CS - 520

Introduction to Artificial Intelligence

Project 2

Group:

1. **Reo Correia** (rc1460)
2. **Manan Davawala** (md1840)
3. **Vijayendra Sai Chennareddy** (vc545)

Problem Statement

Understanding the Problem

The challenges that Bot faces:

- Grid-Based Environment: The ship's layout is represented as a grid, where each cell can be either open or blocked (walls). The bot's movements are constrained to up, down, left, and right, reflecting the grid's structure.
- Hostile Aliens: The presence of hostile aliens poses a significant threat to the bot's mission. Aliens move within the grid, and the bot must avoid collisions with them to ensure the safety of the crew members.
- Crew Member Rescue: The primary objective is to rescue crew members who are randomly placed in open cells. The bot must quickly reach the crew members while adapting to dynamic alien movements
- Bot Knowledge: The bot's knowledge of the alien and crew member locations should be updated based on the sensor data.
- Crew Sensor Detection: When the bot receives a beep, it updates its belief that the crew member is likely to be in cells close to it. If the bot does not receive a beep, it updates its belief that the crew member is less likely to be in cells close to it.
- Alien Sensor Detection: When the bot receives a beep only when the alien is within (2k+1) X (2k+1) square centered at the it's location.
- Probabilistic Approach: Since the bot doesn't know the exact location of the crew and aliens. Bot uses a probabilistic approach to navigate the ship, detect aliens, and find crew members.

Setup

- The grid size of **30 X 30** has been used for the ship layout.
- Alien and Crew Sensor Detection:
  - For a given k, the bot can sense whether there is an alien in the (2k + 1) × (2k + 1) square centered at its location.
  - For a given α > 0, if there is a crewmember hidden d-steps away from the bot, the bot receives a beep with probability e^(−α(d−1))
- Distance between cells:
  - Distance between the cells are ship dependent
  - Calculate distances from cell i to every other cell j using BFS Algorithm.

Knowledge Base

Initial Knowledge Base

We maintain an array called **Belief Network** containing probabilities to have aliens and crew for each cell. This Knowledge Base is updated appropriately at each time step.

The bot knows that the alien is equally likely to be in any open cell outside the bot's detection square and the crew is assumed to be equally likely in any cell except the bot's initial cell.

So, the initialization of the Belief Network is done as follows:
P(alien in cell j) for each cell j = 1 / (No of open cells - 1)
P(crew in cell j) for each cell j = 1 / (No of open cells - 1)

Knowledge Base is updation is necessary:

- If bot moves and enters cell i
- If the bot runs the alien detection and comes back positive
- If the bot runs the alien detection and comes back negative
- If the bot listens for a crew beep, doesn't hear one

- If the bot listens for a crew beep, does hear one
- If alien moves

Bot Modeling and Probability Updation

1) Model 1: Single Alien

Initial Knowledge Base:

P(alien at cell j) for each cell j

Bot is in cell i

DS(i) = {open cells in the detection square centered at i}

Updating Knowledge Base:

I. Alien Detection is Negative

P(alien is in cell j | alien is not in DS(i)) for each cell j
= P(alien is in cell j AND alien is not in DS(i)) / P(alien is not in DS(i))
= P(alien is in cell j) * P(alien is not in DS(i) | alien is in j) / P(alien is not in DS(i)) —> (a)

Now calculate for the denominator,
P(alien is not in DS(i))
= sum_{k} P(alien in cell k AND alien is not in DS(i))
= sum_{k} P(alien in cell k) P(alien is not in DS(i) | alien in cell k)

Substitute in (a):
P(alien is in cell j | alien is not in DS(i))
= P(alien is in cell j) * P(alien is not in DS(i) | alien is in j) / [sum_{k} P(alien in cell k) P(alien is not in DS(i) | alien in cell k)]

As we know,
For every cell k inside DS(i): P(alien is not in DS(i) | alien in cell k) = 0
For every cell k outside DS(i): P(alien is not in DS(i) | alien in cell k) = 1

P(alien is in cell j | alien is not in DS(i))
= P(alien is in cell j) * P(alien is not in DS(i) | alien is in j) / [sum_{k outside DS(i)} P(alien in cell k)]

A. If j is in DS(i):

P(alien is not in DS(i) | alien is in j) = 0

Therefore,
P(alien is in cell j | alien is not in DS(i)) = 0

B. If j is not in DS(i):

P(alien is not in DS(i) | alien is in j) = 1

Therefore,
P(alien is in cell j | alien is not in DS(i)) = P(alien is in cell j) / [sum_{k outside DS(i)} P(alien in cell k)]

II. Alien Detection Positive

P(alien is in cell j | alien is in DS(i)) for each cell j
= P(alien is in cell j AND alien is in DS(i)) / P(alien is in DS(i))
= P(alien is in cell j) * P(alien is in DS(i) | alien is in j) / P(alien is in DS(i)) —> (a)

P(alien is in DS(i))
= sum_{k} P(alien in cell k AND alien is in DS(i))
= sum_{k} P(alien in cell k) P(alien is in DS(i) | alien in cell k)

Substitute in (a):
P(alien is in cell j | alien is in DS(i))
= P(alien is in cell j) * P(alien is in DS(i) | alien is in j) / [sum_{k} P(alien in cell k) P(alien is in DS(i) | alien in cell k)]

As we know,
For every cell k inside DS(i): P(alien is in DS(i) | alien in cell k) = 1
For every cell k outside DS(i): P(alien is in DS(i) | alien in cell k) = 0

P(alien is in cell j | alien is in DS(i))
= P(alien is in cell j) * P(alien is in DS(i) | alien is in j) / [sum_{k inside DS(i)} P(alien in cell k)]

A. If j is in DS(i):

P(alien is in DS(i) | alien is in j) = 1

Therefore,
P(alien is in cell j | alien is in DS(i)) = P(alien is in cell j) / [sum_{k inside DS(i)} P(alien in cell k)]

B. If j is not in DS(i):

P(alien is in DS(i) | alien is in j) = 0

Therefore,
P(alien is in cell j | alien is in DS(i)) = 0

III. Alien Normalization after Bot Movement

P(alien in j | alien not in i)
= P(alien in j AND alien not in i) / P(alien not in i)
= P(alien in j) * P(alien not in i | alien in j) / P(alien not in i)
= P(alien in j) / P(alien not in i)
= P(alien in j) / 1 - P(alien in i)

IV. Alien Belief after Bot Movement

P(alien in j) for each cell j
= sum_{x is valid_neighbors of cell j} P(alien in x) / No of valid neighbors of cell x

2) Model 2: Single Crew

Initial Knowledge Base:

P(crew at cell j) for each cell j

Bot is in cell i
P(crew is in cell i) = 0

DS(i) = {open cells in the detection square centered at i}

Updating Knowledge Base:

I. Beep is heard:

P(crew is in cell j | beep in cell i)
= P(crew in cell j AND beep in cell i) / P(beep in cell i)
= P(crew in cell j) * P(beep in cell i | crew in cell j) / P(beep in cell i)
= P(crew in cell j) * e^(-a*(d(i,j)-1)) / P(beep in cell i)

Now calculate for the denominator,
P(beep in cell i)
= sum_{k} P(crew in cell k AND beep in cell i)
= sum_{k} P(crew in cell k) * P(beep in cell i | crew in cell k)
= sum_{k} P(crew in cell k) * e^(-a*(d(i,k)-1))

II. Beep is not heard:

P(crew is in cell j | no beep in cell i)
= P(crew in cell j AND no beep in cell i) / P(no beep in cell i)
= P(crew in cell j) * P(no beep in cell i | crew in cell j) / P(no beep in cell i)
= P(crew in cell j) *(1 - e^(-a*(d(i,j)-1))) / P(no beep in cell i)

Now calculate for the denominator,
P(no beep in cell i)
= sum_{k} P(crew in cell k AND no beep in cell i)
= sum_{k} P(crew in cell k) * P(no beep in cell i | crew in cell k)
= sum_{k} P(crew in cell k) * (1 - e^(-a*(d(i,k)-1)))

3) Model 3: Two Crew

Initial Knowledge Base:

P(crew in cell x and crew in cell y) for each pair of cells (x, y)

Bot is in cell i
P(crew is in cell i) = 0

DS(i) = {open cells in the detection square centered at i}

Updating Knowledge Base:

I. **Beep is heard:**

P(crew in x and crew in y | beep in cell i)
= P(crew in x and crew in y AND beep in cell i) / P(beep in cell i)
= P(crew in x and crew in y) * P(beep in cell i | crew in x and crew in y) / P(beep in cell i)

Now let's calculate,
P(beep in cell i | crew in x and crew in y)
= P(crew in x) OR P(crew in y)
= P(crew in x) + P(crew in y) - P(crew in x) * P(crew in y)
= e^(-a*(d(i,x)-1)) + e^(-a*(d(i,y)-1)) - [(e^(-a*(d(i,x)-1))) * (e^(-a*(d(i,y)-1)))]

Now calculate for the denominator,
P(beep in cell i)
= sum_{s, t} P(crew in s and crew in t AND beep in cell i)
= sum_{s, t} P(crew in s and crew in t) * P(beep in cell i | crew in s and crew in t)
= sum_{s, t} P(crew in s and crew in t) * P(beep in cell i | crew in s and crew in t)

II. **Beep is not heard:**

P(crew in x and crew in y | no beep in cell i)
= P(crew in x and crew in y AND no beep in cell i) / P(no beep in cell i)
= P(crew in x and crew in y) * P(no beep in cell i | crew in x and crew in y) / P(no beep in cell i)
= P(crew in x and crew in y) * (1 - P(beep in cell i | crew in x and crew in y)) / (1 - P(beep in cell i))

Now let's calculate,
P(beep in cell i | crew in x and crew in y)
= P(crew in x) OR P(crew in y)
= P(crew in x) + P(crew in y) - P(crew in x) * P(crew in y)
= e^(-a*(d(i,x)-1)) + e^(-a*(d(i,y)-1)) - [(e^(-a*(d(i,x)-1))) * (e^(-a*(d(i,y)-1)))]

Now calculate for the denominator,
P(beep in cell i)
= sum_{s, t} P(crew in s and crew in t AND beep in cell i)
= sum_{s, t} P(crew in s and crew in t) * P(beep in cell i | crew in s and crew in t)
= sum_{s, t} P(crew in s and crew in t) * P(beep in cell i | crew in s and crew in t)

III. **Crew Normalization after Bot Movement**

P(crew in x and crew in y | no crew in i)
= P(crew in x and crew in y AND no crew in i) / P(no crew in i)
= P(crew in x and crew in y) * P(no crew in i | crew in x and crew in y) / P(no crew in i)
= P(crew in x and crew in y) * 1 / P(no crew in i)
= P(crew in x and crew in y) / (1 - P(crew in i))

4) Model 4: Two Alien

Initial Knowledge Base:

P(alien in cell x and alien in cell y) for each pair of cells (x, y)

Bot is in cell i
P(crew is in cell i) = 0

DS(i) = {open cells in the detection square centered at i}

Updating Knowledge Base:

I. Alien Detection is Negative

P(alien in x and alien in y | alien is not in DS(i)) for each pair of cells (x, y)
= P(alien in x and alien in y AND alien is not in DS(i)) / P(alien is not in DS(i))
= P(alien in x and alien in y) * P(alien is not in DS(i) | alien in x and alien in y) / P(alien is not in DS(i)) —> (a)

Now calculate for the denominator,
P(alien is not in DS(i))
= sum_{s, t} P(alien in s and alien in t AND alien is not in DS(i))
= sum_{s, t} P(alien in s and alien in t) P(alien is not in DS(i) | alien in s and alien in t)

Substitute in (a):
P(alien in x and alien in y | alien is not in DS(i))
= P(alien in x and alien in y) * P(alien is not in DS(i) | alien in x and alien in y) / sum_{s, t} P(alien in s and alien in t) P(alien is not in DS(i) | alien in s and alien in t)

As we know,
For both cells s, t inside DS(i): P(alien is not in DS(i) | alien in s and alien in t) = 0
For both cells s, t outside DS(i): P(alien is not in DS(i) | alien in s and alien in t) = 1

P(alien in x and alien in y | alien is not in DS(i))
= P(alien in x and alien in y) * P(alien is not in DS(i) | alien in x and alien in y) / [sum_{s, t outside DS(i)} P(alien in s and alien in t)]

A. If x,y is in DS(i):

P(alien is not in DS(i) | alien in x and alien in y) = 0

Therefore,
P(alien in x and alien in y | alien is not in DS(i)) = 0

B. If x,y is not in DS(i):

P(alien is not in DS(i) | alien in x and alien in y) = 1

Therefore,
P(alien in x and alien in y | alien is not in DS(i)) = P(alien in x and alien in y) / [sum_{s, t outside DS(i)} P(alien in s and alien in t)]

II. Alien Detection is Positive

P(alien in x and alien in y | alien is in DS(i)) for each pair of cells (x, y)
= P(alien in x and alien in y AND alien is in DS(i)) / P(alien is in DS(i))
= P(alien in x and alien in y) * P(alien is in DS(i) | alien in x and alien in y) / P(alien is in DS(i)) —> (a)

Now calculate for the denominator,
P(alien is in DS(i))
= sum_{s, t} P(alien in s and alien in t AND alien is in DS(i))
= sum_{s, t} P(alien in s and alien in t) P(alien is in DS(i) | alien in s and alien in t)

Substitute in (a):
P(alien in x and alien in y | alien is not in DS(i))
= P(alien in x and alien in y) * P(alien is in DS(i) | alien in x and alien in y) / sum_{s, t} P(alien in s and alien in t) P(alien is in DS(i) | alien in s and alien in t)

As we know,
For both cells s, t inside DS(i): P(alien is in DS(i) | alien in s and alien in t) = 1
For both cells s, t outside DS(i): P(alien is in DS(i) | alien in s and alien in t) = 0

P(alien in x and alien in y | alien is in DS(i))
= P(alien in x and alien in y) * P(alien is in DS(i) | alien in x and alien in y) / [sum_{s, t inside DS(i)} P(alien in s and alien in t)]

A. **If x,y is in DS(i):**
   P(alien is in DS(i) | alien in x and alien in y) = 1

   Therefore,
   P(alien in x and alien in y | alien is in DS(i)) = P(alien in x and alien in y) / [sum_{s, t inside DS(i)} P(alien in s and alien in t)]

B. **If x,y is not in DS(i):**
   P(alien is in DS(i) | alien in x and alien in y) = 0

   Therefore,
   P(alien in x and alien in y | alien is in DS(i)) = 0

III. Alien Normalization after Bot Movement

P(alien in x and alien in y | alien is not in i)
= P(alien in x and alien in y AND alien is not in i) / P(alien is not in i)
= P(alien in x and alien in y) * P(alien is not in i | alien in x and alien in y) / P(alien is not in i)
= P(alien in x and alien in y) * 1 / P(alien is not in i)
= P(alien in x and alien in y) / (1 - P(alien is in i))

IV. Alien Belief after Alien Movement

P(alien in m and alien in n at time = t+1 | alien in x and alien in y at time = t)
= (1/(No of neighbors of x) if m is a neighbor of x, 0 else) * (1/(No of neighbors of y) if n is a neighbor of y, 0 else) + (1/(No of neighbors of x) if n is a neighbor of x, 0 else) * (1/(No of neighbors of y) if m is a neighbor of y, 0 else)

Analysis of Bot Strategies

One Alien, One Crew Member

1) Bot 1

Bot 1 begins with uncertainty about the alien's location and assumes equal probability for the alien to be in any open cell outside its detection square. Crew members are assumed to be equally likely in any cell except the bot's initial cell.

Bot 1 utilizes a sensor to detect aliens within the (2k + 1) × (2k + 1) square centered at its location as well as the beep from the crew with the probability e^(-α(d-1)). It updates its alien probability and crew probability by using the model 1 and model 2 respectively based on the data received from the sensors.

Bot 1 prioraties moving towards the cell with high probability of containing the crew, while avoiding any cell having alien probability.

We use 2 dimensional matrices to store the probability of crew and an alien.

A* algorithm is used to find the best path for the current bot position to the cell with highest crew probability. The heuristic used is manhattan distance + the probability of alien being in that cell.

Failure:

- Bot 1's purely move while avoiding the aliens, leading to the bot taking longer paths
- Bot 1’s paths towards the cell with highest probability and disregard the cells it crosses over.

Improvement:

- Using a utility function to find a path to the cell with highest crew probability while patching over the cells most likely to have a crew in them, while avoiding cells with alien probability is implemented in Bot 2.

2) Bot 2

Bot 2 begins with uncertainty about the alien's location and assumes equal probability for the alien to be in any open cell outside its detection square. Crew members are assumed to be equally likely in any cell except the bot's initial cell.

Bot 2 utilizes a sensor to detect aliens within the (2k + 1) × (2k + 1) square centered at its location as well as the beep from the crew with the probability e^(-α(d-1)). It updates its alien probability and crew probability by using the model 1 and model 2 respectively based on the data received from the sensors.

We use 2 dimensional matrices to store the probability of crew and an alien.

A* algorithm is used to find the best path for the current bot position to the cell with highest crew probability. The heuristic used is the Manhattan distance + Utility function.

Utility function:

Utility function has been added to the heuristics in the form of h1 and h2 for an improvised performance.
h1: Gives a heuristic of alien probability in a cell
h2: Gives a heuristic of crew probability in a cell

h1 vs h2 represents the same idea of risk-avoidance vs risk-taking behavior for the bot.

For a higher value of h1, the bot will strictly avoid the cells with high alien probability, partially disregarding the crew's position. In this scenario, the bot prioritizes saving itself more than finding the crew.

For a higher value of h2. the bot will majorly move towards the cell with higher crew probability, essentially ignoring the alien's position. In this case, the bot tries to keep following the best path with high crew cell probability, disregarding the possibility of encountering the alien.

A dynamic factor is also chosen to adjust the value of the utility being added to the heuristics. This factor varies with the initial heuristics so as to keep the utility value in a reasonable effective range.

One Alien, Two Crew Member

3) Bot 3

Bot 3 is an extension of Bot 1 in the scenario with two crew members. When Bot 3 receives the crew beep, it ignores by which crew member it received the beep.

Bot 3 begins with uncertainty about the alien's location and assumes equal probability for the alien to be in any open cell outside its detection square. Crew members are assumed to be equally likely in any cell except the bot's initial cell.

Bot 3 utilizes a sensor to detect aliens within the $(2k + 1) \times (2k + 1)$ square centered at its location as well as the beep from any of the two the crews with the probability $e^{(-\alpha(d-1))}$. It updates its alien probability and crew probability by using the model 1 and model 2 respectively.

We use 2 dimensional matrices to store the probability of crew and an alien.

A* algorithm is used to find the best path for the current bot position to the cell with highest crew probability. The heuristic used is manhattan distance + the probability of alien being in that cell.

Failure:

- Bot 3 ignores the fact that the beep could be from any of the two crews and updates the matrices accordingly.

Improvement:

- The probability matrix of the crew should be updated in accordance to the fact that the beep can come from any of the 2 crew.

4) Bot 4

Bot 4 begins with uncertainty about the alien's location and assumes equal probability for the alien to be in any open cell outside its detection square. Crew members are assumed to be equally likely in any open cell pair except for the pairs containing the bot's initial cell.

Bot 4 utilizes a sensor to detect aliens within the $(2k + 1) \times (2k + 1)$ square centered at its location as well as the beep from any of the two crews with the probability $e^{(-\alpha(d-1))}$. It updates its alien probability and crew probability by using the model 3 and model 4 respectively.

We use a 2 dimensional matrix to store the probability of alien and a 4 dimensional matrix to store the probability of the crew.

The crew probability is stored in the form of the probability of a pair of cells containing the crew .

A* algorithm is used to find the best path for the current bot position to the cell with highest crew probability. The heuristic used is manhattan distance + the probability of alien being in that cell.

Failure:

- Since it just focuses on alien avoidance and reaching the cell with highest crew probability.

Improvement:

- Use a utility function such that the path taken to the cell with highest crew probability, not only avoids the cell with alien probability but also tries to maximize the path crossed over cells most likely to have crew.

5) Bot 5

Bot 5 begins with uncertainty about the alien's location and assumes equal probability for the alien to be in any open cell outside its detection square. Crew members are assumed to be equally likely in any open cell pair except for the pairs containing the bot's initial cell.

Bot 5 utilizes a sensor to detect aliens within the $(2k + 1) \times (2k + 1)$ square centered at its location as well as the beep from any of the two crews with the probability $e^{(-\alpha(d-1))}$. It updates its alien probability and crew probability by using the model 3 and model 2 respectively.

We use a 2 dimensional matrix to store the probability of alien and a 4 dimensional matrix to store the probability of the crew.

The crew probability is stored in the form of the probability of a pair of cells containing the crew .

Bot 5 uses A* with utility function to find the path

Utility function:

Utility function has been added to the heuristics in the form of h1 and h2 for an improvised performance.
h1: Gives a heuristic of alien probability in a cell
h2: Gives a heuristic of crew probability in a cell

h1 vs h2 represents the same idea of risk-avoidance vs risk-taking behavior for the bot.

For a higher value of h1, the bot will strictly avoid the cells with high alien probability, partially disregarding the crew’s position. In this scenario, the bot prioritizes saving itself more than finding the crew.

For a higher value of h2. the bot will majorly move towards the cell with higher crew probability, essentially ignoring the alien’s position. In this case, the bot tries to keep following the best path with high crew cell probability, disregarding the possibility of encountering the alien.

A dynamic factor is also chosen to adjust the value of the utility being added to the heuristics. This factor varies with the initial heuristics so as to keep the utility value in a reasonable effective range.

Two Alien, Two Crew Member

6) Bot 6

Bot 6 is an extension of Bot 1 in the scenario with two crew members and 2 aliens. When Bot 3 receives the crew beep, it ignores by which crew member it received the beep. Similarly it also ignores that there are 2 aliens.

Bot 6 begins with uncertainty about the alien's location and assumes equal probability for the alien to be in any open cell outside its detection square. Crew members are assumed to be equally likely in any cell except the bot's initial cell.

Bot 6 utilizes a sensor to detect aliens within the (2k + 1) × (2k + 1) square centered at its location as well as the beep from any of the two the crews with the probability e^(-α(d-1)). It updates its alien probability and crew probability by using the model 1 and model 2 respectively.

We use 2 dimensional matrices to store the probability of crew and an alien.

A* algorithm is used to find the best path for the current bot position to the cell with highest crew probability. The heuristic used is manhattan distance + the probability of alien being in that cell.

Failure:

- Bot 6 ignores the fact that the beep could be from any of the two crews and updates the matrix accordingly.
- Similarly it also ignores that there are 2 aliens

Improvement:

- The probability matrices should be updated in accordance with the fact that the beep can come from any of the 2 crews or aliens.

7) Bot 7

Bot 7 is bot 6 with correct updates to bot alien and crew matrices.

Bot 7 begins with uncertainty about the alien's location and assumes equal probability for the alien to be in any open cell outside its detection square. Crew members are assumed to be equally likely in any open cell pair except for the pairs containing the bot's initial cell.

Bot 7 utilizes a sensor to detect aliens within the (2k + 1) × (2k + 1) square centered at its location as well as the beep from any of the two crews with the probability e^(-α(d-1)). It updates its alien probability and crew probability by using the model 3 and model 4 respectively.

We use a 4 dimensional matrix to store the probability of aliens and a 4 dimensional matrix to store the probability of the crew.

The crew and alien probability is stored in the form of the probability of a pair of cells containing the crew and aliens .

A* algorithm is used to find the best path for the current bot position to the cell with highest crew probability. The heuristic used is manhattan distance + the probability of alien being in that cell.

Failure:

- Since it just focuses on alien avoidance and reaching the cell with highest crew probability.

Improvement:

- Use a utility function such that the path taken to the cell with highest crew probability, not only avoids the cell with alien probability but also tries to maximize the path crossed over cells most likely to have crew.

8) <u>Bot 8</u>

Bot 8 is Bot 7 with the addition of a utility function.

Bot 8 begins with uncertainty about the alien's location and assumes equal probability for the alien to be in any open cell outside its detection square. Crew members are assumed to be equally likely in any open cell pair except for the pairs containing the bot's initial cell.

Bot 8 utilizes a sensor to detect aliens within the $(2k + 1) \times (2k + 1)$ square centered at its location as well as the beep from any of the two crews with the probability $e^{(-\alpha(d-1))}$. It updates its alien probability and crew probability by using the model 3 and model 4 respectively.

We use a 4 dimensional matrix to store the probability of aliens and a 4 dimensional matrix to store the probability of the crew.

The crew and alien probability is stored in the form of the probability of a pair of cells containing the crew and aliens .

Bot 8 uses A* with utility function to find the path

Utility function:

Utility function has been added to the heuristics in the form of h1 and h2 for an improvised performance.
h1: Gives a heuristic of alien probability in a cell
h2: Gives a heuristic of crew probability in a cell

h1 vs h2 represents the same idea of risk-avoidance vs risk-taking behavior for the bot.

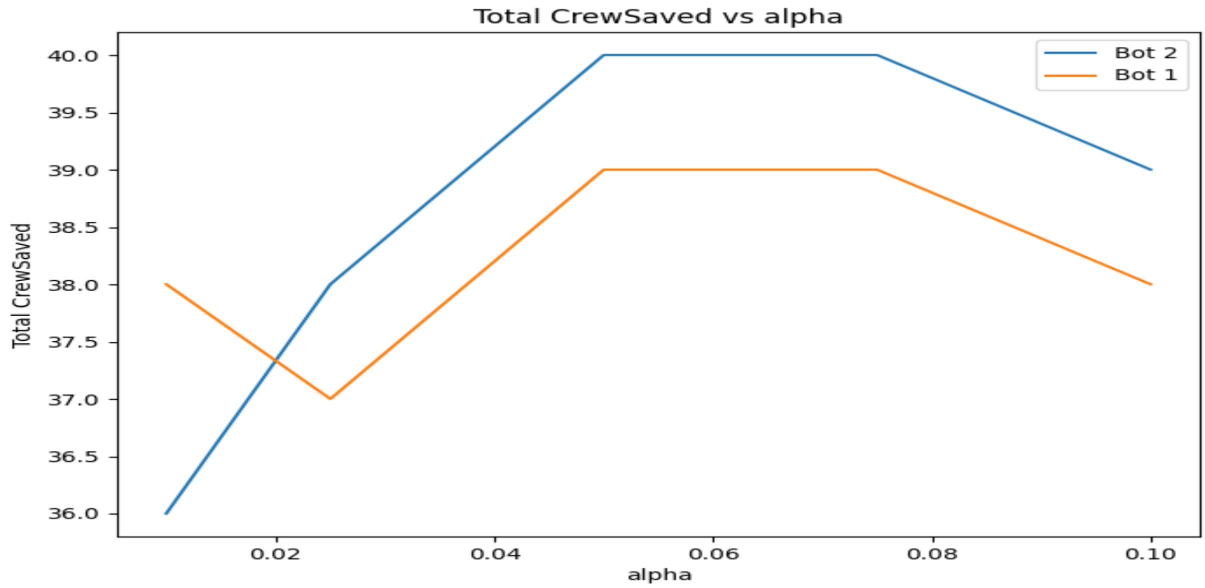
For a higher value of h1, the bot will strictly avoid the cells with high alien probability, partially disregarding the crew’s position. In this scenario, the bot prioritizes saving itself more than finding the crew.

For a higher value of h2. the bot will majorly move towards the cell with higher crew probability, essentially ignoring the alien’s position. In this case, the bot tries to keep following the best path with high crew cell probability, disregarding the possibility of encountering the alien.

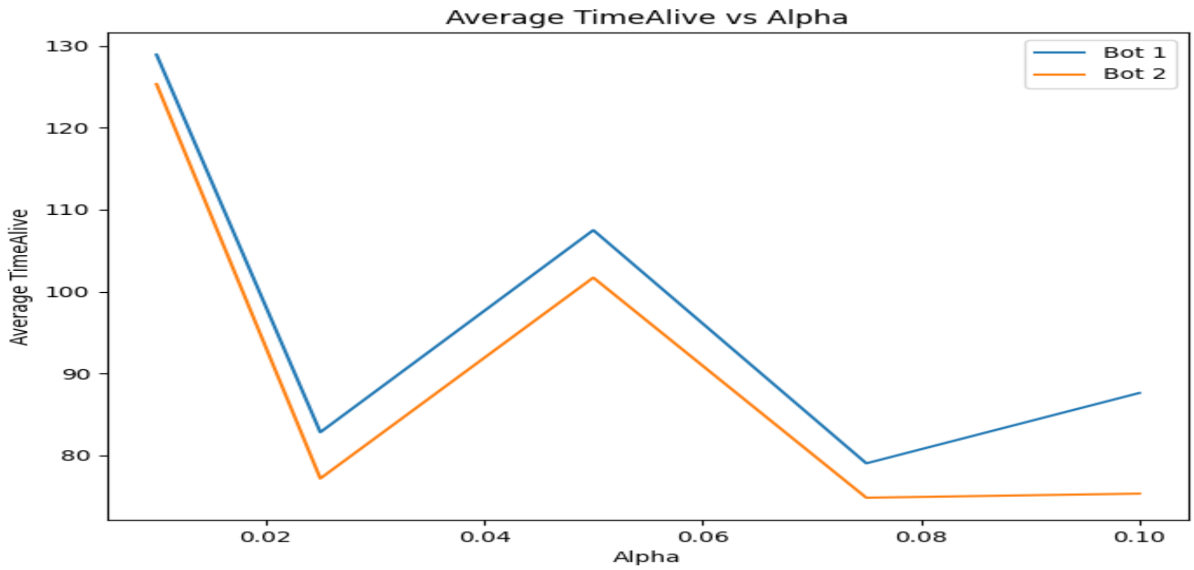
A dynamic factor is also chosen to adjust the value of the utility being added to the heuristics. This factor varies with the initial heuristics so as to keep the utility value in a reasonable effective range.

Bot Performance

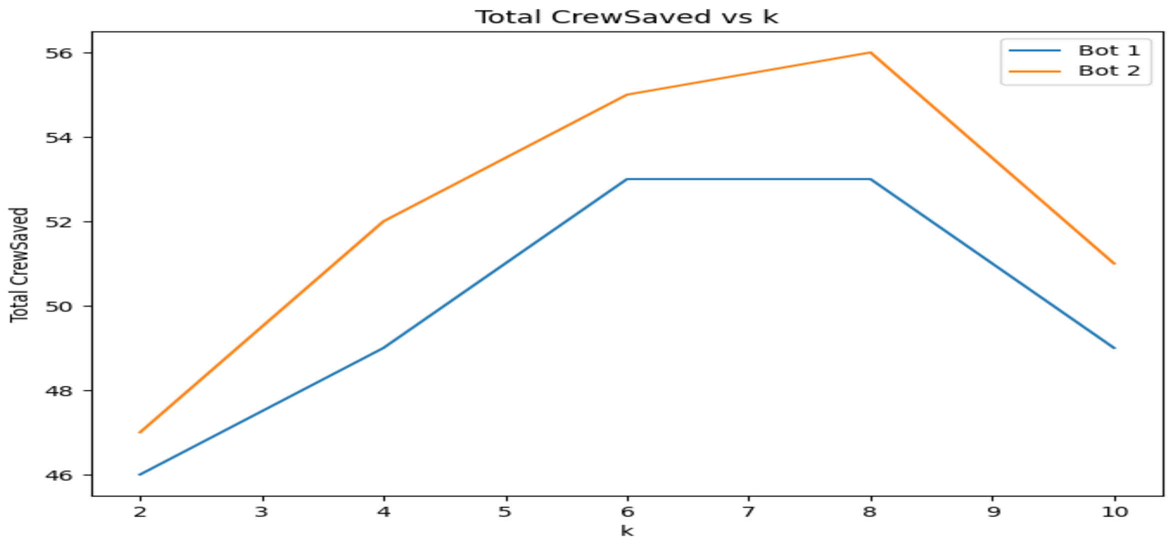
1) One Alien, One Crew Member



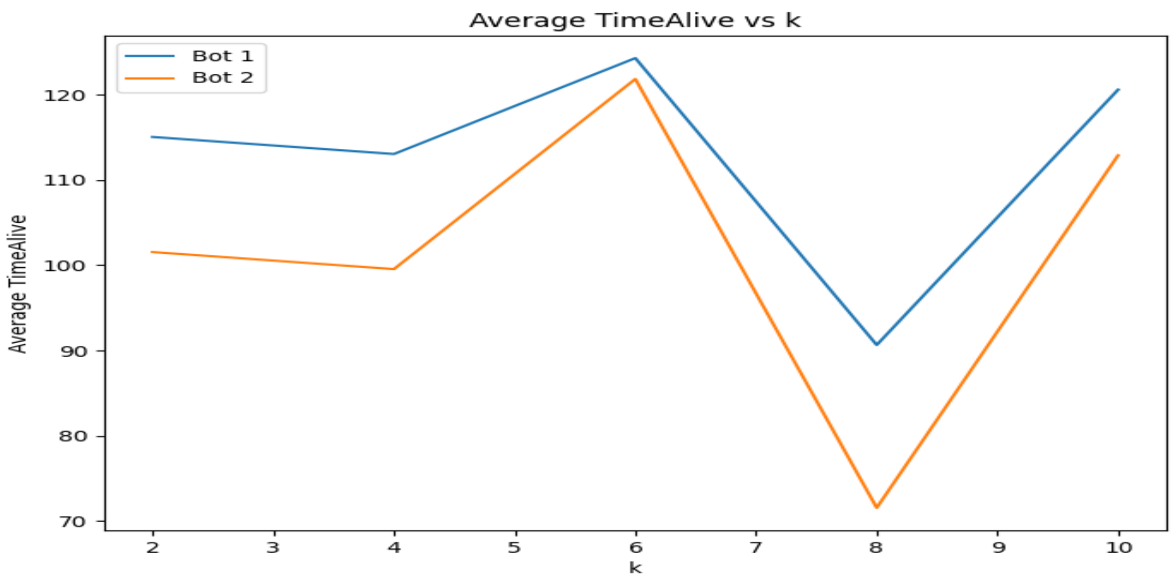
As visible in the graph for 40 epochs, Bot 2 shows improvement over Bot 1. A significant increase in the number of saved crew members can be observed in Bot 2 when alpha is in the interval [0.06, 0.07]. This alpha interval provides the bot with sufficient information about the crew's position.
However, for alpha values lower than 0.03, the bot has more probability of getting a beep for the crew even though the crew is far away from the bot and for alpha values greater than 0.07 the bot has less probability of getting a beep for the crew closer to the bot.



The graph above visually confirms that Bot 2 exhibits improvement over Bot 1. A reduction in the number of bot moves to save crew members can be observed in Bot 2 when alpha is 0.05, providing the bot with sufficient information about the crew's position.
This leads to the inference that the enhanced heuristics for Bot 2 enable it to save more crew lives in less time compared to Bot 1.

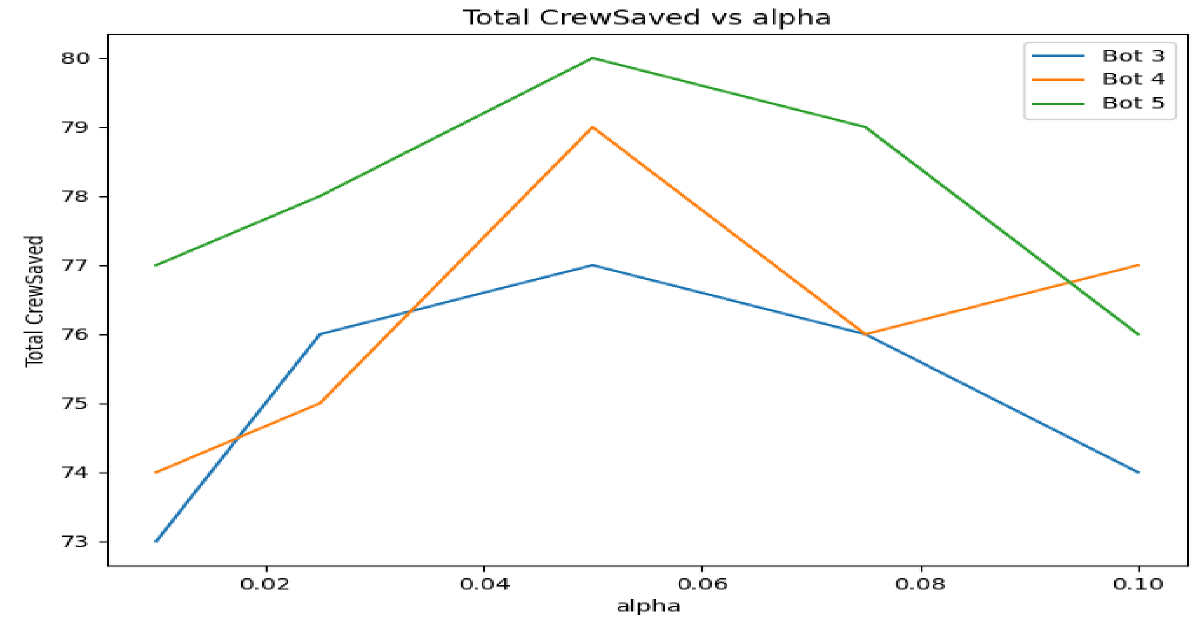


Again, the above graph for 60 epochs visually affirms that Bot 2 exhibits improvement over Bot 1.
An interesting observation here is A significant increase in the number of saved crew members can be observed in Bot 2 when k is in the interval [6, 8]. This k interval provides the bot with sufficient information about the alien position.
However, for k values lower than 6, the alien would be closer to the bot, resulting in bot death. For k values greater than 6, the bot will be less likely to know the alien position, making it unable to avoid the aliens intelligently and resulting in bot death.

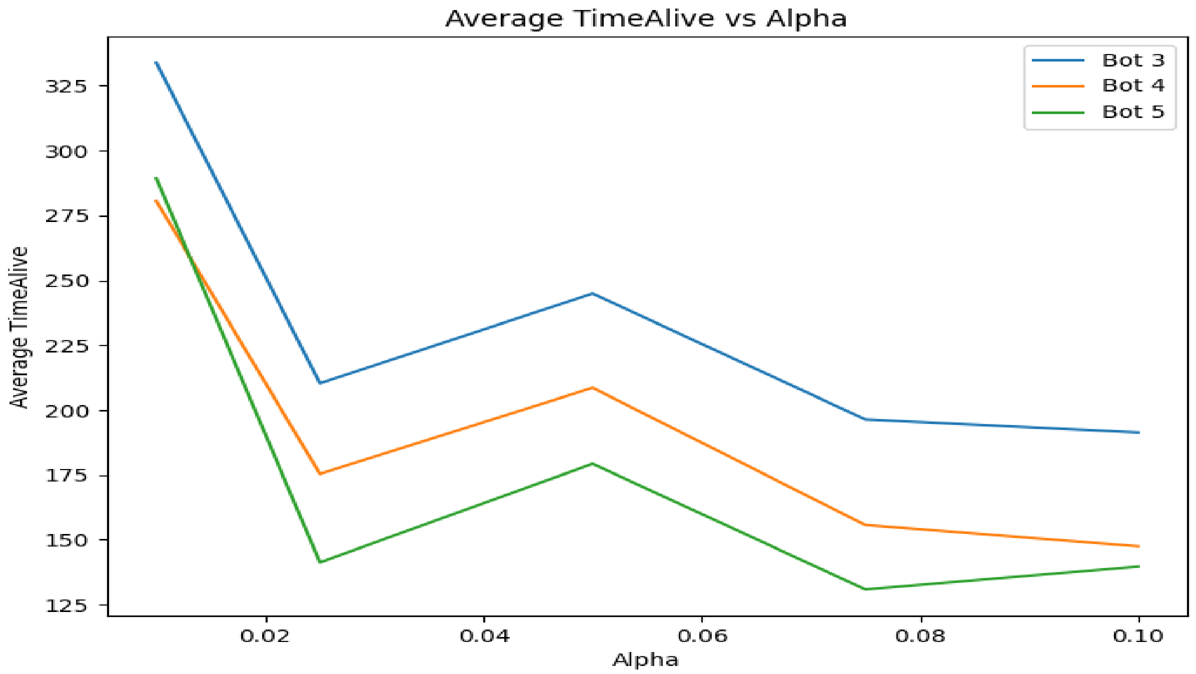


Once again, the graph above visually confirms that Bot 2 exhibits improvement over Bot 1. A reduction in the number of bot moves to save crew members can be observed in Bot 2 when k is 6. This k value provides the bot with sufficient information about the alien position.
However, for k values lower than 6, the alien would be closer to the bot, resulting in bot death. For k values greater than 6, the bot will be less likely to know the alien position, making it unable to avoid the aliens intelligently and resulting in bot death.
This leads to the inference that the enhanced heuristics for Bot 2 enable it to save more crew lives in less time compared to Bot 1.

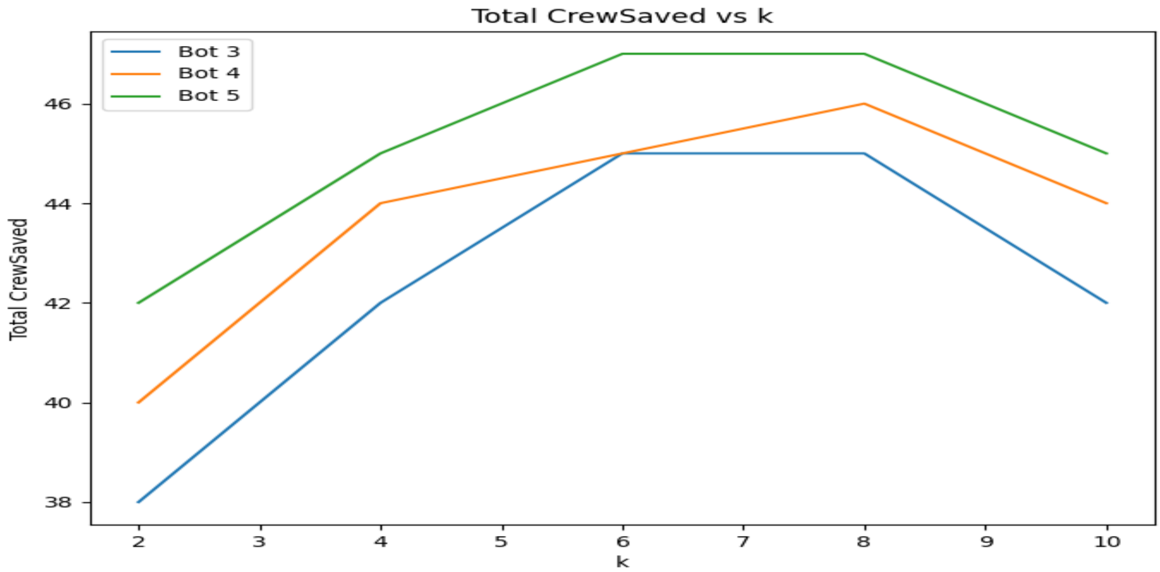
2) One Alien, Two Crew Members



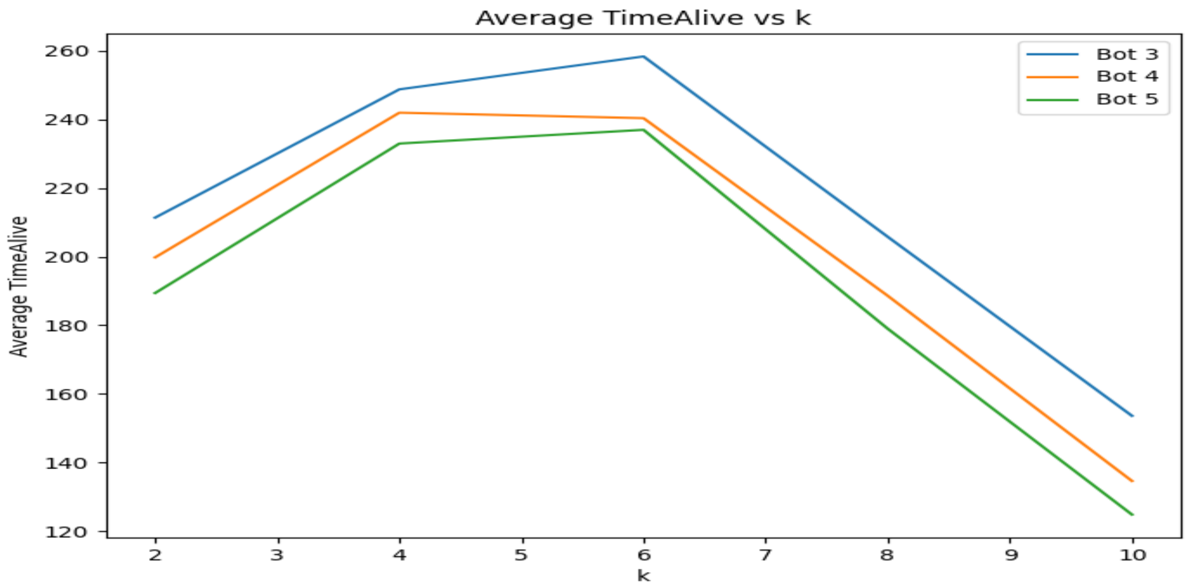
Once again for 40 epochs, successive improvement can be seen from Bot 3 to Bot 4 to Bot 5. The peak denotes the optimal alpha value where a jump in crew saved can be seen. This alpha interval provides the bot with sufficient information about the crew's position.
However, for lower alpha values, the bot receives a beep even when the crew is far away, thus the performance plummets. Also, the bot doesn't receive a beep even for a close crew, thus again a dip in performance is noticed.



The above graph reaffirms that Bot 3, 4, and 5 exhibit improved performance in succession. The enhanced performance can be witnessed in the form of reduced steps required to save the crew. This leads to the inference that the improved heuristics for Bot 5 enable it to save more crew in less time compared to Bot 4.

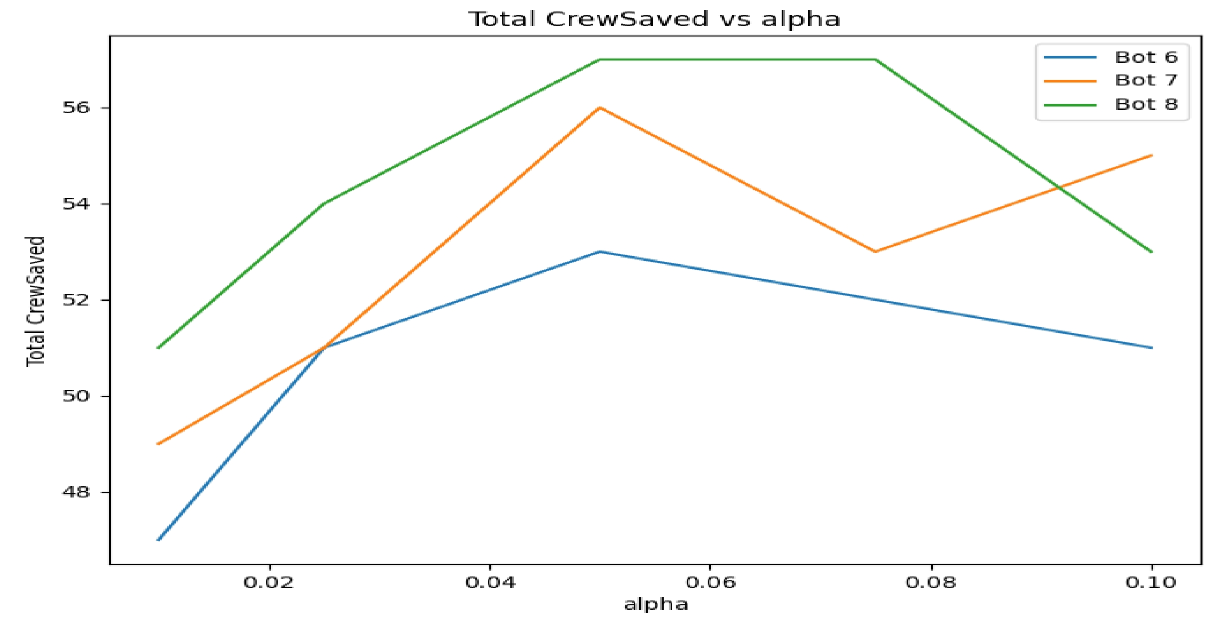


Again for 25 epochs, the above graph visually affirms successive improvements between Bot 3, 4, and 5. A significant increase in the number of saved crew members can be observed in Bot 5 when k is in the interval [6, 8]. This k interval provides the bot with sufficient information about the alien position.
However, for k values lower than 6, the alien would be closer to the bot, resulting in bot death. For k values greater than 8, the bot will be less likely to know the alien position, making it unable to avoid the aliens intelligently and resulting in bot death.



Once again, the graph above visually confirms the successive improvements between Bot 3, 4, and 5. A reduction in the number of bot moves to save crew members can be observed in Bot 4 and 5 when k is 6. This k value provides the bot with sufficient information about the alien position. However, for k values lower than 6, the alien would be closer to the bot, resulting in bot death.For k values greater than 6, the bot will be less likely to know the alien position, making it unable to avoid the aliens intelligently and resulting in bot death.
This leads to the inference that the enhanced heuristics for Bot 5 enable it to save more crew lives in less time compared to Bot 4.

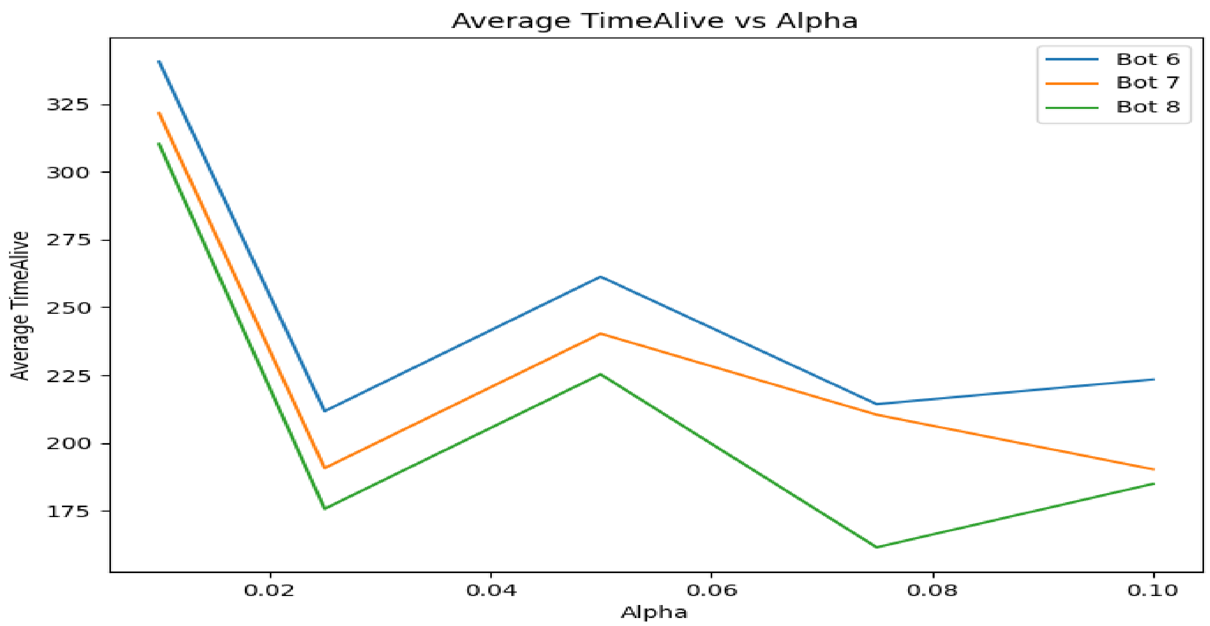
3) Two Aliens, Two Crew Members



Again for 30 epochs, successive improvement can be seen from Bot 6 to Bot 7 to Bot 8. The peak denotes the optimal alpha value where a jump in crew saved can be seen. This alpha interval provides the bot with sufficient information about the crew's position.

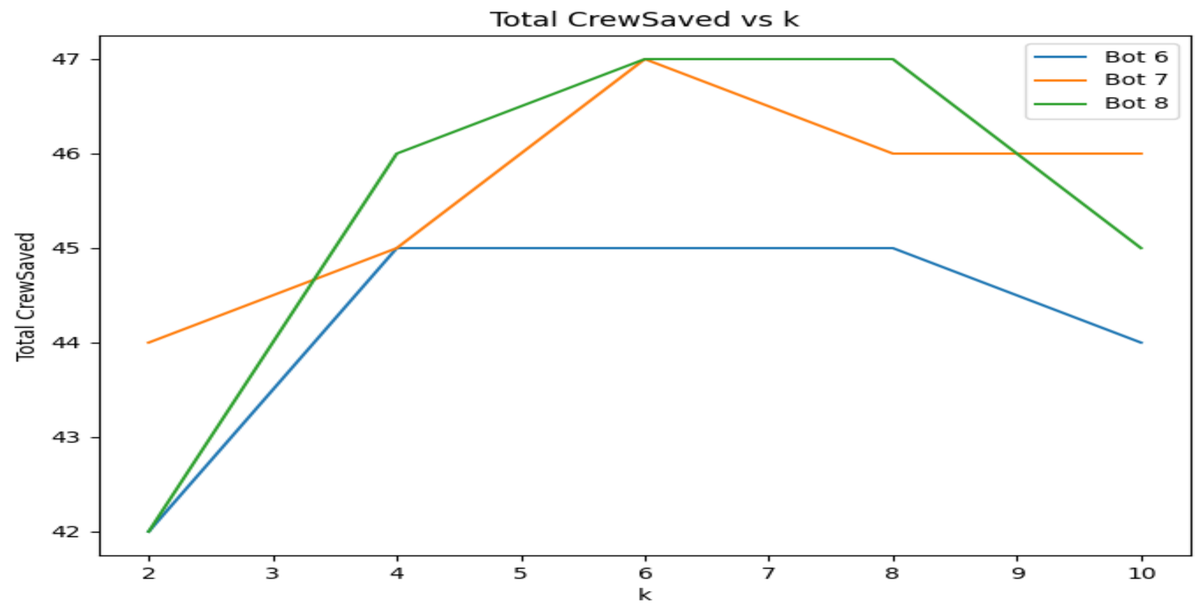
However, for lower alpha values, the bot receives a beep even when the crew is far away, thus the performance plummets. Also, the bot doesn't receive a beep even for a close crew, thus again a dip in performance is noticed.

:

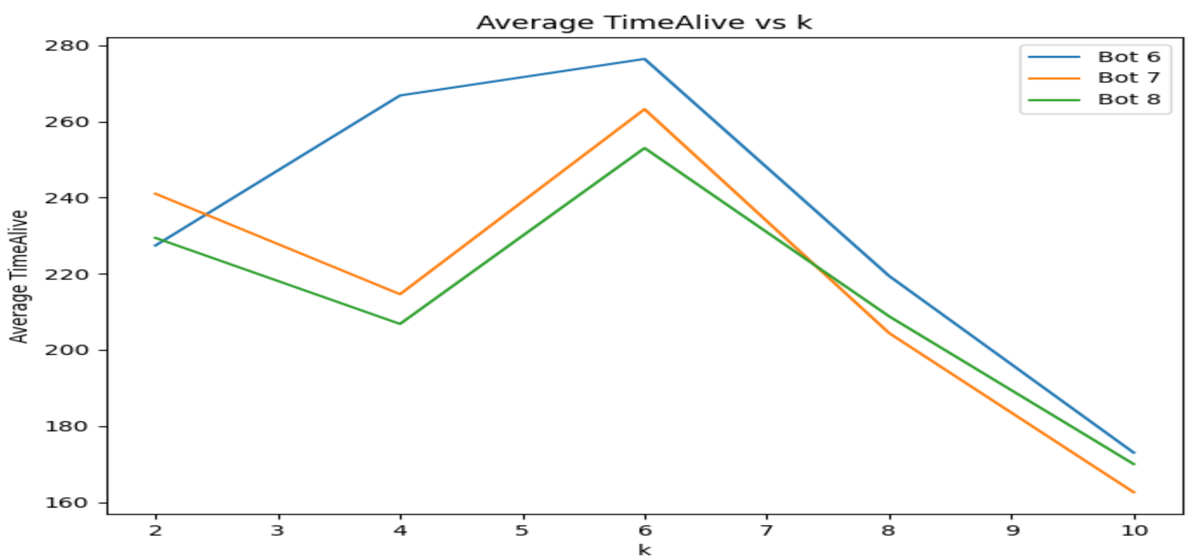


The above graph re-affirms that Bot 6, 7 and 8 exhibit improved performance in succession. The improved performance can be witnessed in the form of reduced steps required to save the crew.

This leads to the inference that the enhanced heuristics for Bot 8 enables it to save more crew lives in lesser time as compared to Bot 7.



Again for 25 epochs, the above graph visually affirms successive improvements between Bot 6, 7 and 8.
A significant increase in the number of saved crew members can be observed in Bot 8 when k is in the interval [6, 8]. This k interval provides the bot with sufficient information about the alien position.
However, for k values lower than 6, the alien would be closer to the bot, resulting in bot death. For k values greater than 6, the bot will be less likely to know the alien position, making it unable to avoid the aliens intelligently and resulting in bot death.



Once again, the graph above visually confirms the successive improvements between Bot 6, 7 and 8. A reduction in the number of bot moves to save crew members can be observed in Bot 6, 7 and 8 when k is 6. This k value provides the bot with sufficient information about the alien position.
However, for k values lower than 6, the alien would be closer to the bot, resulting in bot death. For k values greater than 6, the bot will be less likely to know the alien position, making it unable to avoid the aliens intelligently and resulting in bot death.
This leads to the inference that the enhanced heuristics for Bot 8 enable it to save more crew lives in less time compared to Bot 7.

Ideal Bot

- Optimal value of k and alpha:
  - The performance of the bot depends predominantly on the values of k and alpha. In this project, due to limited computational capabilities, only a handful of k and alpha values could be tested. With more computational resources, a more extensive search for an optimal value of k and alpha can be carried out. With the most accurate optimal value of both k and alpha, the bot can be expected to perform significantly better.

- Dynamic heuristics h1 and h2:
  - Moreover, the heuristics h1 and h2 can also be calculated for an optimal value with more computational power. Also, a better approach would be to update the heuristics h1 and h2 with every step. As the bot covers more of the grid, it gains more knowledge about the cells. With more knowledge about these cells, we are in a better position to change and update the heuristics at each step.

- A similar approach for finding the optimal value of the factor (in the utility function) can be employed for enhanced performance.

BONUS QUESTION

Crew movement adds extra complexity to probability updation of crew.
This can be achieved by adjusting the probability matrix to the crew movement. Which is similar to updating the alien probability matrix after alien movement .
Below shows how it can be done in case of one crew.

Model: Single Crew with movement

Initial Knowledge Base:
P(crew at cell j) for each cell j

Bot is in cell i
P(crew is in cell i) = 0

DS(i) = {open cells in the detection square centered at i}

Updating Knowledge Base:

I. Beep is heard:
P(crew is in cell j | beep in cell i)
= P(crew in cell j AND beep in cell i) / P(beep in cell i)
= P(crew in cell j) * P(beep in cell i | crew in cell j) / P(beep in cell i)
= P(crew in cell j) * e^(-a*(d(i,j)-1)) / P(beep in cell i)

Now calculate for the denominator,
P(beep in cell i)
= sum_{k} P(crew in cell k AND beep in cell i)
= sum_{k} P(crew in cell k) * P(beep in cell i | crew in cell k)
= sum_{k} P(crew in cell k) * e^(-a*(d(i,k)-1))

II. Beep is not heard:

P(crew is in cell j | no beep in cell i)
= P(crew in cell j AND no beep in cell i) / P(no beep in cell i)
= P(crew in cell j) * P(no beep in cell i | crew in cell j) / P(no beep in cell i)
= P(crew in cell j) *(1 - e^(-a*(d(i,j)-1))) / P(no beep in cell i)

Now calculate for the denominator,
P(no beep in cell i)
= sum_{k} P(crew in cell k AND no beep in cell i)
= sum_{k} P(crew in cell k) * P(no beep in cell i | crew in cell k)
= sum_{k} P(crew in cell k) * (1 - e^(-a*(d(i,k)-1)))

III. Crew Belief after Crew Movement

P(crew in j) for each cell j
= sum_{x is valid_neighbors of cell j} P(crew in x) / No of valid neighbors of cell x
= sum_{x is valid_neighbors of cell j} (e^(-a*(d(i,x)-1))) / No of valid neighbors of cell x

IV. Crew Updation after Bot Movement

P(crew in j | crew not in i)
= P(crew in j AND crew not in i) / P(crew not in i)
= P(crew in j) * P(crew not in i | crew in j) / P(crew not in i)
= P(crew in j) / P(crew not in i)
= P(crew in j) / 1 - P(crew in i)
= e^(-a*(d(i,j)-1)) / 1 - P(crew in i)