell.

$$P(\text{Alien in cell } j \ \land \ \text{Alien in cell } k)_{\text{t+1}} =$$
$$\begin{cases} \begin{cases} 0 & \text{if cell j and cell k are outside bot detector zone} \\ \frac{P(\text{Alien in cell } j \ \land \ \text{Alien in cell } k)_{\text{t}}}{\sum_{j}^{\text{all cells}} \sum_{k}^{\text{all cells}} P(\text{Alien in cell } j \ \land \ \text{Alien in cell } k)_{\text{t}}} & \text{if cell j or cell k are inside bot detector zone} \end{cases} & \text{if alien-beep is true} \\ \begin{cases} \frac{P(\text{Alien in cell } j \ \land \ \text{Alien in cell } k)_{\text{t}}}{\sum_{j}^{\text{all cells}} \sum_{k}^{\text{all cells}} P(\text{Alien in cell } j \ \land \ \text{Alien in cell } k)_{\text{t}}} & \text{if cell j and cell k are outside bot detector zone} \\ 0 & \text{if cell j and cell k are inside bot detector zone} \end{cases} & \text{if alien-beep is false} \end{cases}$$

First a hashmap of every possible combination of two cells is created, excluding pairs of the same cell. These values are then initialized with a uniform probability. Every time step depending on whether a beep is received these paired probabilities are set depending on where both cells is in relation to the cell detection square. When the alien is detected within the detection square all paired probabilities of where both pairs of cells are outside the detection area are set to 0. When an alien is detected outside the detection area then all paired probabilities where both cells in the pair are within the detection area are set to 0. Once this has been done all cell values are normalized.

$$\beta_t = \frac{1}{\sum_{j}^{\text{all cells}} \sum_{k}^{\text{all cells}} \sum_{j}^{\text{neighbors}} \sum_{k}^{\text{neighbors}} P(\text{Alien in neighbor of cell } j \land \text{Alien in neighbor of cell } k)} \\ * \, P(\text{Alien moves out of neighbor cell } j \land \text{Alien moves out of neighbor cell } k)$$

$$P(\text{Alien in cell } j \land \text{Cell in cell } k) = \beta_t \cdot \sum_{j}^{neighbors} \sum_{k}^{neighbors} P(Alien\ in\ neighbor\ of\ cell\ j \ \land \ Alien\ in\ neighbor\ of\ cell\ k) \cdot \\ * P(\text{Alien moves out of neighbor cell } j \ \land \ \text{Alien moves out of neighbor cell } k)$$

$$P(\text{alien in cell j}) = \sum_{k} P(\text{alien in cell j} \land \text{alien in cell k})$$

For alien movement the probability of the alien being located in the neighbors of each cell in the pair is multiplied by the probability the alien leaves that neighbor. This is done for every neighbor of each cell. These values are summed and then normalized. An additional case is also being considered for when the cells j and k share a neighbor cell. In this case the probabilities of the alien moving out of this cell are not independent. For this case a joint probability of moving out of the neighbor cell is used.

Evaluation Metrics for the Bots

We use three evaluation metrics to measure the performance of our bots:

$$\text{Average number of time steps taken to rescue all crewmates} = \frac{\text{Total number of time steps of all simulations in which 2 crewmates were rescued}}{\text{Total number of simulations in which 2 crewmates were rescued}}$$

Average number of moves to rescue all crewmates is the total number of time steps taken in the simulations in which the bot rescued both crewmates, divided by the number of crewmates rescued in those simulations

This metric tells us the number of time steps the bot took per simulation. Depending on whether we are analyzing the longevity of the bot's lifespan or the efficiency in finding the crewmate, a large timestep value could be a good or a bad indication.

$$\text{Average number of crewmates rescued} = \frac{\text{Total number of crewmates rescued in all simulations}}{\text{Total number of simulations}}$$

This is calculated as the total number of crewmates rescued (including the cases where the first crewmate was rescued by the bot and the bot got caught by the aliens while rescuing the second crewmate)

$$\text{P(Bot successfully avoiding aliens)} = \frac{\text{Number of simulations in which bot rescued both crewmates}}{\text{Total number of simulations}}$$

Probability of the bot not being caught by the alien is the number of simulations in which the bot rescued both crewmates divided by the total number of simulations. This value shows how good the bot is at avoiding aliens.

Bot 2, 5 and 8 Strategies

The goal of our bot 2, 5, and 8 strategies are to maximize or minimize the three metrics, average number of timesteps needed to rescue both crewmates, average number of crewmates rescued, and probability of successfully avoiding the aliens. To do this we developed three strategies.

Crewmate Strategy Probabilistic

In order to gain as much information as possible as quickly as possible the bots of our own design will move in a straight line away from their spawn location instead of moving randomly at the beginning of the simulation. This will insure the maximum distance between each data collection point minimizing the overlap of the newly generated crewmate probability gradients.

At every time step we receive or do not receive a crewmate beep. This gives us information on how far the crewmate is likely to be from our current position. If a beep is not received the cells close to the bot are given a low probability of containing the crewmate and cells far away are given a high probability of containing the crewmate. This creates a gradient around the bot for every timestep t. Using the probability functions above we update these probabilities at every timestep. The more these gradients around the bot overlap the less information will be gathered at every time step, so in order to gain as much information as possible as quickly as possible the bots of our own design will move in a straight line away from their spawn location instead of moving randomly at the beginning of the simulation. This will ensure the maximum distance between each data collection point minimizing the overlap of the newly generated crewmate probability gradients and maximizing the information gained on the crewmate's location.

Time step strategy Probabilistic

We also decided to have the crewmate move toward the center of the ship initially. This gives us the highest probability of moving toward the crewmates as there will always be more cells in front of us than behind us when moving toward the center.

There are two conditions that decide when the bot stops heading toward the center and starts heading toward a cell with the highest crewmate probability. If the bot reaches the center and or if a cell's probability of containing the crewmate reaches a high enough threshold.

Alien avoidance strategy Deterministic

The bots of our own design move toward the crewmate along the fastest path possible not taking into account the probability of an alien. If an alien is detected within the detection zone the bot switches over to alien avoidance mode. The alien detection zone is split into four zones along the diagonals. The probability of an alien being in each of these zones is summed and the bot moves to the neighbor cell in the zone opposite the zone with the highest alien probability. This results in the bot generally moving away from the alien when the alien is in it's detection zone. This continues until the alien is no longer in the detection zone at which point the bot resumes moving toward the cell with the highest crewmate probability.

Hypothesis

Bot 1 vs 2

Since bot 1 moves exclusively into known alien-free cells, the bot wastes a lot of potential opportunities to get closer to the crewmate merely because of being over-cautious. When the grid size is significantly large (not even too large, a 10*10 matrix would also do), the chances of running into the alien are fairly low and thus, avoiding every cell which has the slightest possibility of harboring an alien is inefficient.

Thus Bot 2 uses a comparatively less risk-averse strategy and focuses more on finding the crewmate by moving towards the cell with highest crewmate probability. In this process, it should cut down on a lot of unnecessary detours and reach the crewmate in fewer time steps. The alien avoidance strategy of bot 2 relies on dividing the grid into 4 sectors across the diagonals. The bot chooses the grid with the least alien density and continues its search for the crewmate in relatively safer areas. To make this search indiscriminate, the bot moves towards the center and collects information about its surroundings in the process to make better decisions. Thus, our expectation dictates that bot 2 should outperform bot 1 in the average step count metric. Also, provided that the matrix size is large enough (which is true in our case), bot 2 should perform better in terms of crewmates rescued as well. Aliens avoided might be close or even inferior to bot 1 in case of smaller-sized grids, but the overall performance on larger sizes should compensate for this.

Bot 3 vs 4 vs 5

Bot 3 is not expected to perform very well since it is not making use of the available knowledge base properly. The probability calculations for bot 3 assume only 1 crewmate while the environment has 2. Bot 4 on the other hand should do better because it uses the appropriate mechanisms for calculating probability in case of 2 crewmates. Bot 5 in turn should outperform these 2 as it also has a better strategy for alien avoidance instead of just waiting out the threat. Thus, we expect the bots to do progressively better.

Bot 6 vs 7 vs 8

Bot 6 (similar to bot 3) assumes only 1 alien when the environment actually has 2. So it would not perform very well. Bot 7 should perform slightly better as it does correct alien probability calculations. Bot 8 should outdo bot 7 since it also factors in alien avoidance and this time there are 2 aliens so avoidance should pay off even better.

Tests

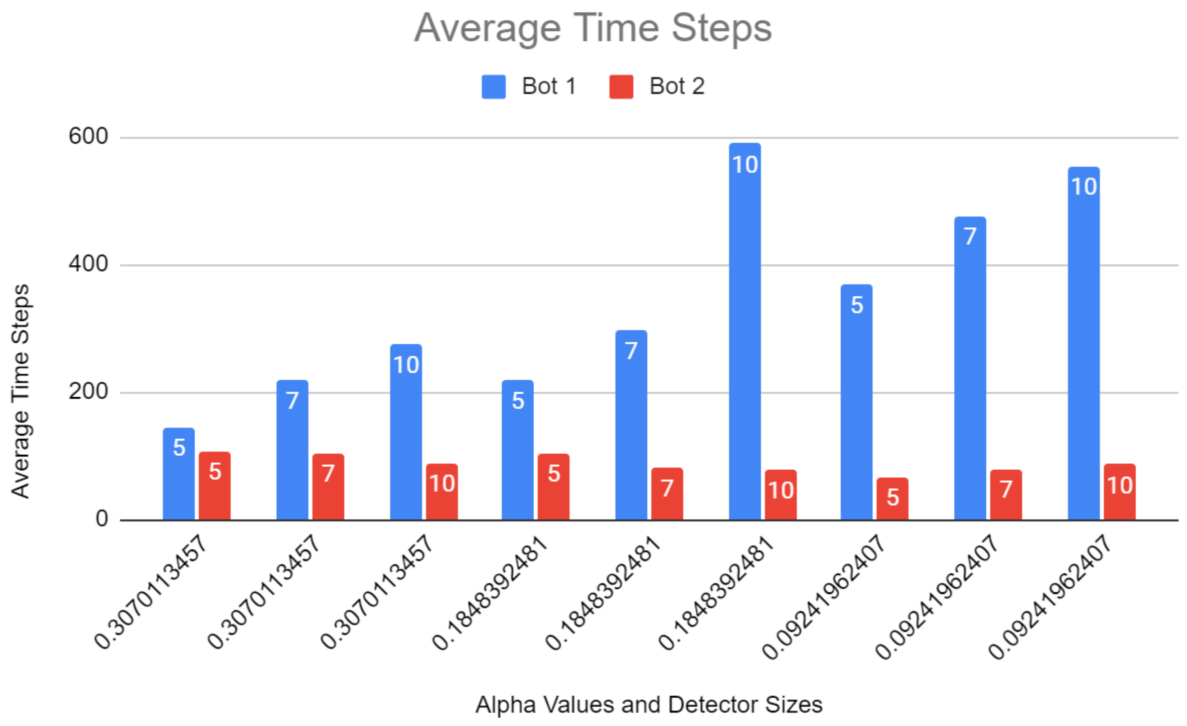
We run 30 simulations for each of the bots on a ship of size 30x30, and for each of these simulations, we evaluate 9 different combinations of alpha and detection size(K). The alpha value represents the sensitivity of our crewmate detector. Using the crewmate beep probability function we can calculate alpha values based on the probability of getting a crewmate beep from halfway across the ship.

We used alpha values that gave us 10%, 25% and 50% probabilities of detecting the crewmate when halfway across the ship. This gave us our three alpha value formulas:
0.3070113457 = -2*math.log(.1)/(30/2) for a 10% probability of detecting the crewmate,
0.1848392481 = -2*math.log(.25)/(30/2) for a 25% probability of detecting the crewmate, and
0.09241962407 = -2*math.log(.5)/(30/2) for a 50% probability of detecting the crewmate.

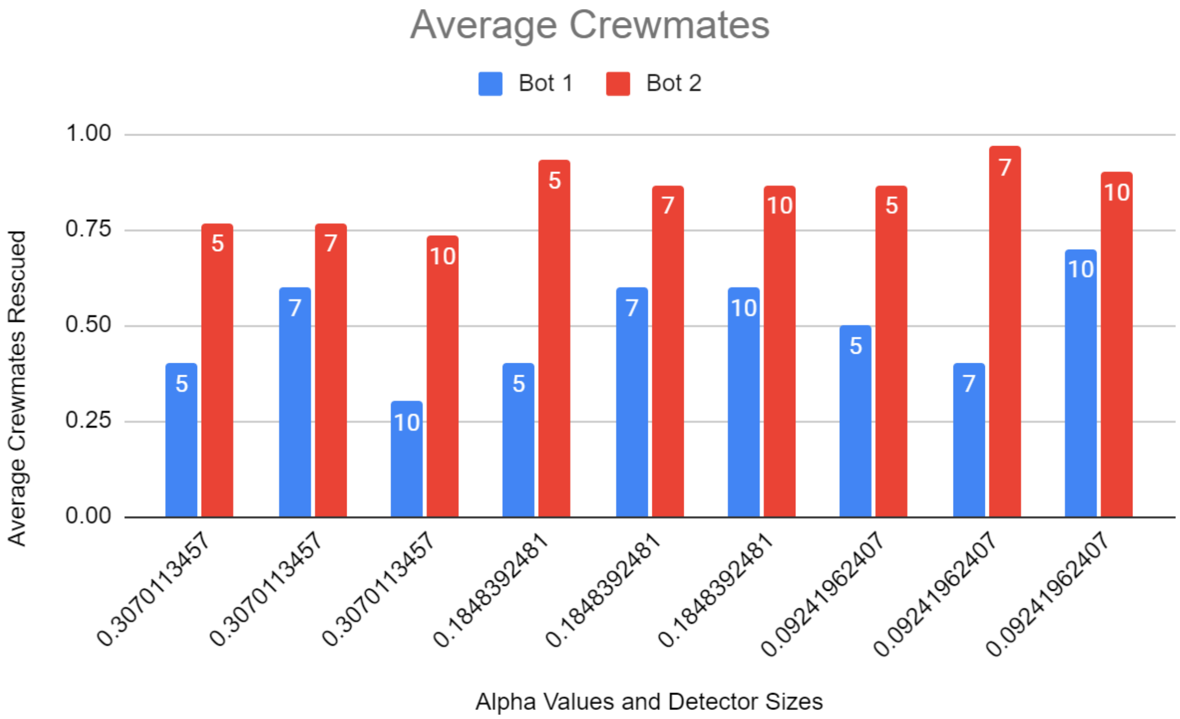
For k values we chose values that spanned as much of the range of possible k values as possible. With a ship size of 30x30 we can have a minimum k value of 1 and a maximum of 15, any higher and there would be a chance the alien couldn't spawn if the bot spawned at the center of the ship. To span this range of k values we used 5, 10, and 15 as our k values.

Results

Bot 1 vs Bot 2.



Since bot 1 moves exclusively into known alien-free cells, the bot wastes a lot of potential opportunities to get closer to the crewmate merely because of being over-cautious.



When the grid size is significantly large (not even too large, a 10*10 matrix would also do), the chances of running into the alien are fairly low and thus, avoiding every cell which has the slightest possibility of harboring an alien is inefficient.

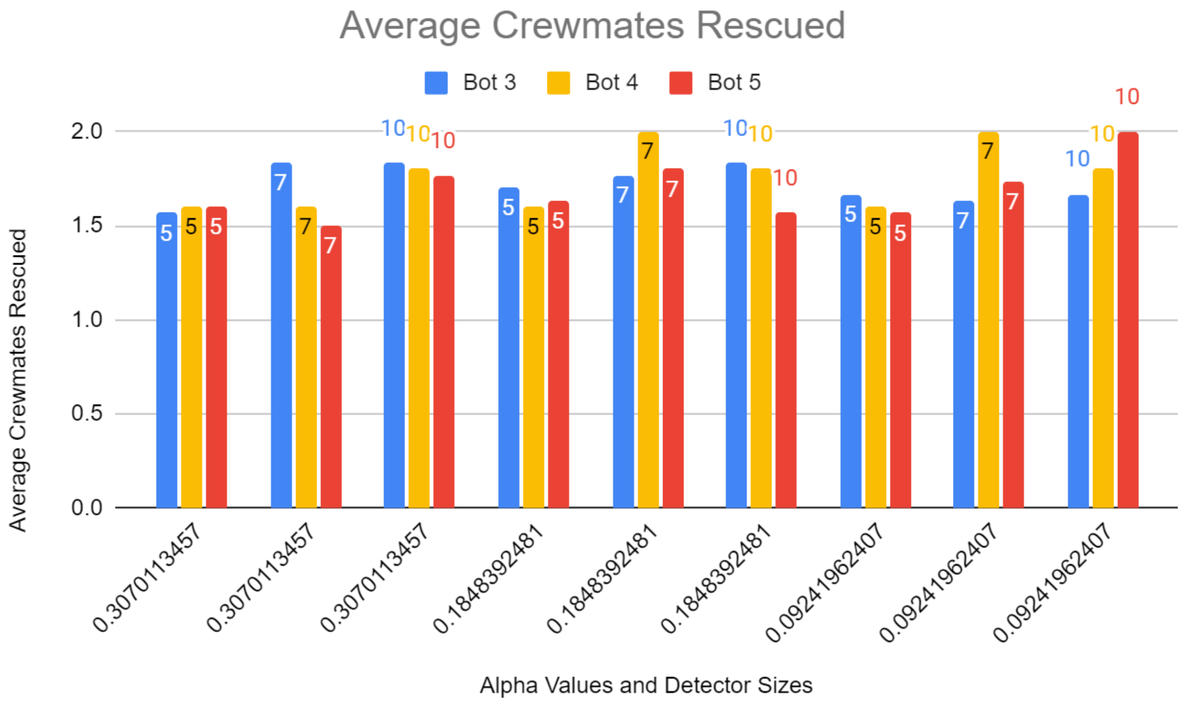
Thus Bot 2 uses a comparatively less risk-averse strategy and focuses more on finding the crewmate by moving towards the cell with highest crewmate probability. In this process, it should cut down on a lot of unnecessary waiting time and reach the crewmate in fewer time steps.

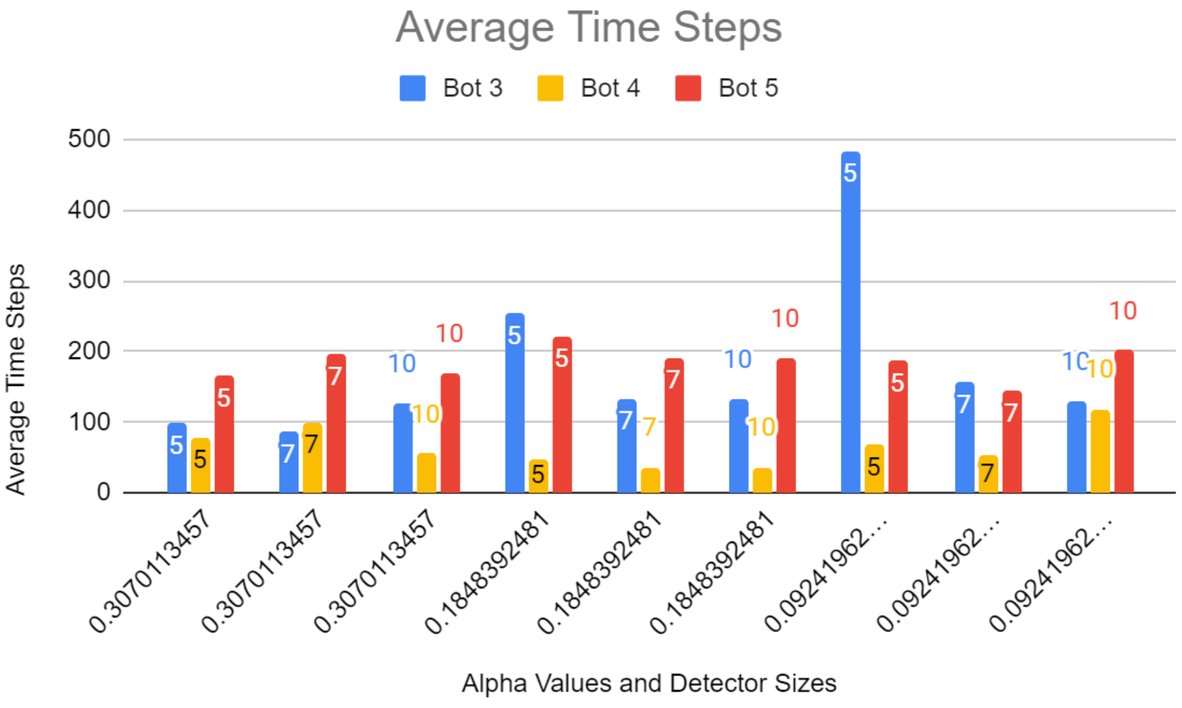
The alien avoidance strategy of bot 2 relies on dividing the alien detection grid into 4 sectors across the diagonals. The bot chooses the sector opposite the sector with the highest alien density and continues to do this until the alien is no longer detected.

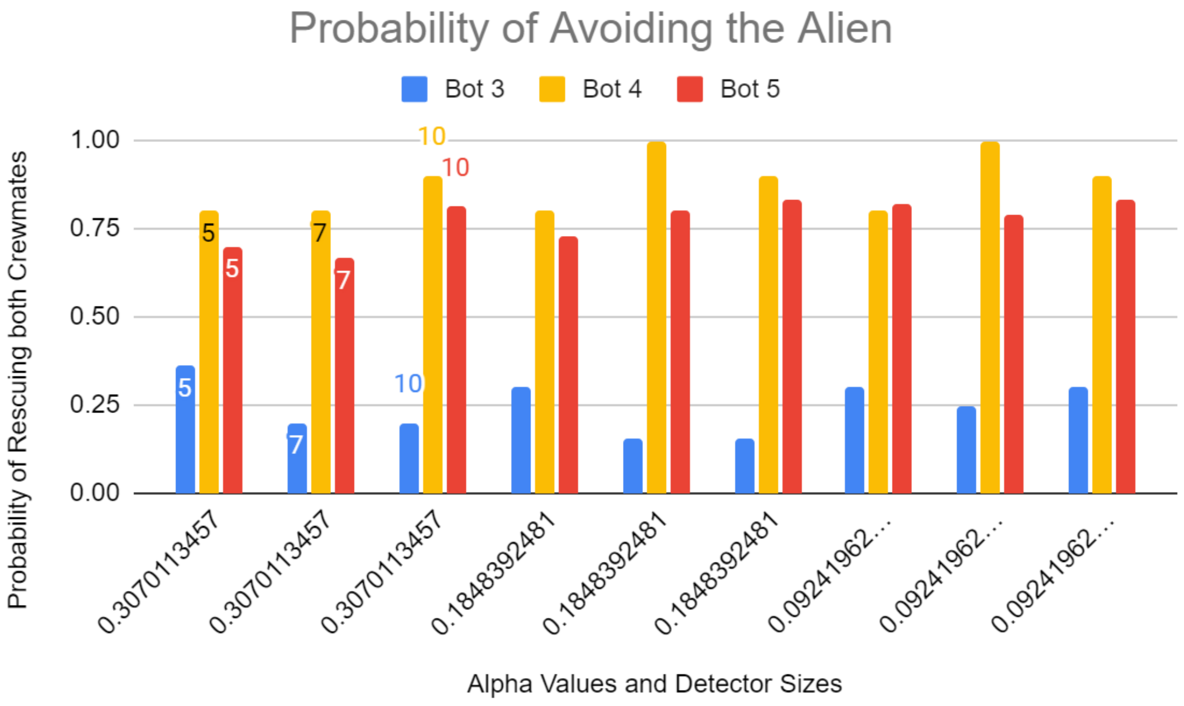
Thus, our expectation dictates that bot 2 should outperform bot 1 in the average step count metric. Also, provided that the matrix size is large enough (which is true in our case), bot 2 should perform better in terms of crewmates rescued as well. Aliens avoided might be close or even inferior to bot 1 in case of smaller-sized grids, but the overall performance on larger sizes should compensate for this.

As is visible in the graph above, the experimental results corroborate our hypothesis as bot 2 takes significantly less number of time steps to rescue crewmates and it also has a higher average count of crewmates rescued.

Bot 3 vs Bot 4 vs Bot 5.





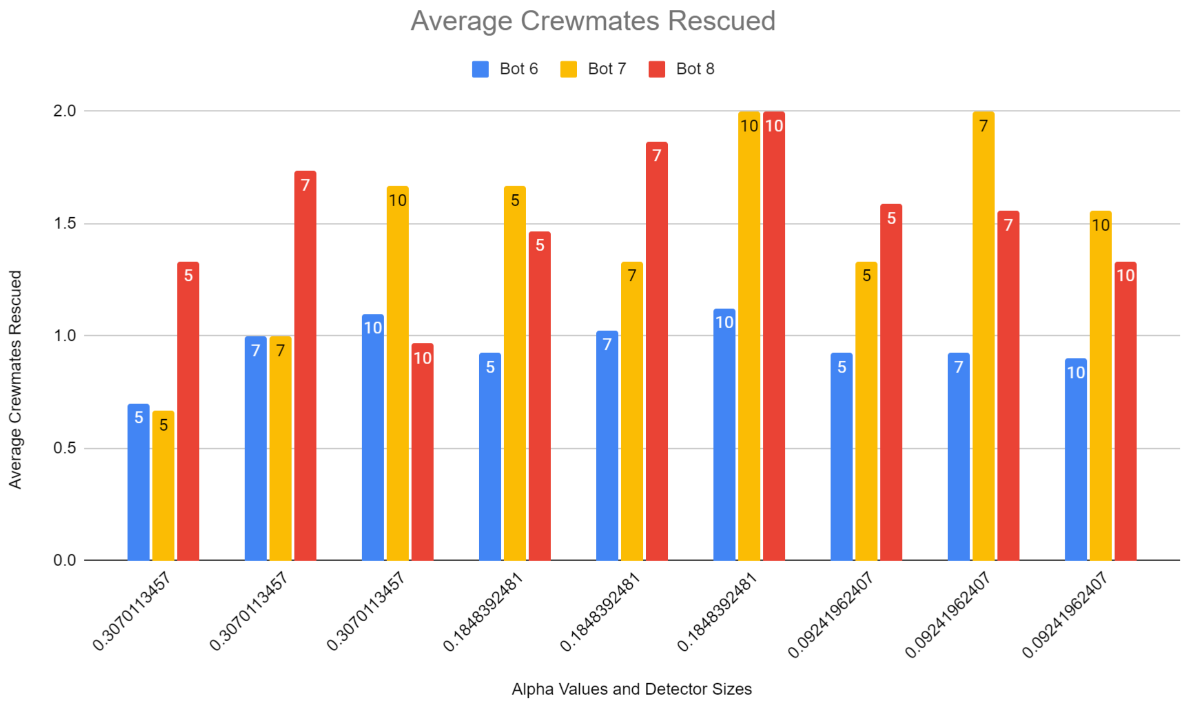


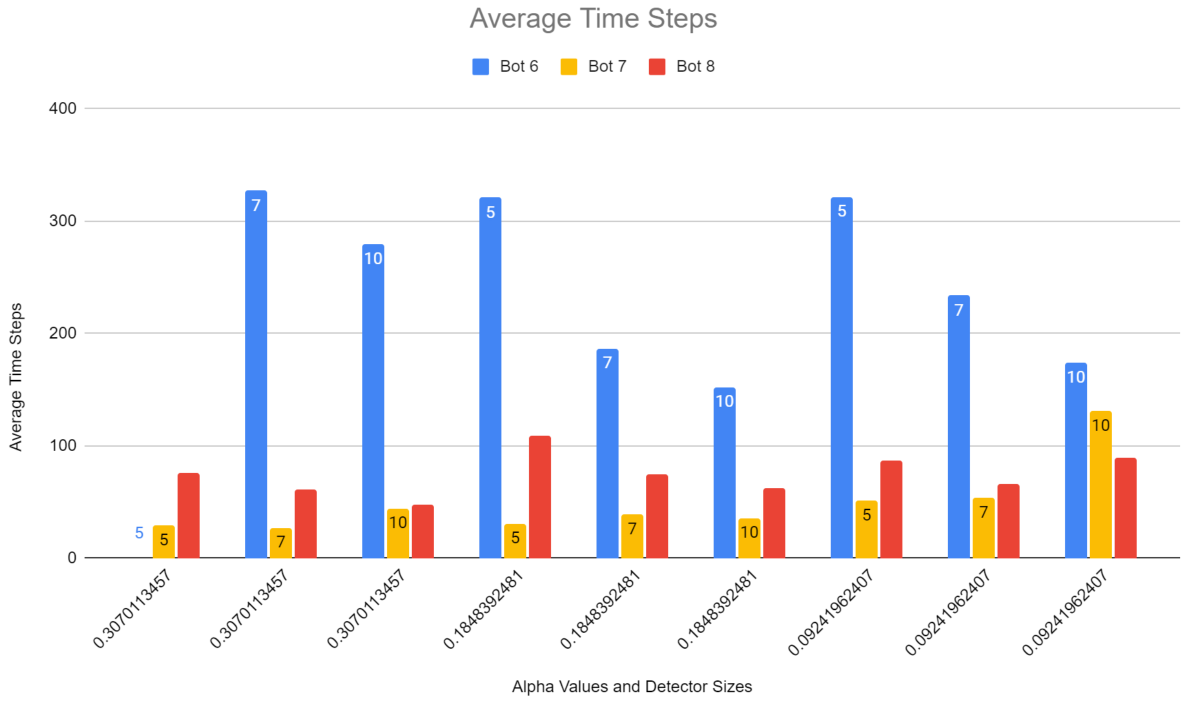
From the graphs, it is clear that Bot 3 performs better than Bot 4, which in turn out-performs bot 5. While this may seem counterintuitive at first, it points to the bias of this measure. The average time steps is only counted for simulations where the bot successfully rescues both crewmates. Because bots 3 and 4 do not use any alien avoidance strategy they are more likely to be caught by the alien in simulations that would require a high amount of time steps to rescue both crewmates. This biases bots 3 and 4 toward lower time steps, because the only time they are able to rescue both crewmates is when they spawn close to the bot, resulting in low time steps. Bot 5 does use an alien avoidance strategy which allows it to avoid being caught by the alien in simulations which require high time steps to rescue the crewmate. This leads to bot 5 having a significantly higher average time step than both bots 3 and 4.

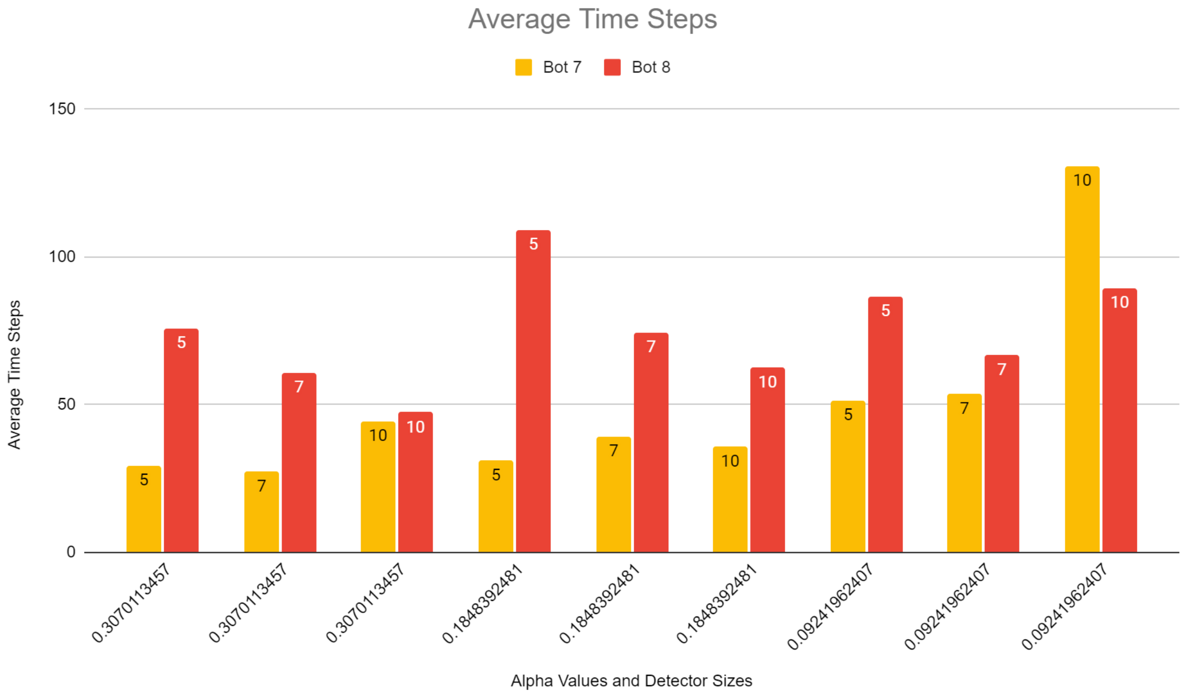
Surprisingly however, the average count of crewmates rescued per simulation is also better for bot 3. While this is not what we had expected from our hypothesis, we believe that the explanation could be related to the available options for alien movement. In a scenario where the alien has limited choices for movement, spawns close to a crewmate and is stuck within that region then the bot it is inevitably doomed to approach and be caught by the alien. This is a specific subset of scenarios however, and we would like to believe that running more simulations would lead us to results where bot 5 performs comparable or even better than bot 3 as our hypothesis postulates. Unfortunately, we could not expand our tests beyond 30 simulations in interest of paucity of time, and hence we acknowledge that bot 5 failed to satisfy our hypothesis.

Another indication that bot 5 is making good decisions is that it is definitely avoiding aliens better than bot 3. In fact, Bot 5 even outlasts bot 4 when we think of time steps as a measure of longevity of the bot's lifespan. Thus the probability of avoiding aliens is much better for bot 5 (can be seen directly when compared with bot 3, and can also be deduced from how frequently bot 5 outlives bot 4). The average crewmate rescue count however, has shown us that it could definitely do better.

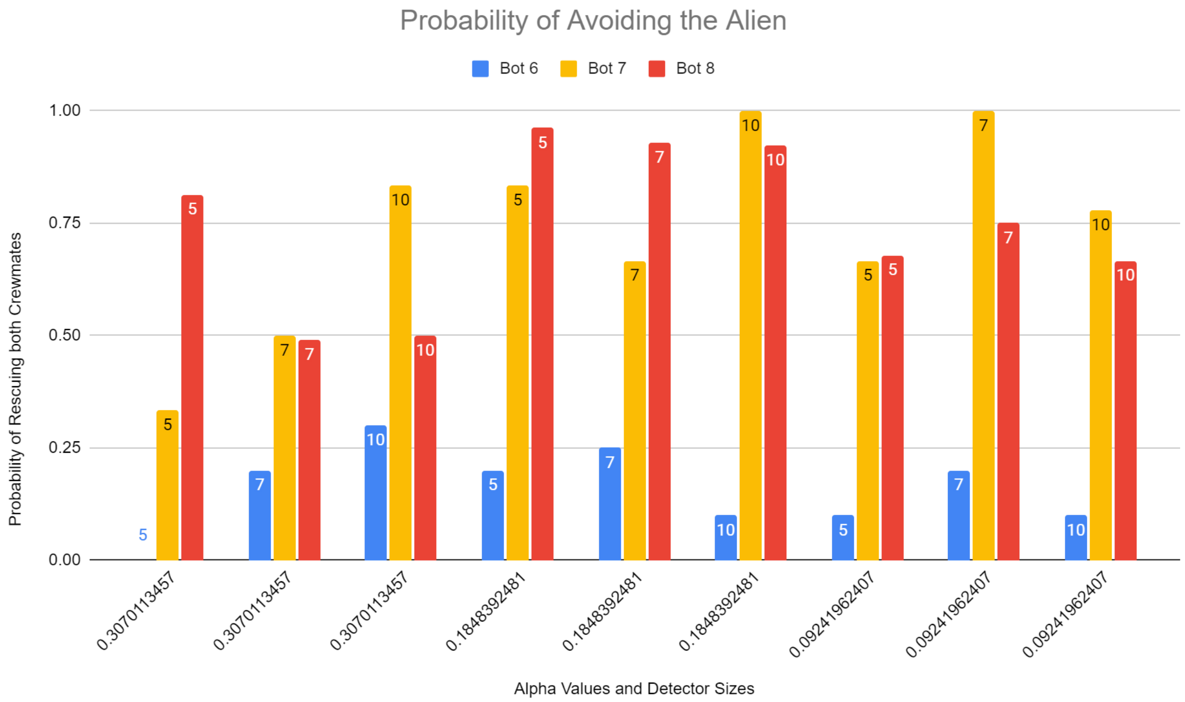
Bot 6 vs Bot 7 vs Bot 8.







Average Time Steps

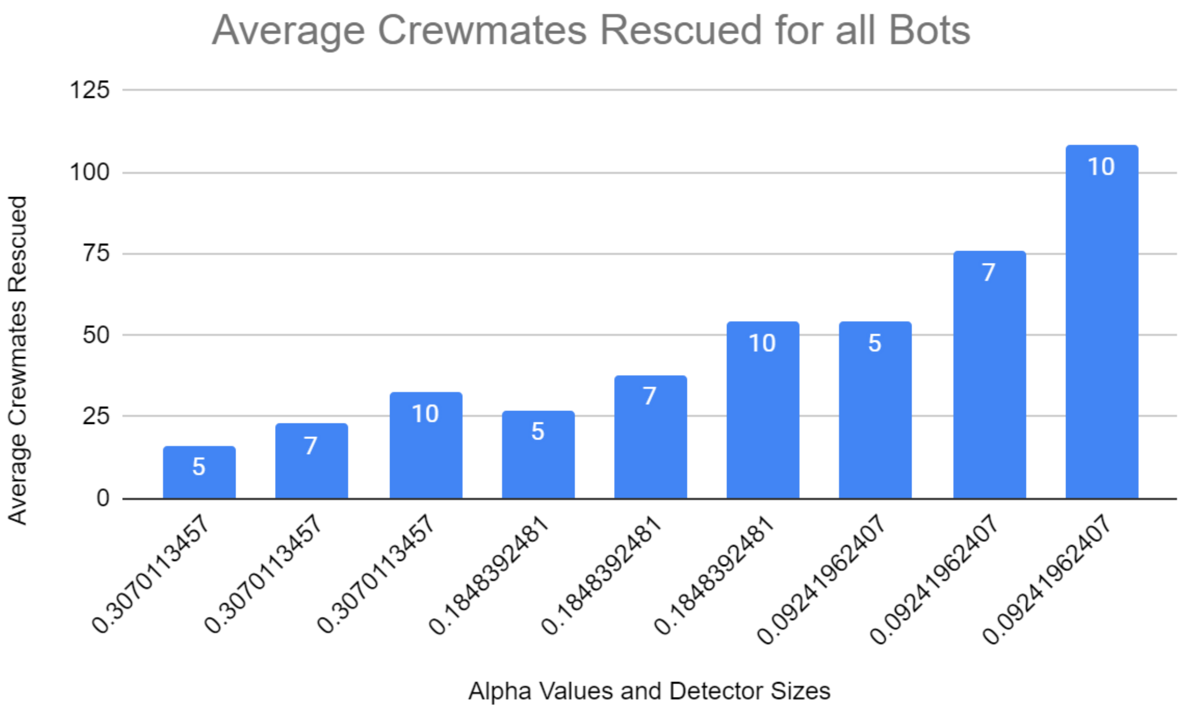


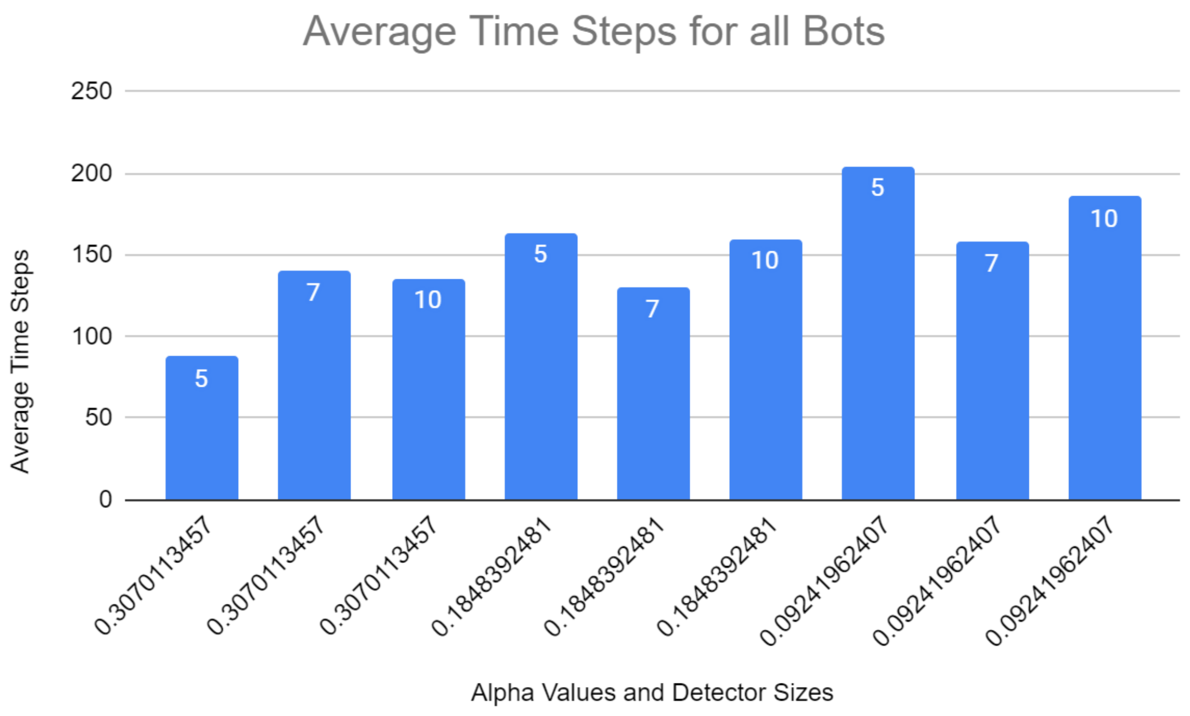
Probability of Avoiding the Alien

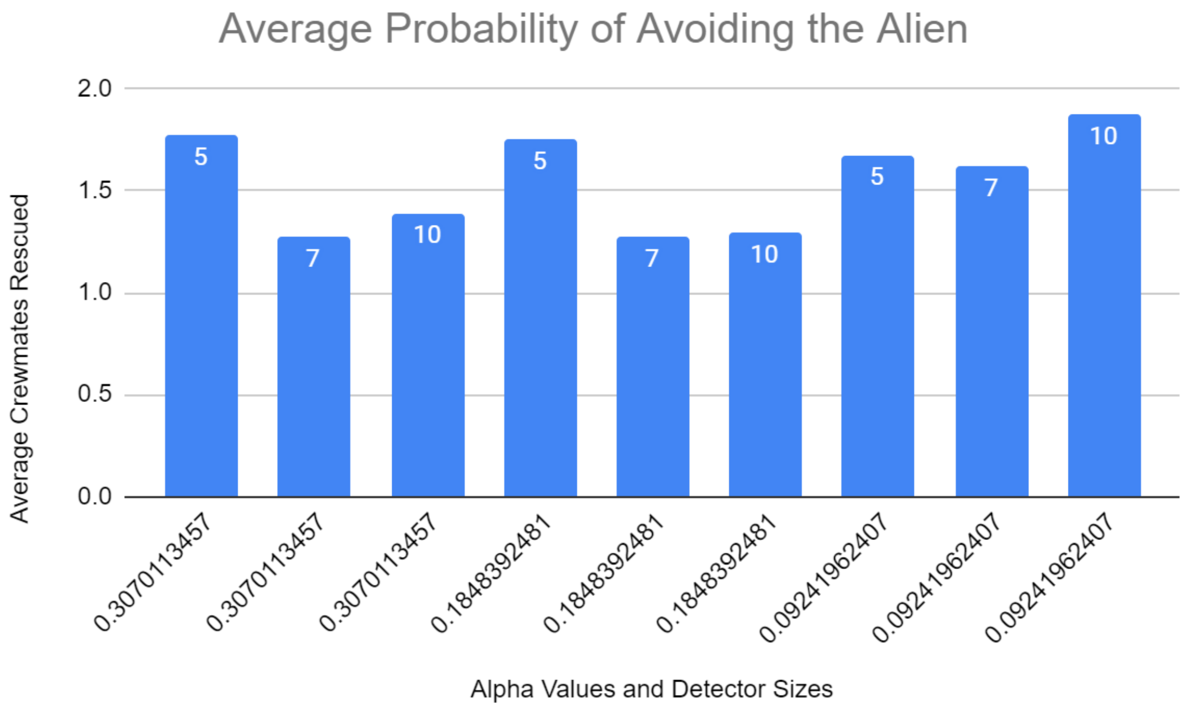
Bot 8 performs better than bot 6 in terms of average number of crewmates rescued. Further, it also takes less time steps per simulation as compared to bot 6. On the other hand, the performance of bot 7 does seem to be fairly similar (and in multiple instances even better) than bot 8. However, it is important to note that bot 7 has not been tested on a sufficient number of simulations to conclude which one is better overall. It is reasonable to assume that a higher number of aliens would eventually lead bot 8 to yield better statistics due to alien avoidance.

When the bot 6 detects that it is close to the crewmate it's probability functions set the farthest cells to 0. Creating a zone at the peripheries of zeros, but the second crewmate is sometimes in this zone of zeros causing it to find the first crewmate quite well but then hurting its chances of finding the second crewmate.

Alpha and Detector Size







It can be seen in the plot Average Crewmates Rescued for all bots that a lower value for alpha and higher value for k result in higher bot performance. Specifically for our tested values an alpha of .0924 and k value of 10 resulted in the highest average crewmates rescued across all bots.

In the Average Time Steps for all bots graph a weak negative correlation can be seen between alpha values and time steps and a weak positive correlation can be seen between k values and time steps. This results in high values of alpha and low values of k leading to lower time steps. Specifically for our data an alpha value of .307 and k value of 5 led to the lowest average time steps across all bots.

Again in the Average Probability of Avoiding the Alien graph a weak negative correlation for alpha and weak positive correlation for k values and probability of avoiding the alien can be seen. This results in lower values of alpha and higher values of k leading to higher probabilities of avoiding the alien. Specifically for our data an alpha value of .092 and k value of 10 lead to the highest probability of successfully avoiding the alien.

Errors

While we have been fairly thorough with our testing, the test results for bot 7 are admittedly on a fewer number of simulations (not 30) since we were running out of time and hence the values for bot 7 are somewhat erratic which is reflected in the experimental observations.

Conclusion

Given the right information and processing it correctly, the bots can make fairly reasonable decisions to rescue the crewmate. The data obtained from the sensors, and the parameters that change the amount of information we get (k and alpha) also impact the bots significantly.

Speculation

The bots could perform better if we assigned a weight based priority to finding the crewmates and avoiding the aliens based on the current situation. Remembering past patterns could also help the bot get clear of a similar situation in its future endeavors. A potential way to simplify the probability calculations would also speed up the bots massively which could reduce the resource constraints on current approaches.