CS 520

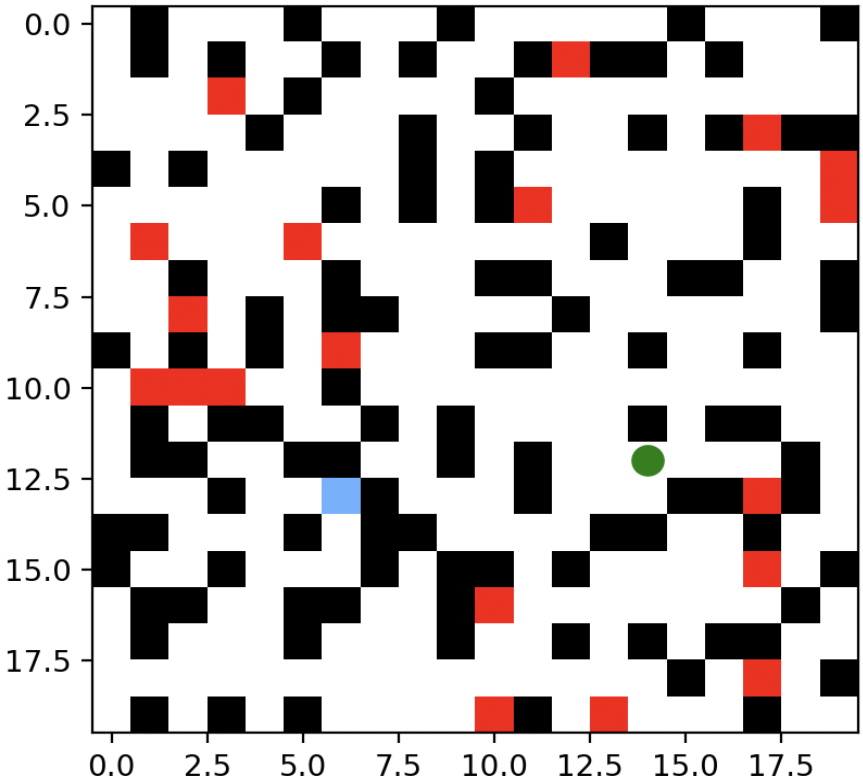
INTRO TO ARTIFICIAL INTELLIGENCE

Project 1 Report: Your Friend is in Danger and You are a Brave Bot

- Deviram Kondaveti (dk1273)
- Vaishnavi Madhavaram (vm695)
- Puja Sridhar (ps1393)

Design Choices

The first step in visualizing the problem was to set up the environment that is generating the ship of size D and then representing the closed cells, open cells, aliens, Bot and Captain. The values used to represent each element of the environment are:



Implementation of Bots:

In pathfinding scenarios like the one presented in the project, the choice of search algorithm is critical for optimizing the bot's decision-making process. Depth-First Search (DFS) and Breadth-First Search (BFS), while simple and easy to implement, lack the optimality and efficiency required for this dynamic environment. DFS may get stuck in deep branches, failing to find optimal solutions, while BFS guarantees optimality but can be memory-intensive. A* strikes a balance between time and space complexity, utilizing a consistent heuristic to guide the search efficiently. This informed search algorithm ensures not only the discovery of the

optimal path but also considers the cost of reaching the goal, making it a suitable choice for navigating through the complex and changing ship environment, avoiding aliens, and successfully rescuing the captain. All four bots are implemented using A* algorithm to find the path from the bot to the captain.

Heuristic:

Here h(child) is the heuristic value calculated from the child to the goal state. In our case we are considering the heuristic value for the base model as the Manhattan distance between the child and the captain (since our bot moves only in up/down/left/right directions).

If $child = (c_1, c_2)$ and $goal = (g_1, g_2)$ then

$$heuristic(child, goal) = abs(c_1 - g_1) + abs(c_2 - g_2)$$

A* Algorithm Implementation

Fringe ← Priority Queue

Dist ← HashMap()

Prev ← HashMap()

While Fringe not empty:

    Curr ← Fringe.pop()

    If curr is a goal node:

        Return SUCCESS, Dist, Prev

    For child of curr:

        temp_dist = Dist[curr] + cost(curr,child)

        if child ∄ Dist or temp_dist < Dist[child]

            Dist[child] = temp_dist; Prev[child] = curr

            Priority = Dist[child] + h(child)

            Fringe.add or update(child.priority)

Return FAILURE, Dist, Prev

Bot 1:

The Bot 1 finds an optimal path from the initial position to the captain by considering the initial ship configuration and the initial alien positions. So it doesn't consider the alien movement. The bot finds an optimal path using A * algorithm and follows the same path until it reaches the captain or gets attacked by any other alien.

Bot 2:

The Bot 2 finds an optimal path from the initial position to the captain before every step it takes. Unlike the Bot 1, it also considers the current alien position on the ship and re-computes the path to the captain. At every step, the bot runs an A* star algorithm to get the path from current position to captain and takes the next step according to it.

Bot 3:

The Bot 3 takes the future steps of the aliens into consideration. It considers the neighbouring cells of the aliens also as the blocked states and computes a path using A* algorithm on this new ship configuration.

Bot 4:

Design Choice

Idea:

Unlike the previous bot simulations, Bot 4 incorporates a dynamic obstacle avoidance mechanism by identifying the nearby aliens and moving away from it. This ensures that the bot avoids getting too close to aliens, minimizing the risk of encountering them during its movement. It calculates the optimal path using an A* star algorithm similar to the Bot 3 but with an improved heuristic estimation. By using this new heuristic, the bot chooses the path by avoiding the aliens that are closer to it and increase its survivability. During each iteration, the bot makes informed decisions to move towards the goal, adapting its route if needed based on the dynamically changing environment. This strategy enhances the bot's survivability and success rate in complex ship layouts with varying alien distributions.

Implementation:

Enhancement 1:

The A* algorithm employed in Bot 4 uses an improved heuristic. Bot 4 also considers the aliens closer to its neighbour cell and estimates heuristic value depending on two factors - the distance from the neighbour to the Captain and the distance between the neighbour and nearest alien.

In each iteration of the A* algorithm, along with calculating the Manhattan distance from a neighbour cell to the captain, we also compute the distance from the neighbour cell to each alien's position. The neighbours that are farther from the aliens are given lesser heuristic value than the neighbours that are closer to aliens.

$$heuristic(neighbour, captain) = manhattan\_dist(neighbour, captain) + alien\_proximity\_cost$$

$$alien\_proximity\_cost = \frac{ship\_size}{min\_dist\_to\ alien(neighbour)}$$

If the distance to the alien found is higher, then the *alien_proximity_cost* will be lesser, which may eventually result in lesser heuristic value. Similarly, if the distance is lesser, then the *alien_proximity_cost* will be high, which may eventually result in higher heuristic value. The bot can make a safer decision based on these updated heuristic values.

Enhancement 2:

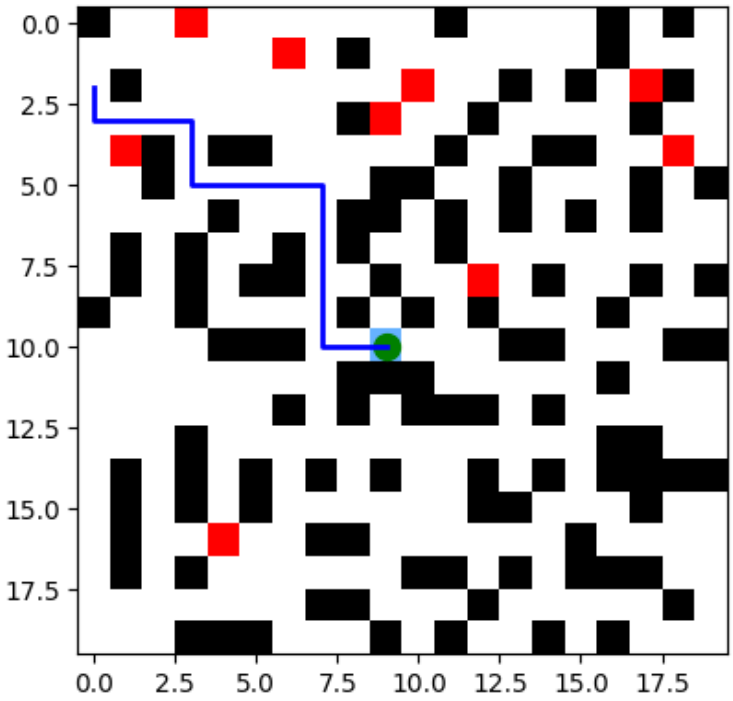
If the A* algorithm finds no path, the bot will stay idle. In our observation, the bot stays for a very long time and eventually gets attacked by the alien. To overcome such scenarios and increase the survivability, the Bot 4 was enhanced. If there is no path found from the A* algorithm and if the Bot 4 stays longer at the same position for longer time, then the Bot 4 moves to a nearby cell.

Data Analysis:

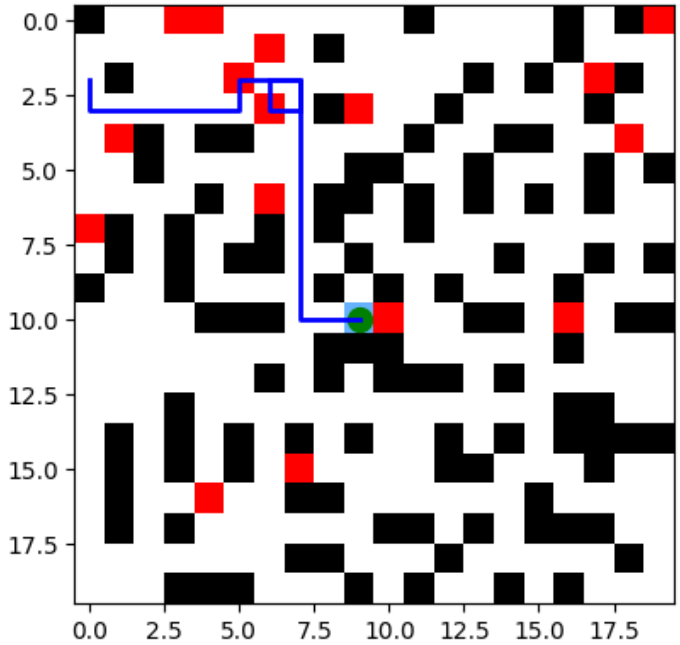
The visual representation of the path taken by each bot in a 20 X 20 ship is shown below.

Legend for the plots –

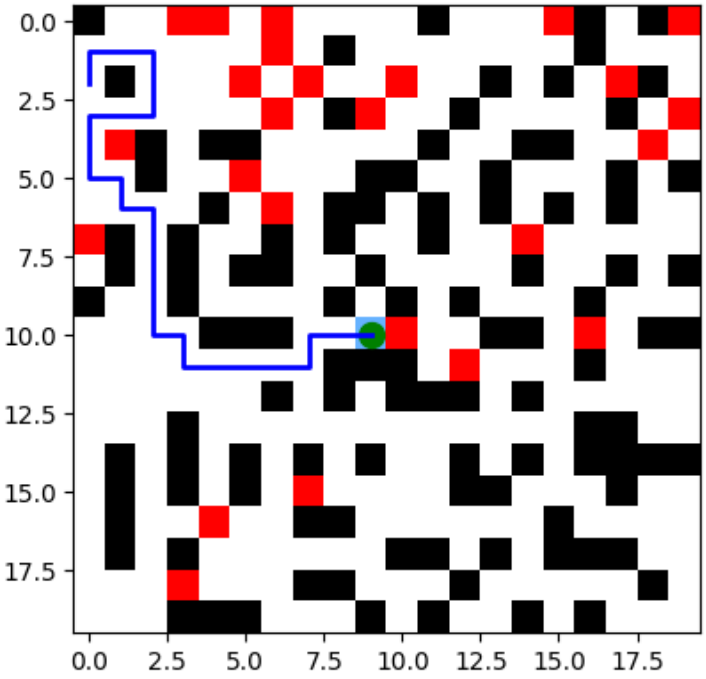
Bot 1 Path:



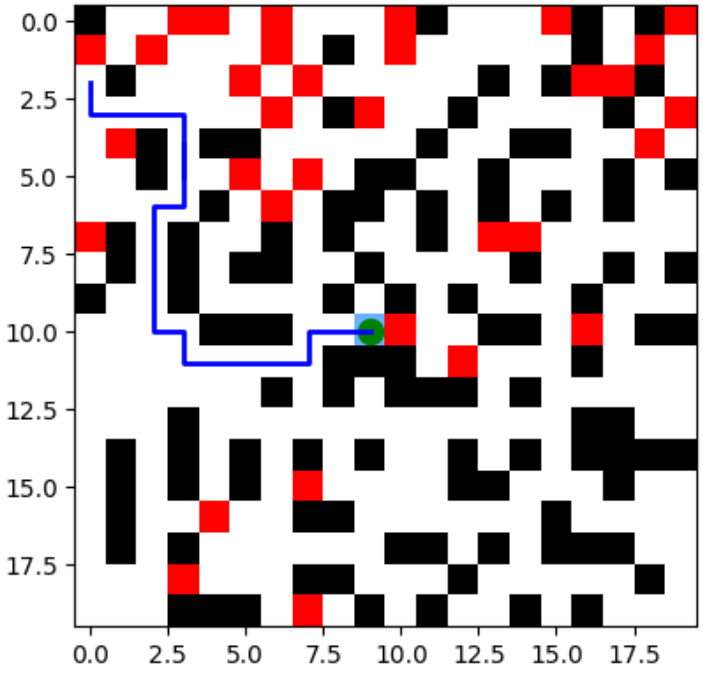
Bot 2 Path:



Bot 3 Path:



Bot 4 Path:



Plotted Data for all Bots:

We have plotted the graph for each bot with the following parameters

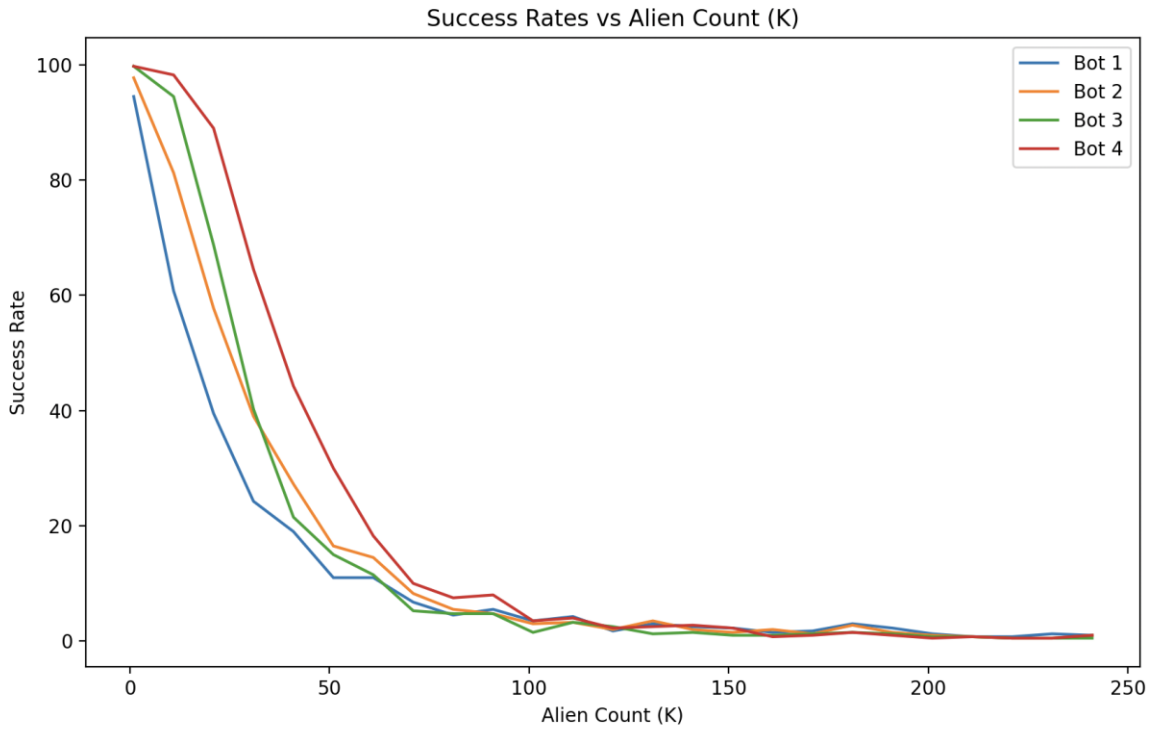
- The success rate as a function of K
- The survival rate among unsuccessful runs as a function of K

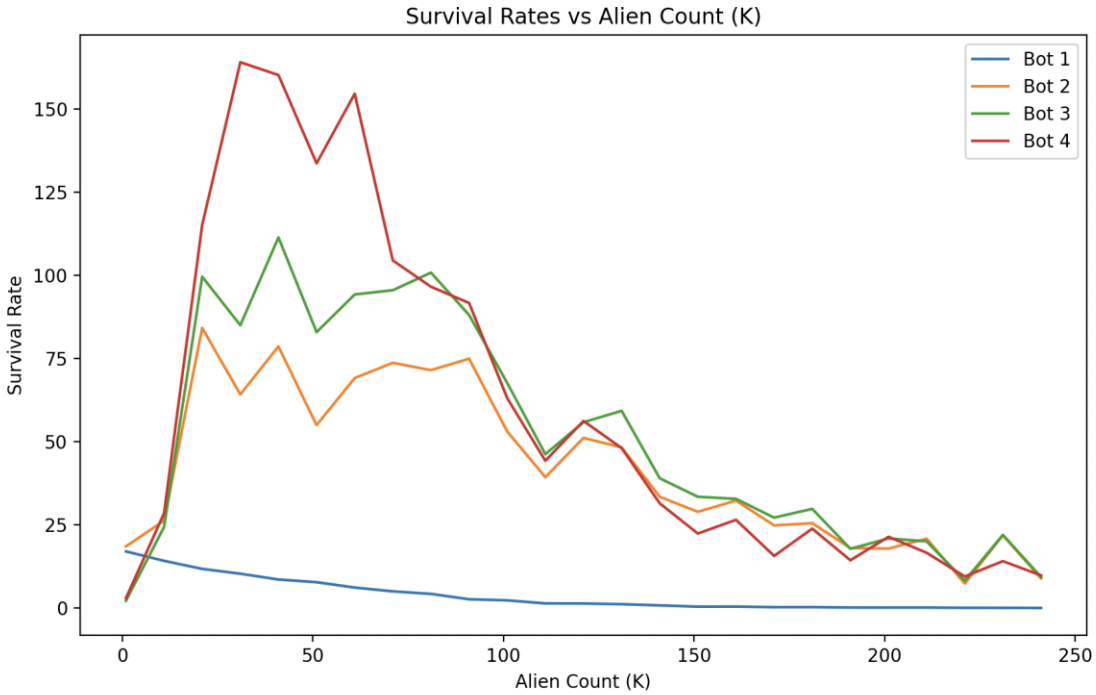
Initially we have tested all the bots for values as follows:

Ship size - 30 x 30

Alien count – 250

Epochs - 400



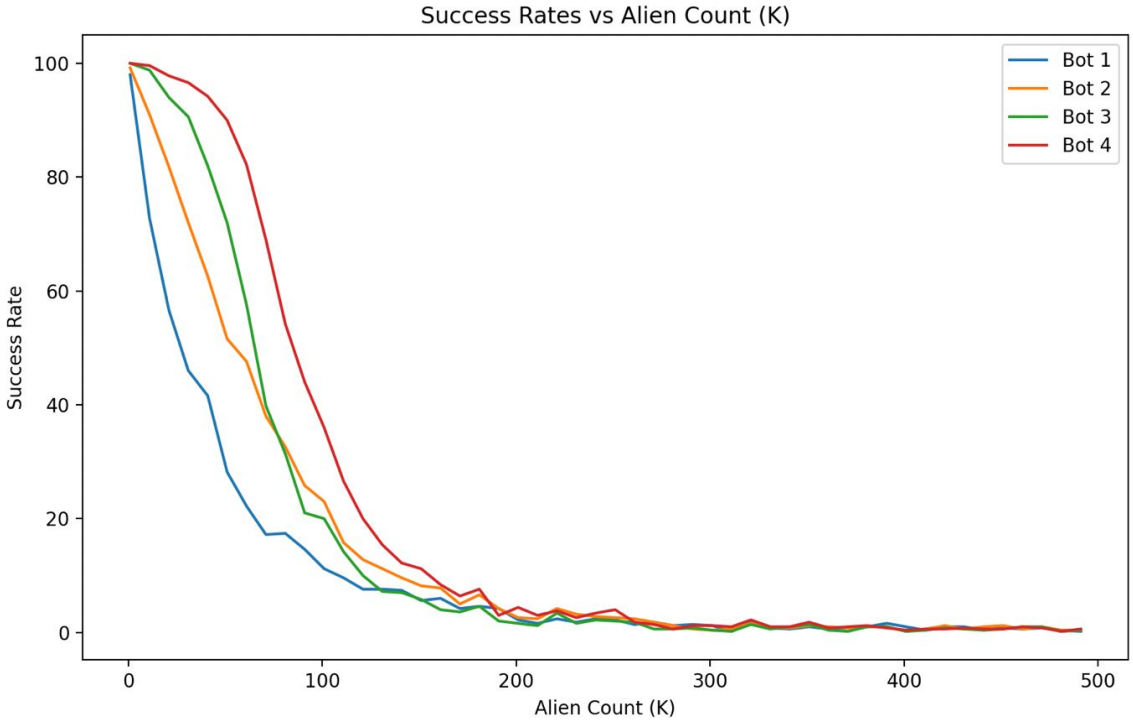


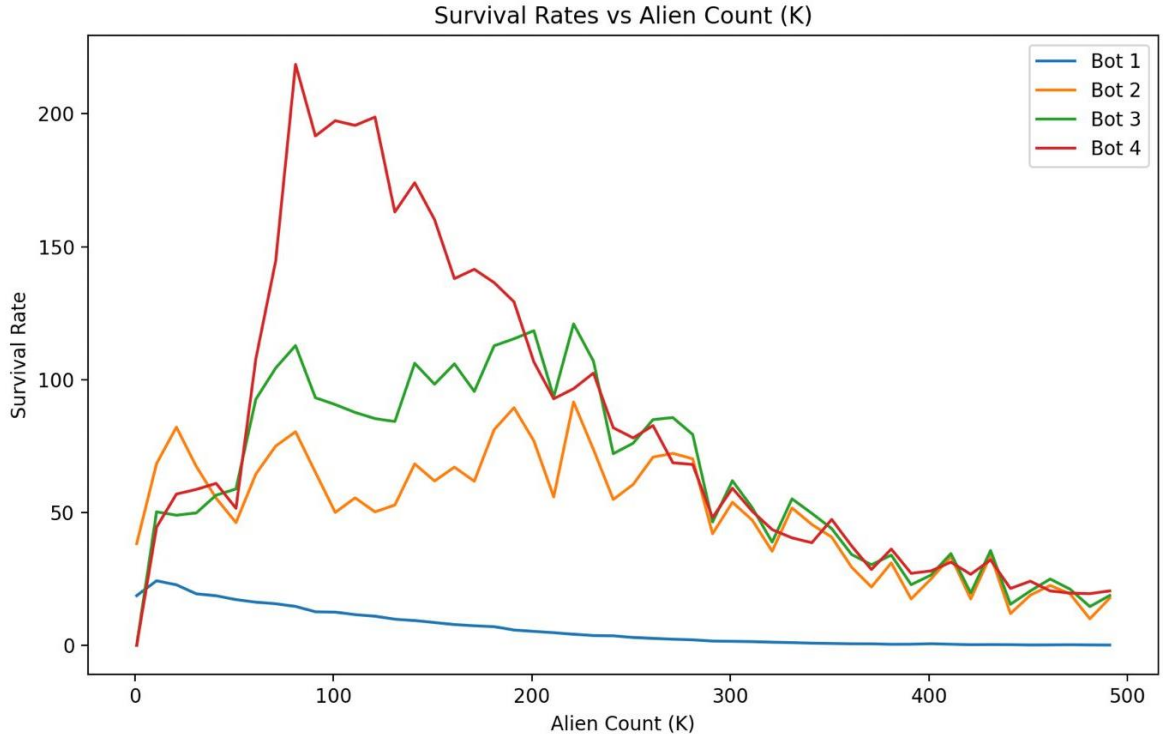
We have also tested all the bots for values as follows:

Ship size - 50 x 50

Alien count – 500

Epochs – 500





While testing with a 50 x 50 ship size with 500 aliens, we collected the data and stored it in a csv file through which we observed that the max aliens count 'K' after which the success rates for each of the bots become zero.

bot	alien_count	success_count	bot_steps
1	11	119	12.86419
1	21	89	10.70270
1	31	50	10.86666
1	41	38	9.154320
1	51	33	8.191616
1	61	16	5.510869
2	11	167	37.66666
2	21	114	57.62790
2	31	85	53.40869
2	41	69	74.16030
2	51	39	73.96273
2	61	21	65.12849
3	11	183	28.09090
3	21	143	58.37209
3	31	80	77.88695
3	41	52	117.5725
3	51	32	98.08695
3	61	14	104.00
4	11	189	70.78787
4	21	163	72.50
4	31	112	103.1478
4	41	88	166.0992
4	51	40	157.3726
4	61	23	115.8491

Fig: The screenshot of the collected data is shown below.

Another observation made was that as the alien's count was increasing the success rate of all the bots was gradually decreasing. The survival rate of the Bot 4 is higher compared to the others as shown in the graphs above, which clearly shows that the main objective of the Bot 4 we implemented is satisfied to some extent. The max alien count 'K' for each of the bots are as follows:

Bot 1 – 331
Bot 2 – 321
Bot 3 – 271
Bot 4 – 231

Bot Performance:

Bot 1 fails because once it plots a path to the captain, it continues to follow that path every step ignoring the alien movements and easily gets caught by the aliens. Bot 1 would be able to avoid easily getting caught and save the captain if it does not ignore the aliens while moving.

Bot 2 corrects the flaw in Bot 1 by plotting a path before each step to factor in current alien positions, but still, it fails in cases where the alien comes into the cell of the bot after the bot has done moving in a step. Bot 2 would do better if it considers future alien positions to plot a path, then it would go on for more timesteps and hence save the captain.

Bot 3 is a more improved version of Bot 2 as it tries to avoid aliens even in the future positions. This makes it last longer than Bot 2 and hence save the captain. But this bot also fails in situations where no valid path without aliens near it is not available. In such cases the bot doesn't move until a path opens up and gets attacked by an alien.

Bot 4 significantly outperforms Bot 3 by making some random movements which gives it a better chance in computing a path, instead of just staying idle at one position. But Bot 4 also fails when the path taken by the Bot is not the shortest path as there is a higher chance of the Bot not reaching the captain as the maximum number of timesteps has been attained. Thereby, leading to a failed mission.

Contributions:

Deviram Kondaveti:

- Worked on Ship layout generation
- Worked on Bot 1 and testing
- Worked on Bot 4-Enhancement 2 and optimising it
- Worked on Project Report

Vaishnavi Madhavaram:

- Worked on the A-star algorithm

- Worked on Bot 3 and testing
- Worked on Bot 4-Enhancement 1 and optimising it
- Worked on Project Report

Puja Sridhar:

- Worked on Bot 2 and testing
- Code optimization and improved efficiency of all the Bots
- Worked on graphs, data generation and analysis of each bot's performance
- Worked on Project Report

All the three of us always discussed the ideation of each bot before implementing it, put forward our opinions and ideas for each bot, mutually decided and then implemented each of the bot.

Each one of us has tested the bots on different ship sizes and alien counts initially to analyse how each bot is working and how it can be optimised.

Each one of us has written the implementation part of the bot they worked on in the report.