CS520 Project 2 Data and Analysis

Aditya Girish, Rishik Sarkar

November 10 2023

1 Bot Designs and Algorithms

1.1 Initial Setup

This project is similar to the first one in that it involves a bot traversing through a grid maze of cells to save crew members while avoiding aliens. In our project version, we created a 30 x 30 NumPy grid containing 0s and 1s, with 0 representing open cells that the bot, aliens, and crew members can exist/move on and 1 representing closed cells (or walls). We have included the Python project within a single file, *main.py*, which contains all the processes for probabilistic updates, sensing, bot and alien movement functions, version-specific bot simulations, and a thorough testing process for each bot that I shall outline in section 3. In our Python project, we use the following characters to represent each entity:

- 0: Open Cell
- 1: Closed Cell
- 2: Bot
- 3: Alien(s)
- 4: Crew Member(s)

Furthermore, as defined in the project outline, there are eight possible scenarios involving different combinations of bots, crew members, and aliens. Bots 1 and 2 involve one bot, one crew member, and one alien; Bots 3, 4, and 5 involve one bot, two crew members, and one alien; and Bots 6, 7, and 8 involve one bot, two crew members, and two aliens. Each scenario involves certain similarities and differences in the grid traversal algorithm for the bot, but all of them involve deterministic and probabilistic processes that I shall outline below. I shall go over Bot 1 thoroughly, but for conciseness, I will only delve into the critical changes made to the other bots.

1.2 One Alien, One Crew

As mentioned earlier, Bot 1 and Bot 2 involve one bot, one alien, and one crew member within a 30 x 30 grid maze.

1.2.1 Bot 1

Bot 1 utilizes a crew sensor and an alien sensor at every time step to determine if a crew member or alien is nearby. On top of that, the probabilities of the crew member and alien being at a particular cell are updated every time the bot and alien move. Bot 1 stores the probabilities of the crew and alien being in a specific cell (i, j) within two probability matrices (represented as a dictionary of open cells + bot's current location) *crew_matrix* and *alien_matrix*. The process for Bot 1's maze traversal is as follows:

1. Initialize a 30 x 30 grid and place the bot, one alien, and one crew member at random locations. The alien is placed outside a (2k + 1) x (2k + 1) square around the bot, and the crew member is placed at any location apart from the bot's current cell. Also, initialize additional metric calculation variables for win_count, loss_count, move (i.e., counter for # of moves per crew member), etc.

2. Check for valid neighbors from the bot's current location (i.e., a list of open cells that the bot can move to at a given time)

3. Determine the best neighbor for the bot to move to (including itself) based on the probabilities in the *crew_matrix* and *alien_matrix*.

    - Bot 1 prioritizes cells with a 0 probability of an alien being in it. Suppose no cells are found with a 0 probability of the alien (reflected in the alien matrix). In that case, the bot prioritizes cells with a non-zero probability of the crew member being in it. This is to prevent the bot from going to cells it has visited before or those unlikely to contain a crew member (unless necessary, in case of an alien being nearby or if no such cells exist). When the bot finds two neighbors with an equal probability of the crew member being in it, it randomly breaks the tie and picks the next move.

4. The bot moves to the determined best neighbor

    - The number of moves is incremented by 1
    - If the bot enters the crew member's cell at this point, a win point is awarded to it, and a new crew member is placed at another random location on the grid (adhering to the previous constraints). The number of moves is also reset. The win point and the number of moves taken are recorded to be returned as metrics later.
    - If the bot enters the alien's cell or if the number of moves $\geq$ a timeout parameter, a marker is returned depicting that it has been captured or timed out, and a new session begins in the same grid with a newly placed bot, alien, and crew member. Except for the win and loss counts, every other variable (including the alien and crew matrices) is reinitialized.

5. After the bot has moved, the alien and crew matrices are updated based on the new beliefs and knowledge:

- Since we know that the new bot location does not contain a crew member (even if it did earlier, it has been teleported away) or an alien, we can set the current cell to have a 0 probability for both crew and alien in their respective matrices (dictionaries).
- We must now normalize the probabilities for the rest of the cells within the two matrices so that the total adds up to 1.

6. Next, the alien moves to a random neighboring cell or stays in place.

   - If the alien enters the bot's cell, a marker is returned depicting that the bot has been captured, and a new session begins in the same grid with a newly placed bot, alien, and crew member. Except for the win and loss counts, every other variable (including the alien and crew matrices) is reinitialized.

7. After the alien moves, the alien matrix is updated once again based on the new beliefs and knowledge:

   - When the alien moves, the probability of the alien being in the cell is distributed over all the neighboring cells and the current cell for every cell. The thought process behind this was that there should be an equal likelihood of the alien moving to any of its neighboring cells (or itself) every time the alien move occurs. This allows for a more thorough probabilistic distribution of where the alien might be after it moves.

8. The alien sensor is activated after the bot and the alien move. The bot inputs the k-value of the sensor as a parameter k, and this deterministic value represents the range of the detection square.

   - As mentioned earlier, the detection square is centered around the bot, and it returns a beep (i.e., the function returns *True*) if the alien is detected within the detection range (if alien-x $<$ bot-x + k and alien-x $>$ bot-x - k and alien-y $<$ bot-y + k and alien-y $>$ bot-y - k) and doesn't return a beep if not (i.e., the function returns *False*).
   - The beep value is stored within a variable *alien_detected*.

9. After the alien sensor activates, the crew sensor is activated. The bot inputs the α-value of the sensor as a parameter alpha, which modulates the probability of a beep occurring for a crew member d-steps away. The process taken by the sensor is as follows:

   - The bot maintains a dictionary called *d_lookup_table* that contains every cell the bot has visited as keys and the corresponding *d_dict* dictionaries as values. Each *d_dict* dictionary contains the shortest distance d from the corresponding bot to every other open cell in the grid as values with the open cells as keys.
   - If the bot's current cell has not been visited before, it utilizes a modified Dijkstra's search (using BFS) to calculate the shortest distance d to every open cell in the grid and stores the dictionary as the value with the bot's cell as the key within *d_lookup_table*.
   - If the bot has visited the current cell before (and it is a key within *d_lookup_table*, the *d_dict* is retrieved.

- The lookup table is maintained to avoid running Dijkstra's whenever a bot enters a cell. Although it increases space complexity, the process dramatically reduces the total computation time needed for the sensor to run.
- The *d_dict* is used to find the actual shortest deterministic distance d to the crew member from the bot, and the sensor returns a beep (i.e., *True*) with probability: $e^{(-\alpha \cdot (d-1))}$.
- As the α value increases, the probability of the beep decreases when d is farther away.
- The beep value is stored within a variable *crew_detected*.

10. After the *alien_detected* and *crew_detected* variables are stored, the *alien_matrix* and *crew_matrix* are updated based on whether a beep was returned for each sensor.

11. The updates occurring within the *alien_matrix* are as follows:

    - If the alien sensor beeps, the bot knows that the alien is within the range of the detection square. Thus, it can be inferred that every cell outside the detection square must have a probability of 0 (since there is only one alien), and the probability of the alien being in any of the cells inside the square must be normalized.
    - If the alien sensor doesn't beep, the bot knows that the alien is not within the range of the detection square. Thus, it can be inferred that every cell inside the detection square must have a probability of 0, and the probabilities of the alien being in any of the cells outside the square must be normalized.

12. The updates occurring within the *crew_matrix* are as follows:

    - If the crew sensor beeps, the bot modifies the probabilities of all the cells within the crew matrix by multiplying the prior probability of the cell containing the crew member by the likelihood of receiving the beep given its deterministic distance d from the bot (i.e., $e^{(-\alpha \cdot (d-1))}$ for the cell's d). The distance d is once again retrieved from the *d_lookup_table* defined earlier (and updated with the crew sensor).
    - If the crew sensor doesn't beep, the bot modifies the probabilities of all the cells within the crew matrix by multiplying the prior probability of the cell containing the crew member by the likelihood of not receiving the beep given its deterministic distance d from the bot (i.e., $1 - e^{(-\alpha \cdot (d-1))}$ for the cell's d).
    - This update is done because the beep being heard is more likely in cells closer to the bot if the crew is there, so the presence of a beep indicates that the crew member might be closer. In the absence of the bot, we do the opposite since the crew member is likelier to be farther away.
    - Finally, the probabilities for all the entries are normalized by dividing them by the sum of all the probability estimates for the total to equal 1.

13. This process continues until the total number of wins + losses equals a maximum iteration threshold. At the end of the simulation loop, the average rescue moves (i.e., sum of move for every win // total wins), the probability of rescuing the crew (i.e., wins / (wins + losses)), and the average number of rescued crew (i.e., wins) are returned.

This was the most thorough explanation of the processes we took for a single bot. From here on, I shall focus only on the steps taken by the other bots that differ from Bot 1.

1.2.2 Bot 2

Bot 2 takes the same approach as Bot 1 for every function except for the one used to calculate the best next move for the bot given the alien and crew matrices.

- The custom Bot 2 calculates the same crew and alien probability matrices as Bot 1 but uses the matrices a little differently. Instead of prioritizing the neighboring cells with a guaranteed probability of 0 for the alien being there, the bot prioritizes cells with a crew probability of > 0. Thus, it takes more risks than Bot 1 and fewer moves to save crew members, but at an increased risk of getting caught by aliens.
- This made sense in this context because we are primarily evaluated on the number of moves taken to save the crew, crew save probability, and average number of crew members saved. Thus, prioritizing saving the crew member over avoiding the alien allows Bot 2 to perform better than Bot 1 potentially.

1.3 One Alien, Two Crew

Bot 3, Bot 4, and Bot 5 involve one bot, one alien, and two crew members within a 30 x 30 grid maze.

1.3.1 Bot 3

Bot 3 takes the same approach as Bot 1, except it has one additional crew member. Since the functions don't have to account for this extra crew member in calculating probabilities, the functionality is generally the same, but with some differences:

- If the bot saves one of the two crew members, nothing happens, and it continues on its path. The crew member probability matrix gets subsequently updated and normalized to account for this, and the bot continues traversing the grid. The bot only receives a win point if it captures both original crew members in a row, so the functions are slightly modified to account for this new distribution.

1.3.2 Bot 4

Bot 4 takes a similar approach to Bot 1 in terms of functionality and setup approach to Bot 3. However, there are some distinct differences since it accounts for both crew members while modifying probabilities:

- Bot 4 uses a 2D matrix with entire cells as indices to account for crew probabilities. It maintains an index-mapping dictionary that maps every open cell (including the bot's current

cell) to a unique index from 0 to len(open_cells) + 1. Then, the indices are used as the row and column indices of the crew probability matrix. This approach is much more efficient than a 4D matrix of [p][q][r][s] for cells (p, q) and (r, s), instead simplifying the process into two dimensions. Furthermore, the probability updates are vectorized using NumPy matrix manipulation techniques and are thus much more streamlined.

- As a consequence of these changes, Bot 4 uses a different method to initialize the matrix and update it. Whenever the bot moves to a cell, the entire row and column for that particular cell index are set to 0 to indicate that neither of the two crew members can exist on any combination of the current cell and any other open cell. Subsequently, the matrix is normalized to ensure consistency.
- The function that enables the bot to select the best neighbor to move to is also updated to work with the new 2D matrix setup and finds the best move by summing the probability of the crew in the current cell with all other cells and avoiding double-counting through subtraction. The likelihood of either of the crew members being at a particular cell can be observed using the probabilities of crew members being in that specific cell and all other cells. The higher the total probability, the more likely the crew member is in the cell.
- Finally, the crew matrix updates due to receiving a crew sensor beep are also updated to accommodate the new matrix structure, and the exact probability updates are outlined within the next section.

1.3.3 Bot 5

Bot 5 takes the same approach as Bot 4 for every function except for the one used to calculate the best next move for the bot given the alien and crew matrices.

- The custom Bot 5 combines a probabilistic and deterministic approach in determining which cell to move to at a given time step. The key difference between Bot 5 and Bot 4 is that Bot 5 uses prior information about the position of the crew members on the grid to determine which one to save first (and, thus, which cell to move to).
- To distinguish between the crew members, the bot uses the d_lookup_table generated by the crew sensor to determine the closest crew member to the bot's current location (i.e., min(d)). No external deterministic calculations are happening within this function, so the bot can only use the d_lookup_table if it has been to the location before and if the crew sensor has run at least once. If the value for the key of the current bot cell in the lookup table returns *None*, the bot uses the first crew member in the crew list in its calculations.
- After the bot finds the closest crew member, it stores the relative position in terms of four boolean variables: up, down, left, and right. The variables are set to true depending on the x and y positions of the crew compared to the bot.
- Finally, Bot 5 uses the same technique as Bot 4 in determining the best cell to move to, except that it prioritizes saving crew members over escaping aliens (just like Bot 2). However, in

the case of two cells having equal probabilities of containing the crew member, the bot uses the position variables instead of randomly breaking ties to filter out the result and return the best relevant move. If there are still multiple relevant tied neighbors, the bot randomly chooses between them.

1.4 Two Aliens, Two Crew

Bot 6, Bot 7, and Bot 8 involve one bot, two aliens, and two crew members within a 30 x 30 grid maze.

1.4.1 Bot 6

Bot 6 takes the same approach as Bot 1, except it has one additional crew member and one more alien. Since the functions don't have to account for this extra crew member and alien in calculating probabilities, the functionality is generally the same, but with some differences:

- If the bot saves one of the two crew members, nothing happens, and it continues on its path. The crew member probability matrix gets subsequently updated and normalized to account for this, and the bot continues traversing the grid. The bot only receives a win point if it captures both original crew members in a row, so the functions are slightly modified to account for this new distribution.
- It is typically a little more difficult for the bot to save crew members since there is one additional alien, but no techniques to account for this.

1.4.2 Bot 7

Bot 7 takes a similar approach to Bot 1 in terms of functionality and setup approach to Bot 6. However, there are some distinct differences since it accounts for both crew members and both aliens while modifying probabilities:

- Bot 7 uses two 2D matrices with entire cells as indices to account for crew and alien probabilities. It maintains an index-mapping dictionary that maps every open cell (including the bot's current cell) to a unique index from 0 to len(open_cells) + 1. Then, the indices are used as the row and column indices of the crew and alien probability matrices. This approach is much more efficient than a 4D matrix of [p][q][r][s] for cells (p, q) and (r, s), instead simplifying the process into two dimensions. Furthermore, the probability updates are vectorized using NumPy matrix manipulation techniques and are thus much more streamlined.
- Due to these changes, Bot 7 uses a different method to initialize and update the matrices. Whenever the bot moves to a cell, the entire row and column for that particular cell index are set to 0 in both the crew and alien matrices to indicate that neither of the two crew members or two aliens can exist on any combination of the current cell and any other open cell. Furthermore, at the start of the simulation, all rows and columns with the mapped

index of the open cells within the detection square range of the bot are set to 0 since the alien cannot be initialized within that threshold. Subsequently, both matrices are normalized to ensure consistency.

- The function that enables the bot to select the best neighbor to move to is also updated to work with the new 2D matrix setup and finds the best move by summing the probability of the crew in the current cell with all other cells and avoiding double-counting through subtraction. The lists of neighbors with probability 0 for aliens and non-zero probability for crew members are also found using row and column sums instead of individual cells because all possibilities involving the particular cell must be considered. The likelihood of either of the crew members being at a particular cell can be observed using the probabilities of crew members being in that specific cell and all other cells. The higher the total probability, the more likely the crew member is in the cell.

- The crew matrix and alien matrix updates due to receiving a crew sensor or alien sensor beep are also updated to accommodate the new matrix structure, and the exact probability updates are outlined within the next section.

1.4.3 Bot 8

Bot 8 takes the same approach as Bot 7 for every function except for the one used to calculate the best next move for the bot given the alien and crew matrices.

- The custom Bot 8 functions similarly to Bot 5 in that it attempts to filter out the most relevant cells it can move to at every time step to break ties between multiple same-probability neighbors instead of randomly selecting a move.

- However, the approach taken by Bot 8 to accomplish this is a little different. Instead of utilizing external information such as the d-value of cells, the bot finds the quadrant that is safest from aliens and most likely to contain crew members. It does so by adding up the probabilities of all the cells within the top, bottom, left, and right halves of the grid for the alien and crew matrices and setting up variables for each cardinal direction for both aliens and crew. This information is subsequently used to determine the best cell to move out of the ones with the highest crew probability.

- The bot breaks ties by finding if the move with equivalent probability is either in the safe quadrant or crew quadrant and only selecting moves that fulfill both criteria. Unlike Bot 5, however, Bot 8 does not prioritize saving crew members over escaping aliens and takes the more orthodox approach. This is because the added risk of having two aliens makes it more difficult for the bot to save crew members if it doesn't prioritize 0 alien probability cells.

2 Probability Models and Updates

2.1 One Alien, One Crew

2.1.1 Bot 1

For this bot, two dictionaries or hash maps were used to manage and keep track of the probabilities. Specifically, one dictionary consisted of the probabilities of an alien being in an open cell i, while the other was the probability of the crew being in an open cell i. First, it is essential to note that the bot has a uniform probability of the crew being in any open cell, but the current one it is in and a uniform probability of the alien being in any of the cells outside the detection square. There are four times that Bot 1 updates its probabilistic beliefs.

One of those times is when the alien detector goes off. Since the bot is only aware of whether the alien sensor comes back to give a positive or negative response, the probability of an alien being in a particular cell is only based on that when doing this update. These probabilities can be written out as P(alien in k — detect from i) and P(alien in k — no detect from i) where the bot is in cell i. For example, in the case that the detector gives a positive detection, the probability can be expanded as P(alien in k — detect from i) = P(alien in k) * P(detect from i — alien in k) $\sum_{jcells}$ P(detect in i and alien in j). Here, P(alien in k) is our prior belief. If it is a positive detection, then it is known that there must be an alien in the detection square. This means that the probability that the alien is in a cell outside the detection square is 0, and consequently, it can be set as such for these cells as the probability P(detect from i — alien in k) is 0 for them. However, P(detect from i — alien in k) is 1 for cells inside the detection square. Consequently, this means that the probability for cells inside the detection square, say like cell k, is updated by P(alien in k) / $\sum_{jcells}$ P(detect in i and alien in j). The denominator is marginalized over all detection square cells to normalize P(alien in k) since the bot does not know which cell in the detection square has the alien if the detector is positive. Alternatively, if the sensor returns negative, the probability to compute is P(alien in k — no detect from i). This is expanded similarly to the other probability and is calculated similarly. The stark difference is that now cells inside the detection square have probability 0, and cells outside the square are updated by P(alien in k) / $\sum_{jcells}$ P(no detect in i and alien in k) where j is all cells outside the detection square.

Another time the bot updates beliefs is when the crew sensor is run. It is important to note again that for a crew member that is d steps away, there is a $e^{(-alpha*(d-1))}$ probability where alpha is user-specified that the sensor gives a beep. The probabilities for this case can be formally represented as P(crew in k — beep in cell i) and P(crew in k — no beep in i), where cell i is the current cell of the bot. In the case that the bot gets a beep, the probability of a crew being in an open cell can be updated on the expansion of P(crew in k — beep in cell i) where it is P(crew in k) * P(beep in cell i — crew in k) / $\sum_{alljcells}$ P(crew in j) * P(beep in i— crew in j) where j is all open cells. In essence, P(crew in k) here is the prior probability of a crew being in cell k, and P(beep in i— crew in j) is the probability that the bot in cell i hears a beep from a crew in cell j. From a coding standpoint, this probability can be interpreted as multiplying the probability of a crew being there for all cells by P(beep in cell i — crew in k), which is $e^{(-alpha*(d-1))}$ where d is the

shortest distance from that cell to the bot's cell. Subsequently, you divide these probabilities by the sum of the probability of a crew in a cell times the probability of it giving a beep over all the open cells. Alternatively, if there is no beep for a bot in cell i, the method for updating the beliefs is similar, with the only difference being that the conditional probability now in the expansion is P(no beep in i — crew in j), which is just 1 - $e^{(-alpha*(d-1))}$.

Another time the bot updates its beliefs is when it moves to a new cell. This probability can be formally represented as P(alien in j — alien not in i) and P(crew in j — crew not in i) for all open cells where i is the new cell that the bot has moved into. To update the probabilities of an alien being in a cell j for all open cells, the expansion of the earlier probability is helpful where P(alien in j — alien not in i) = P(alien in j) * P(alien not in i — alien in j) / P(alien not in i). Here, P(alien not in i — alien in j) is 1 for all other cells except for when cell j is the same as cell i, and P(alien not in i) is just 1 - the prior probability of the alien being in cell i or (1 - P(alien in i). As a result, from a coding standpoint, the probability of an alien being in the respective cell for all open cells is divided by 1 minus the probability of an alien in the new cell of the bot if the open cell is different from the bot's new cell. If the cell is the same as the bot's new cell, then the probability of an alien there is 0. The same type of update is done for crew probabilities as that probability is similarly expanded as P(crew in j) * P(crew not in i — crew in j) / P(crew not in i). It is important to note the bot has the option to stay in place. In the case that this happens, the probabilities stay the same as this "move" of the bot, which has not provided any new information to update its beliefs.

Lastly, the bot updates beliefs when the alien moves. For example, consider that the alien was in cell k at time t, and at time t+1, the alien has moved to cell j. Formally, this probability can be represented as P(alien is now in cell j at time t+1), which expands into $\sum_{kcells}$ P(alien was in cell k at time t) * P(alien now in cell j at time t+1 — alien was in cell k at time t) for all open cells. Evidently, cells not neighbors of cell k will have a probability of 0 since the alien cannot move there from cell k in 1 time-step. For cells that are neighbors of cell k, P(alien now in cell j at time t+1 — alien was in cell k at time t) is 1/(number of neighbors of j + 1) since the alien can move randomly to any of its neighbors or stay in place. From a coding standpoint, once an alien moves, the alien probabilities for cells not neighbors to the previous cell of the alien were set to 0, while the probabilities of neighbor cells were updated as mentioned above.

It is important to note that the order of updating beliefs proved to be important for the alien probabilities. For example, since the alien detection was run after the alien moved, the probability updates from the alien moving were often overridden by the probability update based on the detection.

2.1.2 Bot 2

The probability updates for Bot 2 are exactly the same as Bot 1, with the only difference being in the way the probabilities are used.

2.2 One Alien, Two Crew

2.2.1 Bot 3

Like Bot 1 and 2, this bot kept track of the probabilities of the alien being in an open cell and a crew member being in an open cell using two dictionaries. A difference to note this time is that the initial probabilities for a crew being in an open cell considered the fact that two crew members could not be in the same cell and that a crew member cannot be in the bot's initial position. This can be represented as 1/ (number of open cells - 1).

In terms of the probabilities updates, this bot updates its beliefs for the alien and crew probabilities exactly like Bot 1. However, since there are now two crew members, the bot continues to run after capturing the first crew member. Probabilistically, once the first crew is rescued, the probability of a crew being in that cell is 0. Subsequently, since there is still a crew member left, the crew probabilities for all the other open cells are normalized to adjust for the rescue of this crew member.

2.2.2 Bot 4

For this bot, a dictionary was used to keep track of the alien probabilities, which is similar to the previous bots. However, for the crew probabilities, a 2D matrix was used. In essence, this matrix represents all the possible pairs of cells from the open cells and has a size of (number of open cells) * (number of open cells). In order to index this matrix efficiently, another dictionary was used to map each cell to an "index value," which would then be used to look up the crew probability for a certain pair of cells. For example, if (1,1) maps to index 2 and (3,4) maps to index 32, then the value matrix[2][32] would be the probability of a crew being in (1,1) and a crew being in (3,4). Similar to Bot 3, the initial crew probabilities are based on the fact that two crew members cannot be in the same cell, and there can be no crew member in the bot's initial cell. Consequently, the probability of all pairs that include the bot's cell is 0, and the probability of all pairs that consist of the same two cells is 0 as well. The probability of the two crew being in the pair is then distributed uniformly among the remaining pairs. Along with the new way to manage the crew probabilities, the updates to the crew probabilities are also different.

One of the times the bot updates its crew beliefs is when the crew sensor is run. Based on whether a beep is heard or not from the bot at cell i, the probabilities can be formally represented as P(crew in cell k and cell j — beep from cell i) and P(crew in cell k and cell j — no beep from cell i). In the case that the bot heard a beep in cell i, the probability of the two crews being in a pair of cells k and m from the set of open cells is updated on the expansion of the previous probability, which is P(crew in cell k and cell m — beep from cell i) = P(crew in cell k and cell m) P(beep from cell i — crew in cell k and cell m) / $\sum_{(cell a,b) pairs}$ P(crew in cell a and b) P(beep from cell i — crew in cell a and b) where (a,b) is all pairs of cells. The summation in the denominator is over all pairs of cells. The probabilities P(crew in k and cell m) and P(crew in cell a and b) are our prior probabilities for the respective pair. However, the conditional probability in the expansion is more challenging to calculate due to the possibility it corresponds to the probability of getting a beep from both crew members or one of the crew members. The beep itself does not provide any information to

distinguish this. As a result, this conditional probability can be considered as 1 - P(no beep from the crew members). Subsequently, P(no beep from the crew members) can be broken down into independent events of both the crew not sending a beep: P(no beep from crew at cell k) * P(no beep from crew at cell m). Both these probabilities are in the form of 1 - $e^{(-alpha*(d-1))}$ where d is the shortest distance from the bot's cell i to the cell the crew member is in. From this point, the conditional probability P(beep from cell i — crew in cell k and cell m) can be calculated as it is 1 - ((1 -$e^{(-alpha*((dforcrewk)-1))}$) * (1 -$e^{(-alpha*((dforcrewm)-1))}$. From a coding standpoint, this update was done by multiplying the crew probability for each pair of cells by the expression above and then dividing by the summation of all the probabilities to normalize. It is important to note that the same approach is used to update the crew probabilities for when no beep is detected as the probabilities expand similarly. However, the difference is in the conditional probability in the expansion in that it is now P(no beep from cell i — crew in cell k and cell m), which would break down as P(no beep from crew k) * P(no beep from crew m) or ((1 -$e^{(-alpha*((dforcrewk)-1))}$) * (1 -$e^{(-alpha*((dforcrewm)-1))}$. The other time the bot updates its crew beliefs is when the bot moves. In the case that a crew member is not rescued, the probability can be formally represented as P(crew in cell k and j — crew not in cell i) for all pairs of cells (k, j) where i is the cell the bot moved into. The probability can be expanded into P(crew in cell k and j — crew not in cell i) = P(crew in cell k and j) P(crew not in cell i — crew in cell k and j) / $\sum_{(cella,b)pairs}$ P(crew in cell a and b) P(crew not in cell i — crew in cell a and b). Evidently, the probability is 0 for all pairs of cells that include cell i. This means that all the pairs of cells that do not have cell i as one of the cells are updated by dividing by the summation of all probabilities of pairs that do not have cell i as one of the cells.

2.2.3 Bot 5

The probability updates for Bot 5 are exactly the same as Bot 4, with the only difference being in the way the probabilities are used.

2.3 Two Aliens, Two Crew

2.3.1 Bot 6

In terms of the probabilities updates, this bot updates its beliefs for the alien and crew probabilities exactly like Bot 3. This means that the bot continues to run after capturing the first crew member, and the crew probabilities for all the other open cells are normalized to adjust for the rescue of this crew member. It is also important to note that while there are now two aliens, this is not truly accounted for in how the probabilities are kept track of or updated.

2.3.2 Bot 7

For this bot, a 2D matrix was used to keep track of the crew probabilities, similar to that used in Bot 4. However, given that there are now two aliens, a similar 2D matrix was also used for the alien probabilities. This means that this matrix represents all the possible pairs of cells from the open cells and has a size of (number of open cells) * (number of open cells). Similar to the crew

2D matrix, a dictionary was used to map each cell to an "index value," which would then be used to look up the crew probability for a certain pair of cells. The initial alien probabilities are based on the fact there can be no alien member in the bot's initial cell. Consequently, the probability of all pairs that include the bot's cell is 0. The probability of the two aliens being in the pair is then distributed uniformly among the remaining pairs. The updates to the alien beliefs are also different.

One of the times the bot updates its alien beliefs is when the alien detector is run. Based on whether the alien detector detects an alien or not, the probabilities can be formally represented as P(alien in cell m and j — detect from cell i) and P(alien in cell m and j — detect from cell i), where i is the bot's cell. When the alien detector detects an alien, the alien probabilities are updated based on the expansion of the above probability as P(alien in cell m and j — detect from cell i) = P(alien in cell m and j) P(detect from cell i — alien in cell m and j) / $\sum_{(cella,b)pairs}$ P(alien in cell a and b)* P(detect from cell i — crew in cell a and b). All pairs of cells that have both cells that are outside the detection square will have probability 0 because of the conditional probability P(detect from i — alien in cell m and j). This means that the alien probability of all other remaining pairs of cells will be updated by dividing by the summation of all alien probabilities of pairs that have at least one cell in the detection square. The same approach is used to update the alien probabilities for when no alien is detected as the probabilities expand similarly. The only difference is that the conditional probability in the expansion is different as it is P(no detect from cell i — alien in cell m and j). As a result, the probabilities of the pairs of cells that at least one cell in the detection square will have probability 0. The probability of all the other remaining pairs of cells will be updated by dividing by the summation of all alien probabilities of pairs that have no cells in the detection square. Another time the bot updates its alien beliefs is when it moves. In the case that there is no alien in the cell that the bot moved into, the probability can be formally represented as P(alien in cell m and j — alien not in cell i) for all pairs of cells (m, j) where i is the cell the bot moved into. The probability can be expanded as P(alien in cell m and j — alien not in cell i) = P(alien in cell m and j) P(alien not in cell i — alien in cell m and j) / $\sum_{(cella,b)pairs}$ P(alien in cell a and b) P(alien not in cell i — alien in cell a and b). For pairs of cells that include cell i, the alien probability for that pair is 0. Alternatively, all the pairs of cells that do not have cell i as one of the cells are updated by dividing by the summation of all alien probabilities of pairs that do not have cell i as one of the cells. The bot also updates beliefs when the aliens move. The probability can be formally represented as P(alien in m and alien in j at time t+1). This probability can be expanded as follows: $\sum_{(x,y)}$ P(alien in cell x and alien in cell y at time t) * P(alien in m and alien in j at time t+1 — alien in cell x and alien in cell y at time t) where (x,y) is all pairs of cells. Essentially, there are two possibilities here: the alien in cell x can move to cell m, and the alien in cell y can move to cell j, or the alien in cell x can move to cell j, and the alien in cell y can move to cell m. Evidently, pairs of cells where both cells are not neighbors of either cell x or cell y will have probability 0 as no alien will be able to move there in 1 time-step. For the remaining pairs of cells, the probability is updated by the summation of two separate products. The first product is 1/(number of neighbors of cell x) * 1/(number of neighbors of cell y) if cell m is a neighbor of cell x and cell j is a neighbor of cell y. The second product is 1/(number of neighbors of cell x) *

1/(number of neighbors of cell y) if cell j is a neighbor of cell x and cell m is a neighbor of cell y.

2.3.3 Bot 8

The probability updates for Bot 8 are exactly the same as Bot 7, with the only difference being in the way the probabilities are used.

3 Performance Evaluation

3.1 Testing Setup

All the tests we ran on each of the following metrics were on the following parameters and ranges:

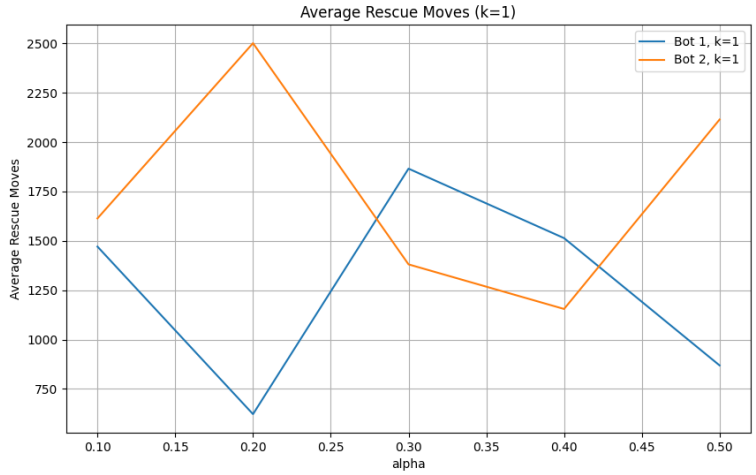
- alpha_values = [0.1, 0.2, 0.3, 0.4, 0.5]
- k_values = [1, 3, 5]
- max_iter = 30
- timeout = 10000

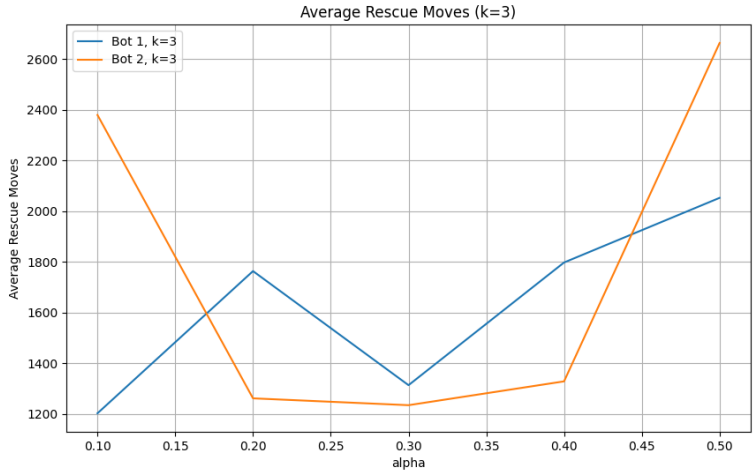
We tested out Bots 1 vs. 2, Bots 3 vs. 4 vs. 5, and Bots 6 vs. 7 vs. 8.
Due to space constraints, here were some of the general findings:

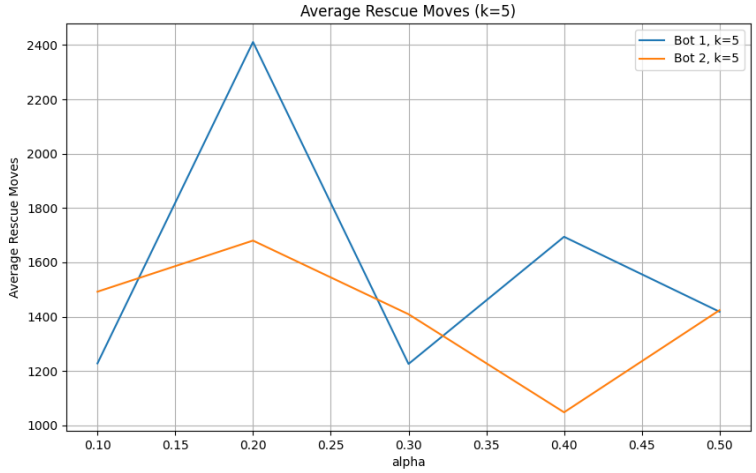
- Typically, a higher α-value resulted in a more significant number of average moves taken by each bot to find crew members.
- Correspondingly, a lower α-value resulted in quicker crew finding but often resulted in a lower crew saving rate (possibly because of risk-seeking behavior in bots).
- A high k-value resulted in higher survival probability (due to the bots being able to predict alien behavior from farther away), while a lower k-value resulted in a lower survival rate.
- However, a high k-value also caused the bots to take far longer to save crew members, possibly due to the way priorities were set up in most bots (i.e., avoid aliens at the cost of going farther away from crew members).
- Bot 2 generally performed better than Bot 1 at saving crew members in fewer moves, but it was ambiguous whether or not it had a higher probability of survival.
- Bot 5 performed around the same as Bot 4 on every criterion, except it was much faster at seeking out crew members (due to its priorities). It consistently performed better than Bot 3 on every metric.
- Bot 8 significantly outperformed Bot 7 and 6 on every metric except the number of moves. It was much slower than both of them, but it is not entirely clear why.

3.2 Average Moves to Save All Crew

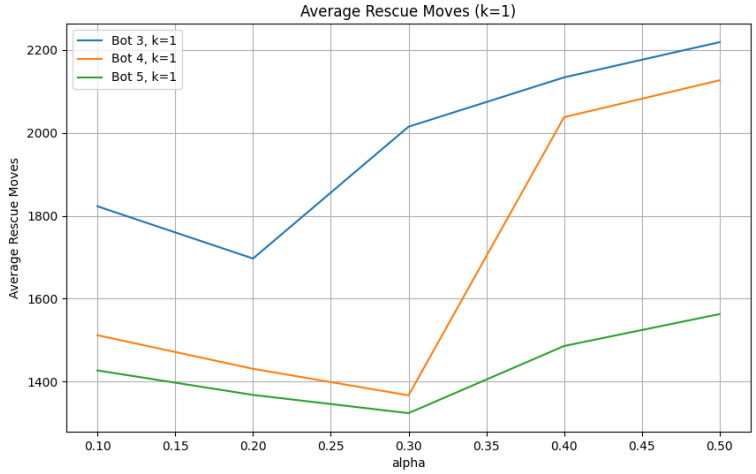
- Bot 1 vs. Bot 2

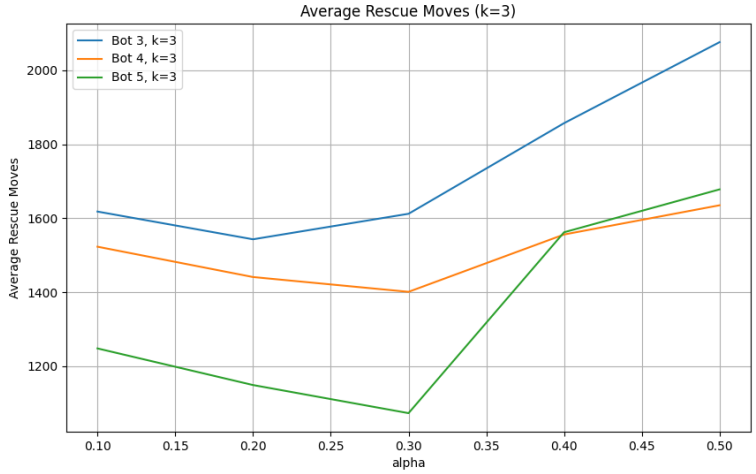


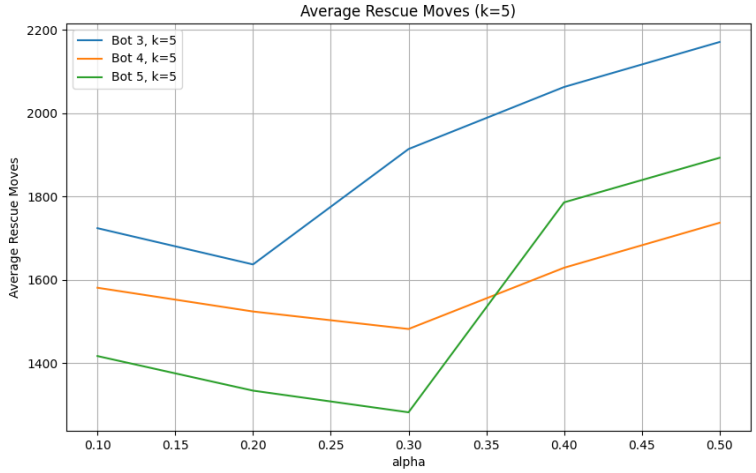




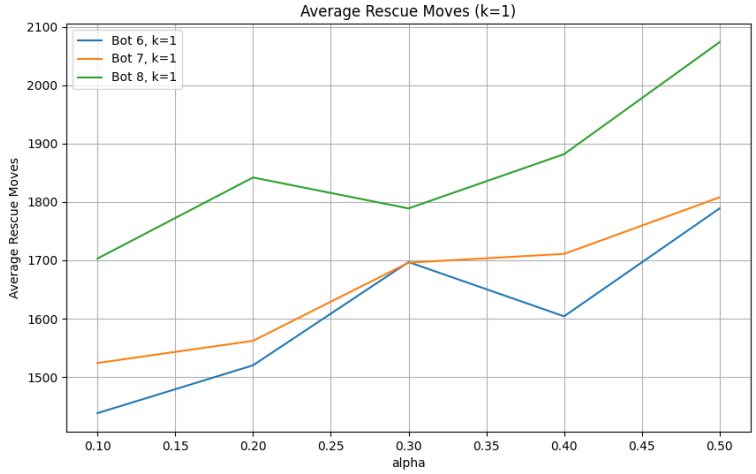
- Bot 3 vs. Bot 4 vs. Bot 5

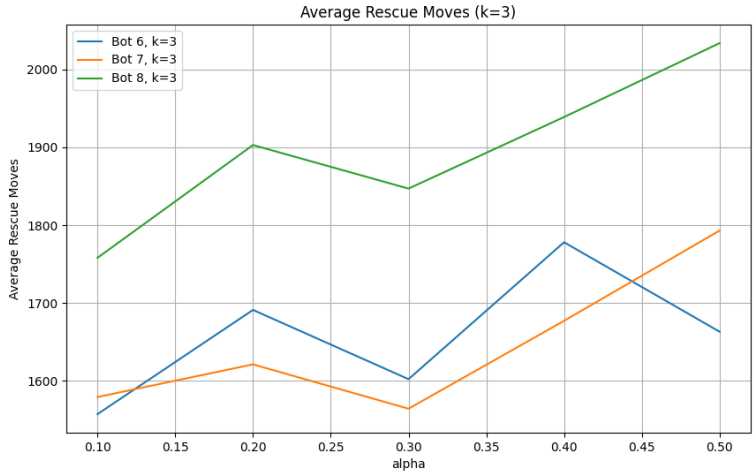


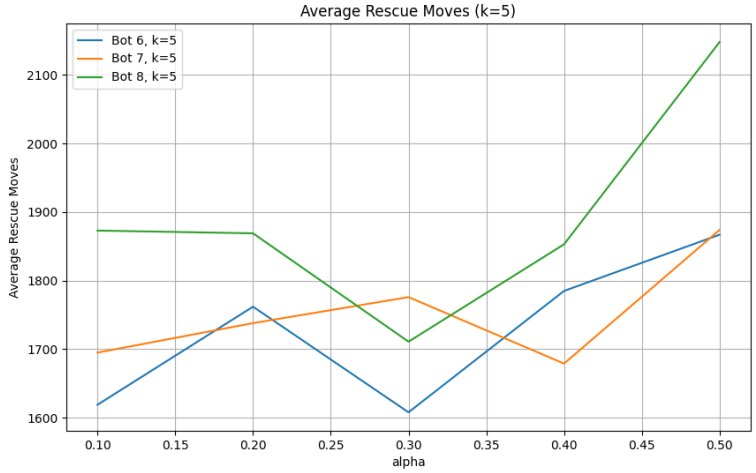




- Bot 6 vs. Bot 7 vs. Bot 8

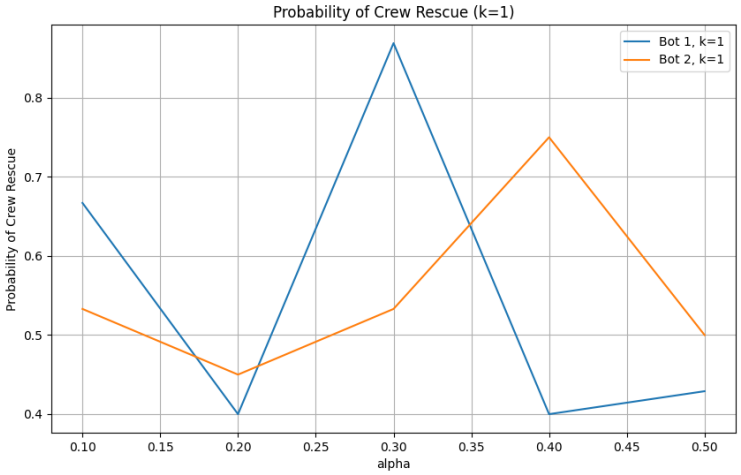


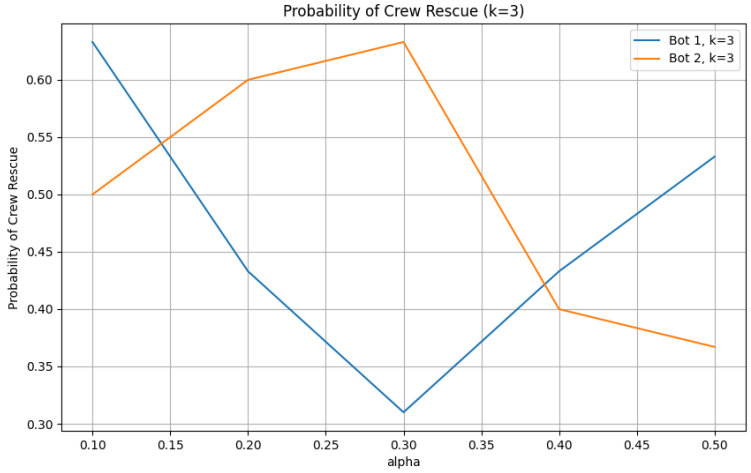


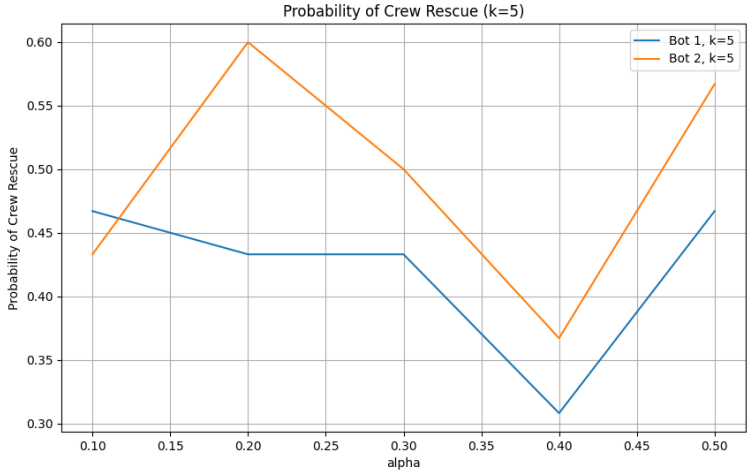


3.3 Probability of Avoiding Alien and Saving Crew

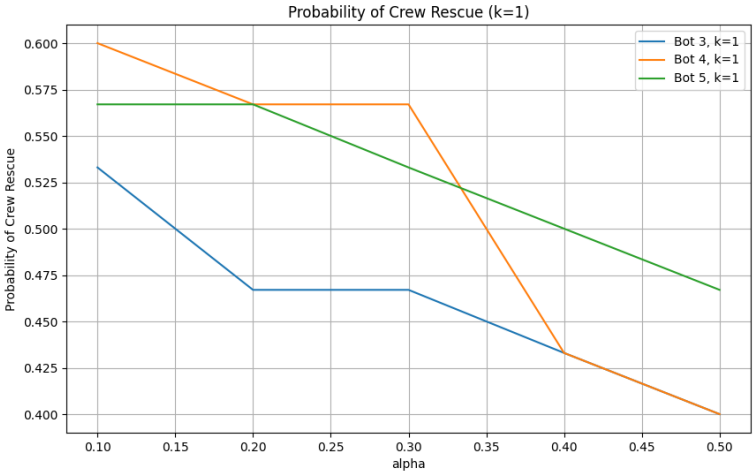
- Bot 1 vs. Bot 2

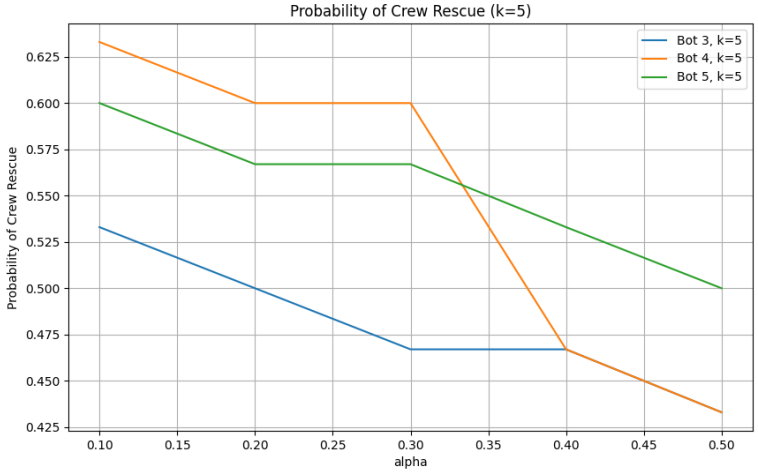


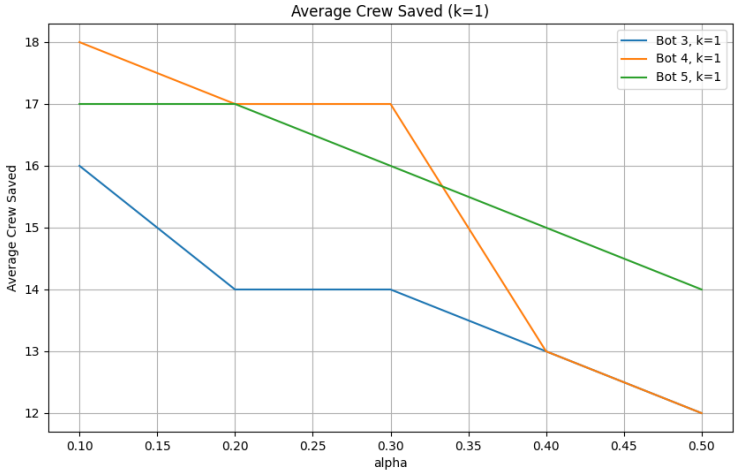




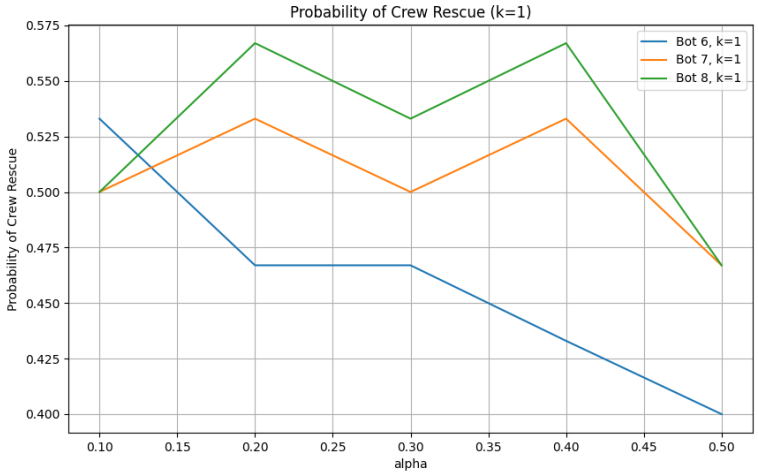
- Bot 3 vs. Bot 4 vs. Bot 5

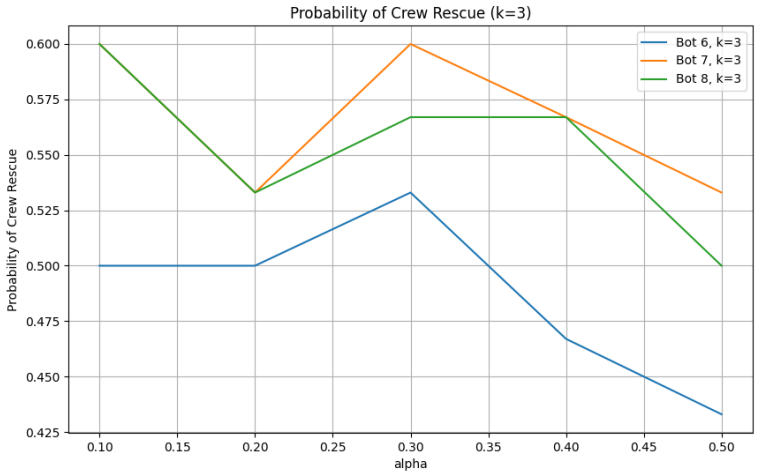


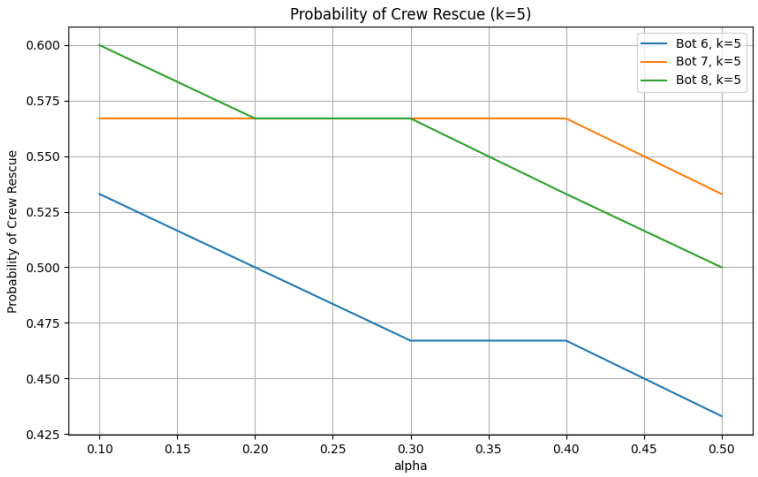




- Bot 6 vs. Bot 7 vs. Bot 8

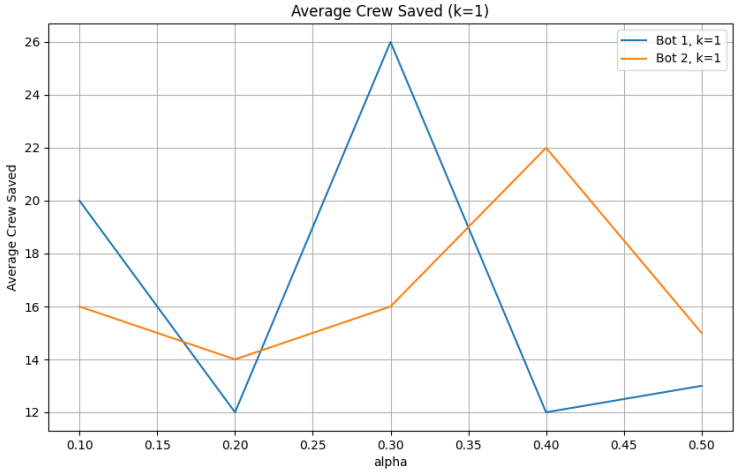


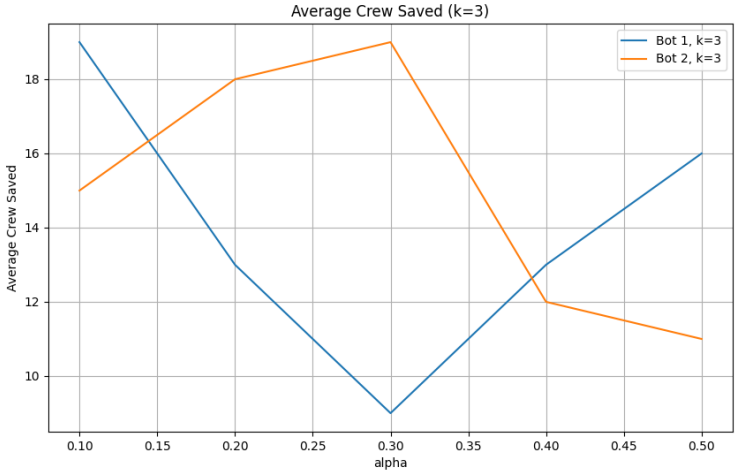


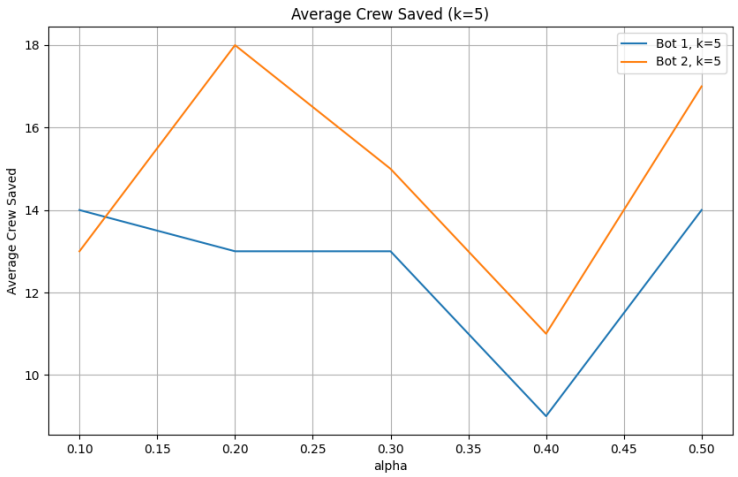


3.4 Average Number of Crew Saved

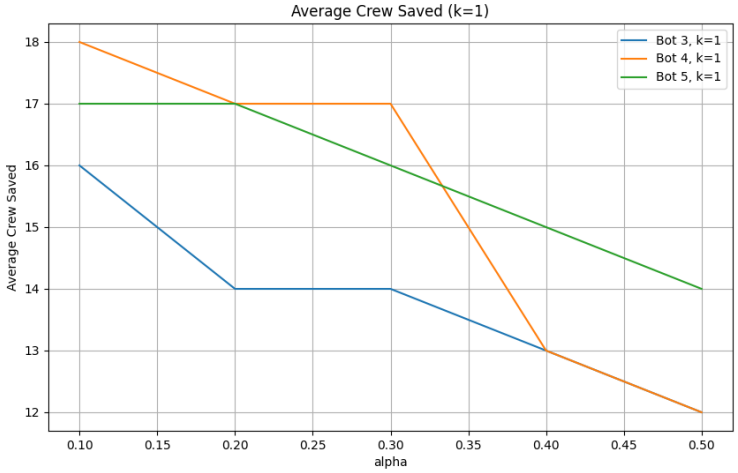
- Bot 1 vs. Bot 2

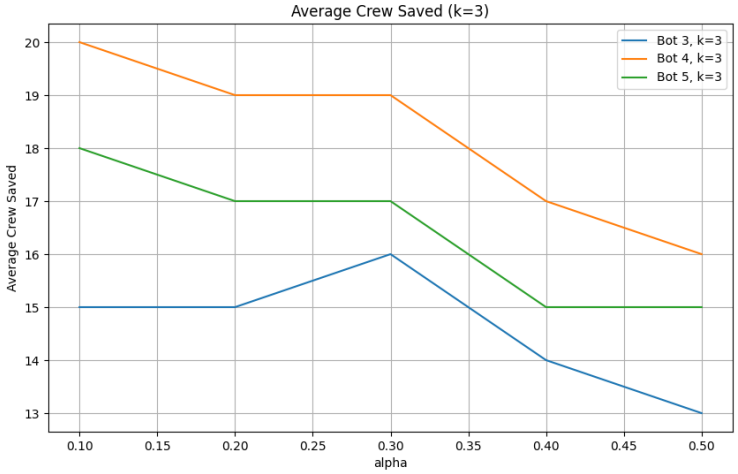


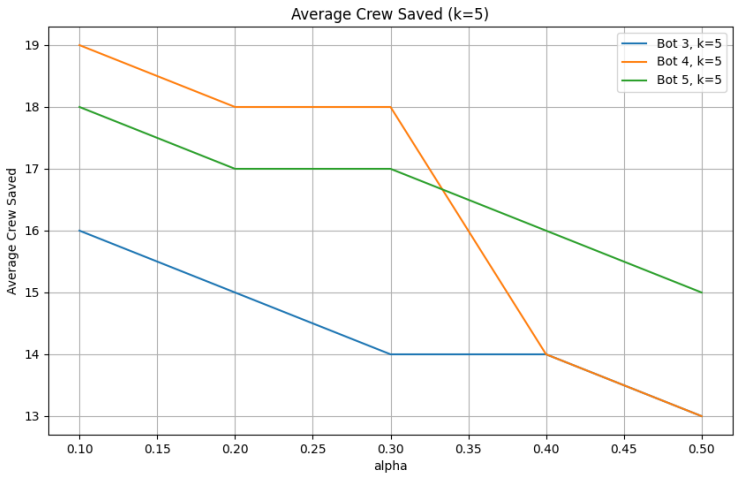




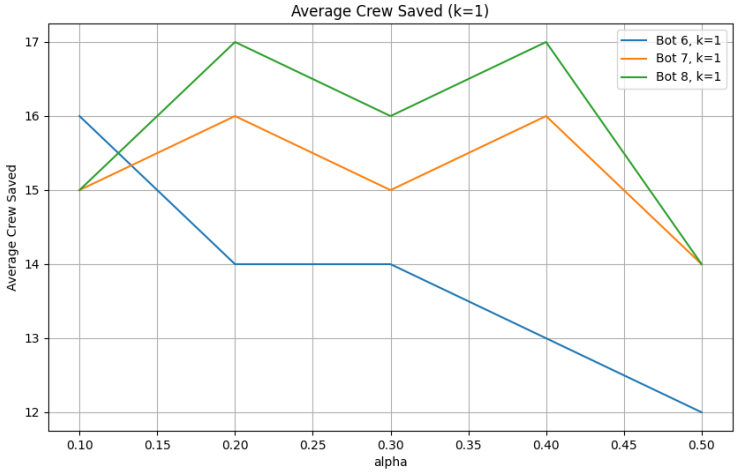
- Bot 3 vs. Bot 4 vs. Bot 5

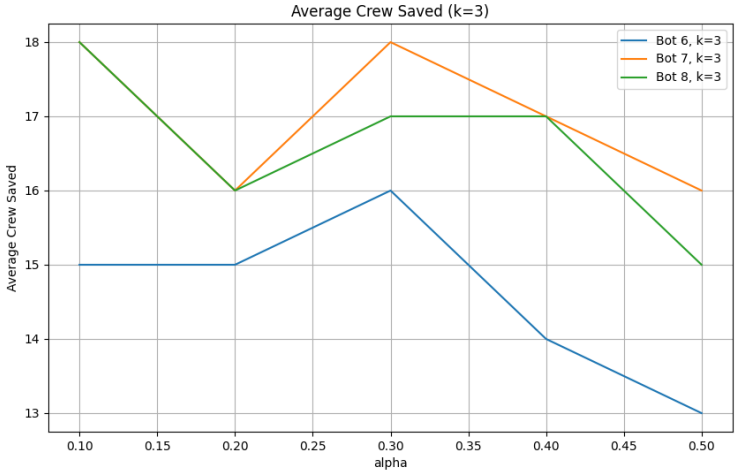


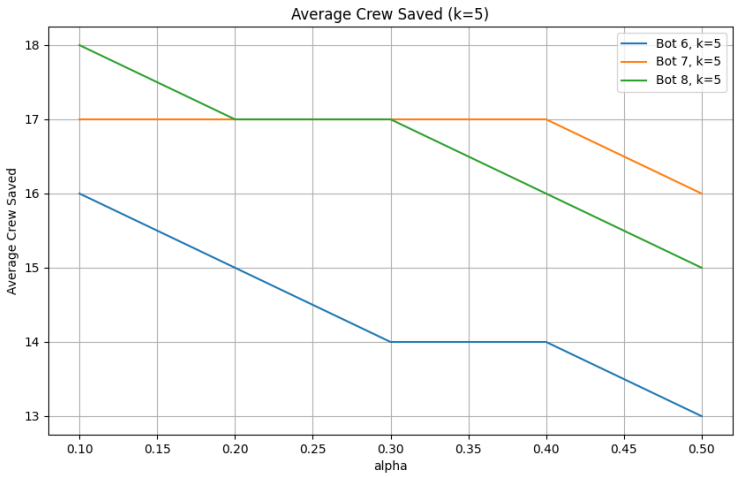




- Bot 6 vs. Bot 7 vs. Bot 8


Average Crew Saved (k=1)
Bot 6, k=1
Bot 7, k=1
Bot 8, k=1
Average Crew Saved
alpha
17
16
15
14
13
12
0.10
0.15
0.20
0.25
0.30
0.35
0.40
0.45
0.50


Average Crew Saved (k=3)
Bot 6, k=3
Bot 7, k=3
Bot 8, k=3
Average Crew Saved
alpha
18
17
16
15
14
13
0.10
0.15
0.20
0.25
0.30
0.35
0.40
0.45
0.50


Average Crew Saved (k=5)
Bot 6, k=5
Bot 7, k=5
Bot 8, k=5
Average Crew Saved
alpha
18
17
16
15
14
13
0.10
0.15
0.20
0.25
0.30
0.35
0.40
0.45
0.50

4 The Ideal Bot

To recap, the information that the bot has is the probabilities of where the alien is in the ship for each cell and the probabilities of where the crew is in the ship for each cell. An possible approach an ideal bot could take is utilize deterministic information with this probabilistic information. For example, the bot could determine the shortest path to the to the crew initially using BFS. As a result, the bot would now have knowledge of what cells to go to. However, since the environment is changing, the bot can use the probabilistic information introduced in this project, particularly the probabilities of where the alien is, to try to become aware of the alien's position and try to avoid it. Using this information, the bot can deviate from its calculated path as necessary. The reason why this bot could be considered ideal is because it would be efficient. For example, it would have more direction than the bots in this project in terms of what cells to go to due to the path planning in the beginning. In addition, it would be more efficient than the bots in project 1 as it would not have to recalculate its path each time step and could update its path based on the probabilities.