

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 8 possible scenarios involving different combinations of bots, crew members, and aliens. Bots 1 and 2 involve 1 bot, 1 crew member, and 1 alien; Bots 3, 4, and 5 involve 1 bot, 2 crew members, and 1 alien; and Bots 6, 7, and 8 involve 1 bot, 2 crew members, and 2 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 the sake of conciseness, I will only delve into the key changes made to the other bots.

1.2 One Alien, One Crew

As mentioned earlier, Bot 1 and Bot 2 both involve 1 bot, 1 alien, and 1 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 single time step to figure out if there is a crew member or alien 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 certain 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 $(2k + 1) \times (2k + 1)$ 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 that are unlikely to contain a crew member at all (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 is reinitialized (including the alien and crew matrices).
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 at this point, 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 is reinitialized (including the alien and crew matrices).
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. After the bot and alien move, the alien sensor is activated. 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 $\text{alien-x} < \text{bot-x} + k$ and $\text{alien-x} > \text{bot-x} - k$ and $\text{alien-y} < \text{bot-y} + k$ and $\text{alien-y} > \text{bot-y} - k$) 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 α , 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 for 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 more likely 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 exact 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.

1.3 One Alien, Two Crew

1.3.1 Bot 3

1.3.2 Bot 4

1.3.3 Bot 5

1.4 Two Aliens, Two Crew

1.4.1 Bot 6

1.4.2 Bot 7

1.4.3 Bot 8

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 . To start, it is important to note that the bot has uniform probability of the crew being in any open cell but the current one it is in and uniform probability of the alien being in any of the cells outside of 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 detector comes back to give a positive or negative response, the probability of an alien being in a certain cell is only based on that when doing this update. These probabilities can be written out as $P(\text{alien in } k \text{ — detect from } i)$ and $P(\text{alien in } k \text{ — no detect from } i)$ 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 $P(\text{alien in } k \text{ — detect from } i) = P(\text{alien in } k) * P(\text{detect from } i \text{ — alien in } k) \sum_{j \in \text{cells}}$

$P(\text{detect in } i \text{ and alien in } j)$. Here, $P(\text{alien in } k)$ is our prior belief. Evidently, if it is a positive detect, 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 can be set as such for these cells as the probability $P(\text{detect from } i \text{ — alien in } k)$ is 0 for them. However, $P(\text{detect from } i \text{ — alien in } k)$ 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 $P(\text{alien in } k) / \sum_{j \text{ cells}} P(\text{detect in } i \text{ and alien in } j)$. The denominator is marginalized over all detection square cells in order to normalize $P(\text{alien in } k)$ since the bot does not know which cell in the detection square has the alien if the detector is positive. Alternatively, if the detector returns negative, the probability to compute is $P(\text{alien in } k \text{ — no detect from } i)$. This is expanded similarly as the other probability and is computed similarly as a result. The stark difference is that now cells inside the detection square have probability 0 and cells outside the square are updated by $P(\text{alien in } k) / \sum_{j \text{ cells}} P(\text{no detect in } i \text{ and alien in } k)$ 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 α is user-specified that the sensor gives a beep. The probabilities for this case can be formally represented as $P(\text{crew in } k \text{ — beep in cell } i)$ and $P(\text{crew in } k \text{ — no beep in } i)$, 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 $P(\text{crew in } k \text{ — beep in cell } i)$ where it is $P(\text{crew in } k) * P(\text{beep in cell } i \text{ — crew in } k) / \sum_{\text{all } j \text{ cells}} P(\text{crew in } j) * P(\text{beep in } i \text{ — crew in } j)$ where j is all open cells. In essence, $P(\text{crew in } k)$ here is the prior probability of a crew being in cell k and $P(\text{beep in } i \text{ — crew in } j)$ 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 $P(\text{beep in cell } i \text{ — crew in } k)$, 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 $P(\text{no beep in } i \text{ — crew in } j)$, which is just $1 - e^{(-\alpha * (d-1))}$.

Another time the bot updates beliefs is when the bot moves to a new cell. This probability can be formally represented as $P(\text{alien in } j \text{ — alien not in } i)$ and $P(\text{crew in } j \text{ — crew not in } i)$ 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 $P(\text{alien in } j \text{ — alien not in } i) = P(\text{alien in } j) * P(\text{alien not in } i \text{ — alien in } j) / P(\text{alien not in } i)$. Here $P(\text{alien not in } i \text{ — alien in } j)$ is 1 for all other cells except for when cell j is the same as cell i , and $P(\text{alien not in } i)$ is just $1 - \text{the prior probability of the alien being in cell } i$ or $(1 - P(\text{alien in } i))$. 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 $P(\text{crew in } j) * P(\text{crew not in } i \text{ — crew in } j) / P(\text{crew not in } i)$. 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 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 $P(\text{alien is now in cell } j \text{ at time } t+1)$, which expands into $\sum_{k \in \text{cells}} P(\text{alien was in cell } k \text{ at time } t) * P(\text{alien now in cell } j \text{ at time } t+1 \mid \text{alien was in cell } k \text{ at time } t)$ 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 , $P(\text{alien now in cell } j \text{ at time } t+1 \mid \text{alien was in cell } k \text{ at time } t)$ is $1/(\text{number of neighbors of } j + 1)$ 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

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 $1/(\text{number of open cells} - 1)$.

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 $(\text{number of open cells}) * (\text{number of open cells})$. In order to index this matrix efficiently, another dictionary was used to map each cell to an "index value", which then would be used for looking 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 is 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 $P(\text{crew in cell } k \text{ and cell } j \text{ — beep from cell } i)$ and $P(\text{crew in cell } k \text{ and cell } j \text{ — no beep from cell } i)$. 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 $P(\text{crew in cell } k \text{ and cell } m \text{ — beep from cell } i) = P(\text{crew in cell } k \text{ and cell } m) P(\text{beep from cell } i \text{ — crew in cell } k \text{ and cell } m) / \sum_{(cella,b) \text{ pairs}} P(\text{crew in cell } a \text{ and } b) P(\text{beep from cell } i \text{ — crew in cell } a \text{ and } b)$ where (a,b) is all pairs of cells. The summation in the denominator is over all pairs of cells. The probabilities $P(\text{crew in } k \text{ and cell } m)$ and $P(\text{crew in cell } a \text{ and } b)$ 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 in distinguishing this. As a result, this conditional probability can be considered as $1 - P(\text{no beep from the crew members})$. Subsequently, $P(\text{no beep from the crew members})$ can be broken down into independent events of both the crew not sending a beep: $P(\text{no beep from crew at cell } k) * P(\text{no beep from crew at cell } m)$. 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 $P(\text{beep from cell } i \text{ — crew in cell } k \text{ and cell } m)$ 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 $P(\text{no beep from cell } i \text{ — crew in cell } k \text{ and cell } m)$, which would break down as $P(\text{no beep from crew } k) * P(\text{no beep from crew } m)$ 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 $P(\text{crew in cell } k \text{ and } j \text{ — crew not in cell } i)$ for all pairs of cells (k, j) where i is the cell the bot moved into. The probability can be expanded into $P(\text{crew in cell } k \text{ and } j \text{ — crew not in cell } i) = P(\text{crew in cell } k \text{ and } j) P(\text{crew not in cell } i \text{ — crew in cell } k \text{ and } j) / \sum_{(cella,b) \text{ pairs}} P(\text{crew in cell } a \text{ and } b) P(\text{crew not in cell } i \text{ — crew in cell } a \text{ and } b)$. 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

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 is now 2 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 then would be used for looking 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 $P(\text{alien in cell } m \text{ and } j \text{ — detect from cell } i)$ and $P(\text{alien in cell } m \text{ and } j \text{ — no detect from cell } i)$ 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 $P(\text{alien in cell } m \text{ and } j \text{ — detect from cell } i) = P(\text{alien in cell } m \text{ and } j) P(\text{detect from cell } i \text{ — alien in cell } m \text{ and } j) / \sum_{(cell a, b) \text{ pairs}} P(\text{alien in cell } a \text{ and } b) * P(\text{detect from cell } i \text{ — crew in cell } a \text{ and } b)$. All pairs of cells that have both cells that is outside the detection square will have probability 0 because of the conditional probability $P(\text{detect from } i \text{ — alien in cell } m \text{ and } j)$. 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 $P(\text{no detect from cell } i \text{ — alien in cell } m \text{ and } j)$. 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 the bot moves. In the case that there is no alien in the cell that the bot moved into, the probability can be formally represented as $P(\text{alien in cell } m \text{ and } j \text{ — alien not in cell } i)$ for all pairs of cells (m, j) where i is the cell the bot

moved into. The probability can be expanded as $P(\text{alien in cell } m \text{ and } j \text{ — alien not in cell } i) = P(\text{alien in cell } m \text{ and } j) P(\text{alien not in cell } i \text{ — alien in cell } m \text{ and } j) / \sum_{(cell a, b) \text{ pairs}} P(\text{alien in cell } a \text{ and } b) P(\text{alien not in cell } i \text{ — alien in cell } a \text{ and } b)$. 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 $P(\text{alien in } m \text{ and alien in } j \text{ at time } t+1)$. This probability can be expanded as follows: $\sum_{(x,y)} P(\text{alien in cell } x \text{ and alien in cell } y \text{ at time } t) * P(\text{alien in } m \text{ and alien in } j \text{ at time } t+1 \text{ — alien in cell } x \text{ and alien in cell } y \text{ at time } t)$ where (x,y) is all pairs of cells. Essentially, there are two possibilities here as 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 a neighbor 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 $1/(\text{number of neighbors of cell } x) * 1/(\text{number of neighbors of cell } y)$ if cell m is a neighbor of cell x and cell j is a neighbor of cell y . The second product is $1/(\text{number of neighbors of cell } x) * 1/(\text{number of neighbors of cell } y)$ if cell j is a neighbor of cell x and cell m is a neighbor of cell y .

2.3.3 Bot 8

3 Performance Evaluation

3.1 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

3.2 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

3.3 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

4 The Ideal Bot

5 Bonus

6 Final Thoughts and Relevant Information