2022

Project 1: Ghosts in a Maze

GYANA DEEPIKA DASARA (GD452)
TEJASHWINI VELICHETI (TV186)

Index of Contents

1. Introduction

1.1 Problem Statement

For this assignment, the problem statement is to develop multiple agents that have the capability of traversing through a maze from start to finish. The agents must find a path from the start to the goal node in a grid that is filled with blocked cells and ghosts. The behaviour about how an agent needs to respond to its environment is defined for each agent. Based on this, the report lays out various approaches which help to create the agents with varied degrees of intelligence and accuracy.

2. Implementation

2.1 Creating the environment

This part of the implementation requires multiple steps revolving around maze creation, ghost creation and placement, and obstacle placement within the maze.

2.1.1 Creating the maze

For creating the maze, a function named createGrid() is defined. It takes the dimension as the input (we pass d=51) and proceeds to generate a multidimensional array of size d*d. The upper left corner is defined as the start node and the lower right corner is defined as the goal node. The blocks are placed in the grid using the isBlock() function that uses a probability of 0.28 to block the cell. The upper left and lower right cells are left unblocked. After creating the grid with blocks, we need to check if a path exists from the start node to the end node. The unblocked cells are represented by 0 and blocked cells are represented by -1.

What algorithm is most useful for the existence of these paths? why?
To check whether a path exists in the maze, we have used the Depth First Search (DFS) algorithm. As our goal is to just know if a path exists and finding one of many paths is sufficient, the backtracking property of DFS is helpful to find a path. As we need to find a path from one corner to diagonally another corner of the grid, the depth first approach might help find the goal node faster. DFS is also better in terms of space complexity compared to other algorithms such as BFS.

If a path doesn't exist from the start node to the end node in the grid, we keep calling this function until we get a maze that is valid.

2. 2. Ghosts (Creation, Placement, and Movement)

In addition to navigating the maze by avoiding blocked cells, the agent has to deal with ghosts. Ghosts are not in a static position but keep moving at every timestep that the agent moves.

2.2.1 Creating the Ghosts

We have created a function create_ghosts() that places ghosts in positions within the grid. We first find all the reachable nodes from the start node using Depth First Search. Then randomly place N ghosts in different reachable positions within the grid.

2.2.2 Moving the ghosts

The ghosts can move in four directions - left, right, up, down. If the neighbour that the ghost decides to move to is unblocked, the ghost directly moves to it. If the neighbour that the ghost decides to move to is blocked, the ghost might move to that cell with a 0.5 probability. The function move_ghosts() helps with carrying out the above movement of the ghosts at each timestep.

3. Agents

3.1 Movement and Constraints

The agent can move in any one of the four directions - up, down, left, right. The agent must start at the start position (upper left corner) and move to the goal position (lower right corner). The agent, unlike the ghosts, cannot move through blocked cells. The agent movement is restricted within the 51 * 51 grid and cannot move to places which are not bounded by this dimension.

The agent is considered successful if it can move from start to end in such a way that it does not encounter a ghost in the path. If the ghost and agent are in the same cell, the agent dies. Either if the ghost moves to the agent's cell or if the agent moves to the ghost's cell, the agent dies.

3.2 Agent 1

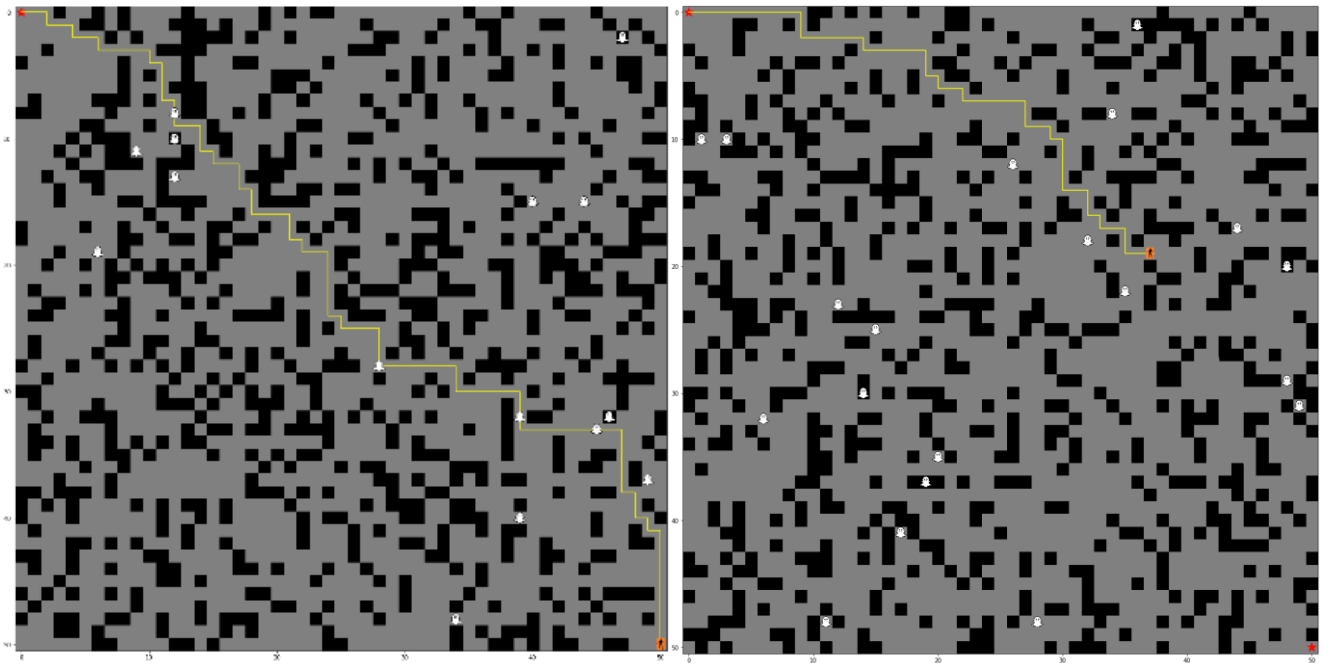
3.2.1 Approach

Agent 1 ignores the ghosts, plans the path once and goes along with the plan. No re-routing is required.

Below are the steps that were followed for Agent 2:

1. Use Breadth First Search to find the shortest path between the start and the end node. As no re-planning is required, BFS can be used here and we won't face any major efficiency issues.
2. Move along the planned path till the agent reaches the goal node.
3. If a ghost appears within the agent's current position on the planned path, Agent 1 dies. Else, agent wins.

3.2.1.1 Visualization

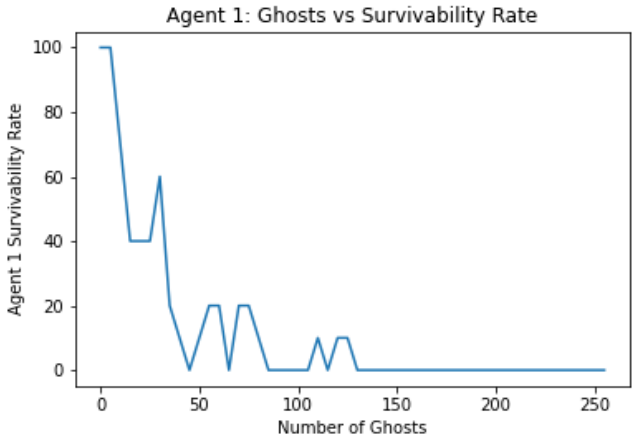


Agent 1 Wins

Agent 1 Dies

The first grid is an example of a maze where Agent 1 wins. Second grid is an example of a situation where the agent was following it's planned path, but the ghost enters the cells and kills the agent.

3.2.2 Survivability



The success rate has been computed by running the agent 1 multiple times on multiple grids for ghosts increasing in increments of 5. At 5, 10, 20 ghosts, the success rate of Agent 1 is at 100, 70, 40 respectively. Agent 1 converges to 0 success rate when the ghosts increase to 130 and beyond.

3.2.3 Limitations

This approach doesn't consider the changing environment of the maze. The ghosts keep moving in the environment and can get into the planned path and kill the ghost.

3.3 Agent 2

3.3.1 Approach

Agent 2 needs to re-plan at every time step. As the ghosts keep moving within the maze at every timestep, agent 2 takes into consideration the changing environment.

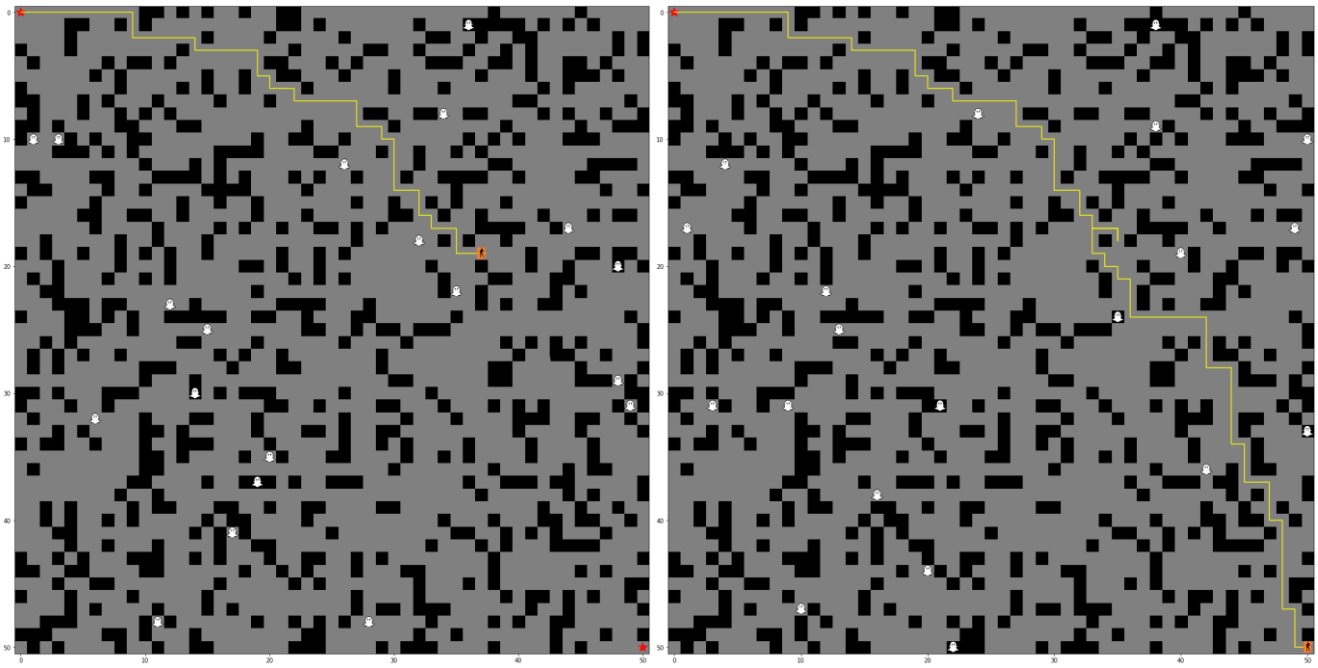
Below is the steps that were followed for Agent 2:

1. Calculate the path from the start node to the goal node using the A* Algorithm which has the Manhattan distance as the heuristic function. Manhattan distance was chosen as the heuristic as the movements of both the agent and the ghost are limited to 4 directions - up, down, left, right. As path search is required to be done multiple times, A* Algorithm was chosen for agent 4 due to the efficiency that the algorithm provides through the heuristic.
2. To cut down on the number of times the path needs to be re-calculated, a condition has been specified. Apart from the ghosts appearing in the planned next move (n), if the ghost appears in the next two moves (n+1, n+2), the path gets re-planned. If the ghosts appear at a node in the planned path that is further away than positions n, n+1, n+2 from the current position of the agent, the path remains the same.
3. If all paths to the goal are blocked, the agent decides to either move away from the nearest ghost or remain in the same position.
    a. The nearest ghost/ ghosts is found out using Manhattan distance from the agent's current position. Then the Manhattan distance is found out from the nearest ghost to the valid neighbours (unblocked and within the grid). The neighbour which is the farthest away from the ghost is chosen as the next move for the agent.
    b. If there are multiple ghosts or if there is a situation where the agent's current position is better than moving to other locations, the agent remains in the same position. To determine if the agent should remain in the same position, Manhattan distance is used.
4. If the agent encounters a ghost in the same cell as the agent, the agent dies. If the agent reaches the goal node, the agent wins.

Agent 2 requires multiple searches - you'll want to ensure that your searches are efficient as possible so they don't take much time. Do you always need to replan? When will the new plan be the same as the old plan, and as such you won't need to recalculate?

To make the searches efficient, A* algorithm is used with Manhattan distance as the heuristic. This helps to reduce the time taken to find a path compared to BFS. We don't always have to replan. We can continue with the same path if there are no ghosts in the next few immediate moves of the planned path or in their neighbourhood.

3.3.3.1 Visualization

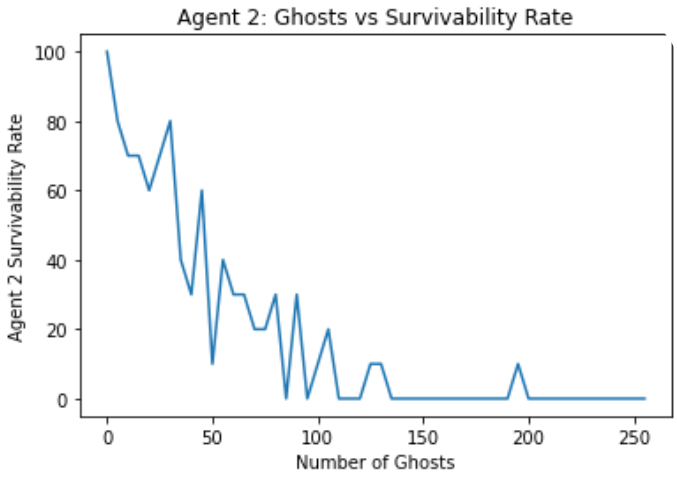


Agent 1 Dies

Agent 2 Wins

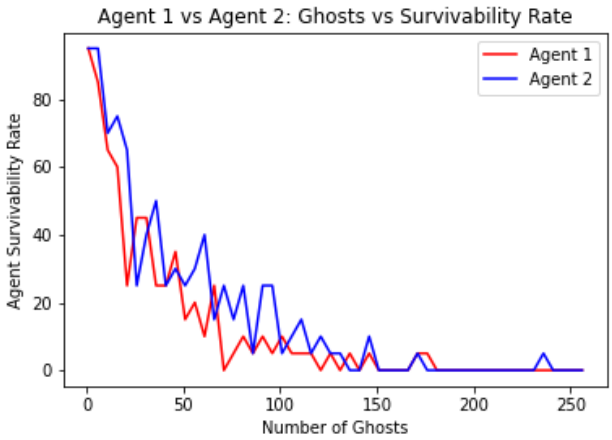
The above is an example of a maze where Agent 1 loses and Agent 1 wins. The same ghost movements are replicated in both mazes. We can see that agent 2 is able to reroute its path and avoid getting killed by the Ghost.

3.3.2 Survivability



The success rate has been computed by running the agent 2 multiple times on multiple grids for ghosts increasing in increments of 5. At 5, 10, 20 ghosts, the success rate of Agent 2 is at 80, 70, 70 respectively. Agent 2 converges to 0 success rate when the ghosts increase to 195 and beyond.

3.3.3 Performance - Agent 2 vs Agent 1



For multiple simulations, both Agent 1 and Agent 2 have been run on the same maze with the same movement of the ghosts. By comparing the performance of both Agent 1 and Agent 2, we can see that Agent 2 has a better performance than Agent 1 in terms of survivability rate.

3.3.4 Limitations

The replanning of the path that the agent requires at every time step t is time consuming and memory consuming. To optimize this, condition has been implemented in the agent 2 to replan only if the ghost is present in the cells of the planned path which are in close vicinity to the agent's current position. However, if the ghosts are present in the neighbours of the nodes in the planned path, agent 2 doesn't take that into consideration.

3.4 Agent 3

3.4.1 Approach

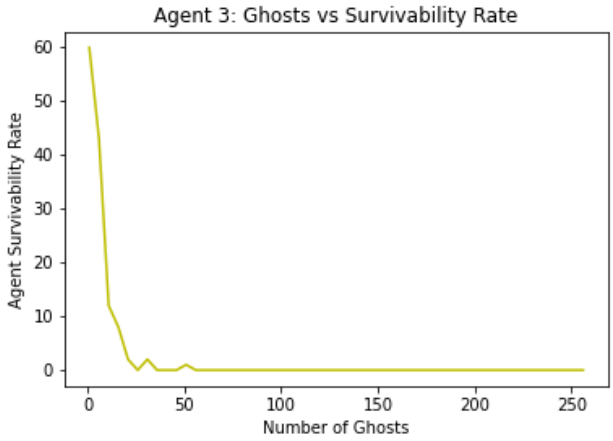
Agent 3 needs the ability to forecast the future based on which it decides on which move to take next.

Below are the steps that were followed for Agent 3:

1. Firstly, we find all the possible valid neighbours that the agent can move to from its current position. Valid neighbours are defined by a cell not being in a blocked state, cell being within the bounds of the grid, and the cell not containing a ghost.
2. Secondly, the success rate of the valid neighbours is computed by calling agent 2 over x=10 simulations. Success rate is calculated by (total_wins/x)*100.
3. We then find the neighbour with the maximum success rate and choose it as the next move. If more than one such neighbour exists, we calculate the distance from the neighbours to the goal node and choose the one which has the minimum distance using Manhattan distance. If the distance to the goal node is equal as well, we choose one of the neighbours as the next move randomly.
4. If the agent encounters a ghost in the same cell as the agent, the agent dies. If the agent reaches the goal node, the agent wins.

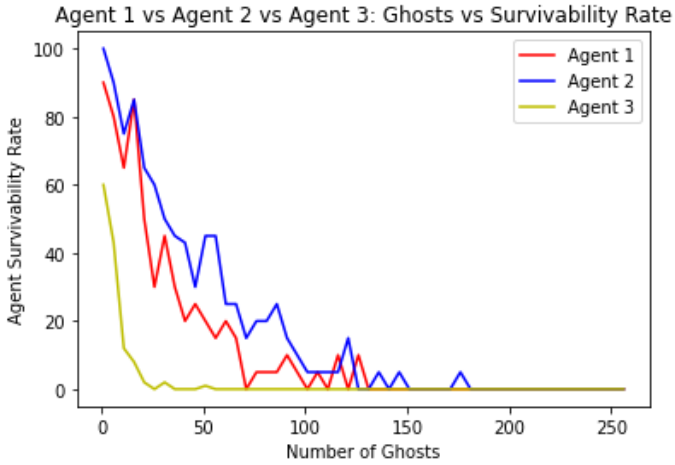
Additionally, if Agent 3 decides there is no successful path in its projected future, what should it do with that information? Does it guarantee that success is impossible?
The agent can use this information to move to other locations (and see if it can find a successful path from there) or remain in its position and wait for the environment to change. No, it does not guarantee that success is impossible. If the environment keeps changing, and the simulation did not simulate this current situation, it's possible that a successful path may exist.

3.4.2 Survivability



From the graph, we notice that there is a steep drop in the survivability very quickly at about 25 ghosts. This outcome is unexpected as Agent 3 is based on multiple simulations of Agent 2.

3.4.3 Performance - Agent 3 vs Agent 2 vs Agent 1



From the graph, we notice that Agent 3 is performing poorly compared to other agents. We notice that survivability goes down to 0 at about 25 ghosts itself for the agent. While for other agents, survivability goes down at a later stage.

3.4.4 Limitations

If you are unable to get Agent 3 to beat Agent 2 - why? Theoretically, Agent 3 has all the information that Agent 2 has, and more. So why does using this information as Agent 3 does fail to be helpful?
While using this approach, it is observed that most times, the agent wiggles back and forth between nodes and gets stuck there. It is also observed that the agent sometimes gets stuck in a cyclic path due to which it never reaches the goal and eventually dies when ghosts enter the cyclic path. The multiple simulations that are required to compute the success rate at each neighbour node are time consuming and require more computational effort. As the environment keeps changing in the maze and the possibility of a lot of paths and lot of ghost movements, moving according to success rate may not be enough. More simulations to compute the success rate may help to improve the results. Adding conditions based on penalty for the agent if it visits a cell too many times may help with the wiggle problem.

3.5 Agent 4

3.5.1 Approach

Agent 4 has been implemented as an extension to Agent 2. In addition to replanning when the ghosts are in the next 3 positions of the planned path, Agent 4 replans when ghosts are present in the neighbourhood of the next three planned moves.

Below are the steps that were followed for Agent 4:

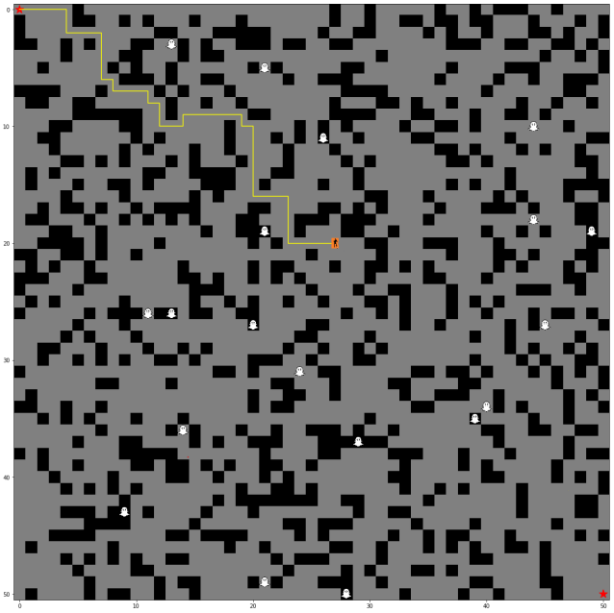
1. Calculate the path from the start node to the goal node using the A* Algorithm which has the Manhattan distance as the heuristic function. Manhattan distance was chosen as the heuristic as the movements of both the agent and the ghost are limited to 4 directions - up, down, left, right. As path search is required to be done multiple times, A* Algorithm was chosen for agent 2 due to the efficiency that the algorithm provides through the heuristic.
2. To cut down on the number of times the path needs to be re-calculated, a condition has been specified. Apart from the ghosts appearing in the planned next move (n), if the ghost appears in the next two moves (n+1, n+2), the path gets re-planned. If the ghosts appear at a node in the planned path that is further away than positions n, n+1, n+2 from the current position of the agent, the path remains the same.
3. Agent 4 also checks the neighbourhood of the planned next move (n), the next two moves (n+1 & n+2). If the ghosts are at a Manhattan distance of 1 from any of the next three moves, agent 4 finds a new path to take.
4. If all paths to the goal are blocked, the agent decides to either move away from the nearest ghost or remain in the same position.
    a. The nearest ghost/ ghosts is found out using Manhattan distance from the agent's current position. Then the Manhattan distance is found out from the nearest ghost to the valid

neighbours (unblocked and within the grid). The neighbour which is the farthest away from the ghost is chosen as the next move for the agent.

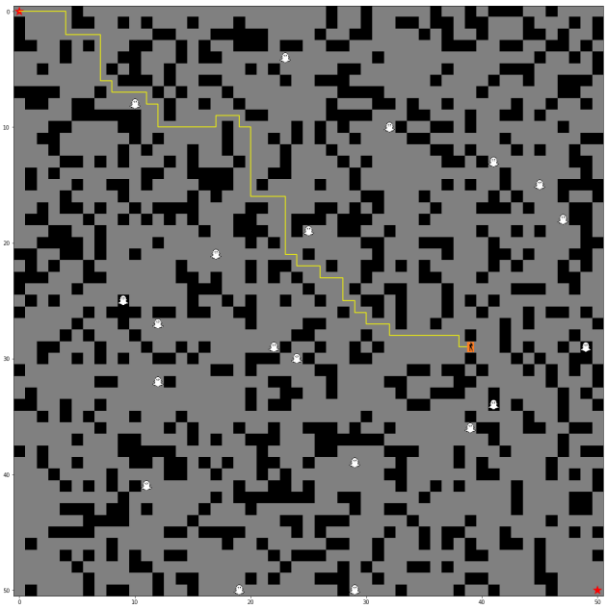
b. If there are multiple ghosts or if there is a situation where the agent's current position is better than moving to other locations, the agent remains in the same position. To determine if the agent should remain in the same position, Manhattan distance is used.

5. If the agent encounters a ghost in the same cell as the agent, the agent dies. If the agent reaches the goal node, the agent wins.

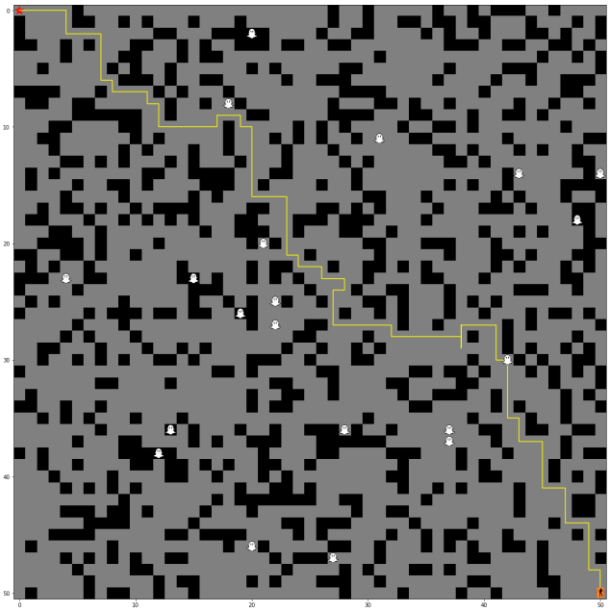
3.5.5.1 Visualization



Agent 1 Dies



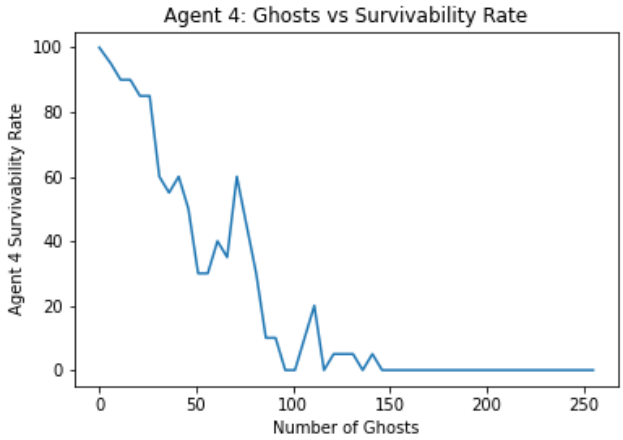
Agent 2 Dies



Agent 3 Wins

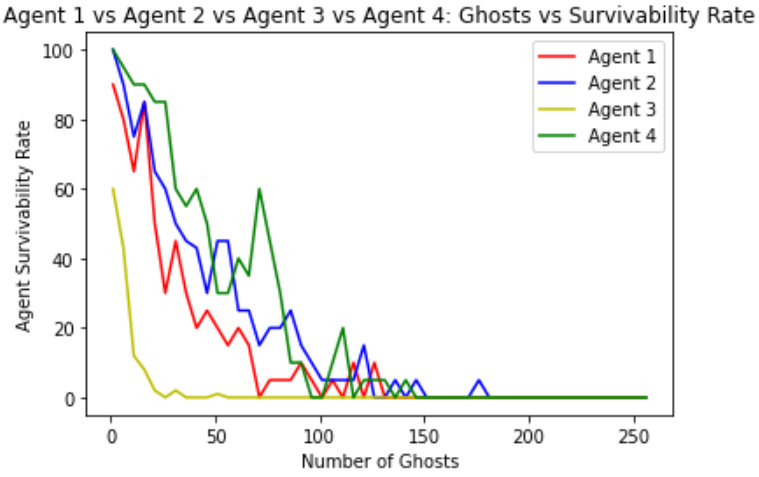
The above figure is an example of a grid and ghost movements where Agent 1 and Agent 2 fail, but Agent 4 succeeds. While Agent 1 ignores the ghosts, Agent 2 checks if the ghosts are present in the planned path. Agent 4 also checks if the ghosts are present in the neighbourhood of the planned path and reroutes accordingly.

3.5.2 Survivability



In this graph we can see that the survivability goes down to 0 at ghosts = 145. For ghosts<= 25, the survivability is higher than 80.

3.5.3 Performance - Agent 4 vs Agent 3 vs Agent 2 vs Agent 1



As we can see from the above graph, Agent 4 performs slightly better than Agent 2. It performs visibility better than Agent 1.

3.5.4 Limitations

Agent 4 checks if the ghosts are present in the next moves or if they are present in the neighbourhood of the next planned moves or if they are present in the immediate neighbourhood of the agent. When the ghosts increase, the agent gets trapped and its chances to win get reduced. Agent 4 can be incorporated with forecasting capabilities that can guide it to prevent a situation where it is less likely to reach the goal node successfully.

3.6 Agent 5

3.6.1 Approach

Agent 5 has been implemented as a modification to Agent 4. In addition to replanning when the ghosts are in the next 3 positions of the planned path, Agent 5 replans when ghosts are present in the neighbourhood of the next three planned moves. The agent can't see the ghosts if they are present in the blocked cells.

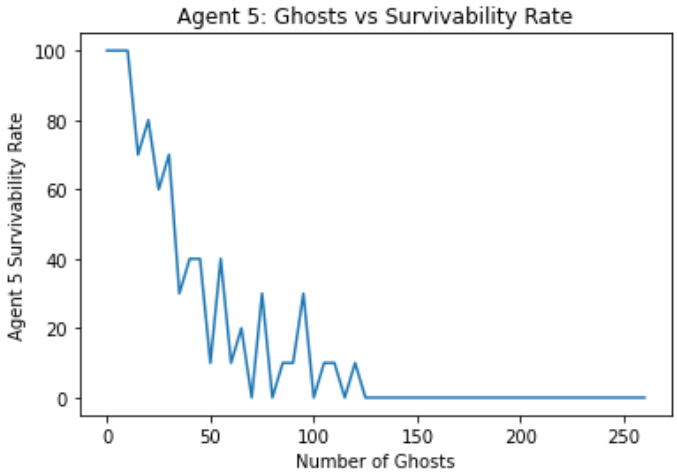
Below are the steps that were followed for Agent 5:

1. Calculate the path from the start node to the goal node using the A* Algorithm which has the Manhattan distance as the heuristic function. Manhattan distance was chosen as the heuristic as the movements of both the agent and the ghost are limited to 4 directions - up, down, left, right. As path search is required to be done multiple times, A* Algorithm was chosen for agent 2 due to the efficiency that the algorithm provides through the heuristic.
2. To cut down on the number of times the path needs to be re-calculated, a condition has been specified. Apart from the ghosts appearing in the planned next move (n), if the ghost appears in the next two moves (n+1, n+2), the path gets re-planned. If the ghosts appear at a node in the planned path that is further away than positions n, n+1, n+2 from the current position of the agent, the path remains the same.
3. Agent 5 also checks the neighbourhood of the planned next move (n), the next two moves (n+1 & n+2). If the ghosts are at a Manhattan distance of 1 from any of the next three moves, agent 4 finds a new path to take.
4. If all paths to the goal are blocked, the agent decides to either move away from the nearest ghost or remain in the same position.
    a. The nearest ghost/ ghosts is found out using Manhattan distance from the agent's current position. Then the Manhattan distance is found out from the nearest ghost to the valid neighbours (unblocked and within the grid). The neighbour which is the farthest away from the ghost is chosen as the next move for the agent.
    b. If there are multiple ghosts or if there is a situation where the agent's current position is better than moving to other locations, the agent remains in the same position. To determine if the agent should remain in the same position, Manhattan distance is used.
5. If the agent encounters a ghost in the same cell as the agent, the agent dies. If the agent reaches the goal node, the agent wins.

Agent 5 can be further improved by adding conditions such as -

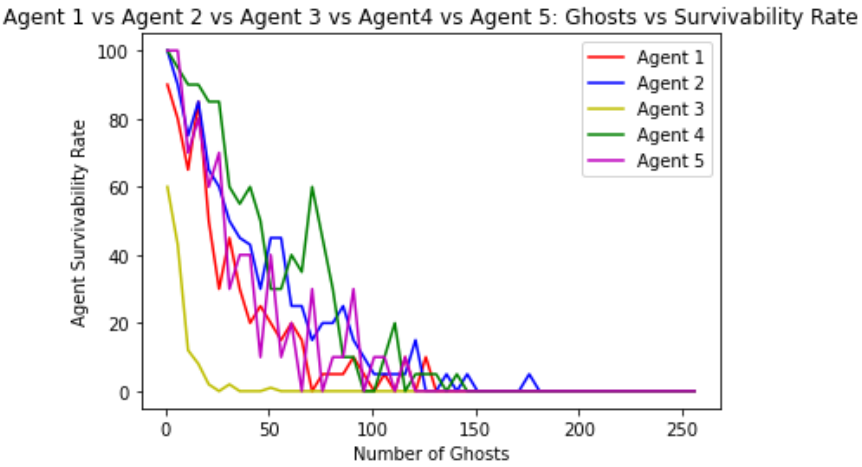
1. The agent avoids the cells adjacent to blocked cells.
2. If the agent already knows the total number of ghosts (n) in the maze, if x ghosts are in un-blocked cells, the agent knows that there are (n-x) ghosts in the blocked cells (but doesn't know where these ghosts are present). Based on this information, the probability of danger of visiting a cell adjacent to a blocked cell can be computed, and the further path can be planned.

3.6.2 Survivability



Here we notice that the survivability goes down to zero at ghosts =125. The performance drops after ghosts increase more than 25.

3.6.3 Performance - Agent 1/2/3/4/5



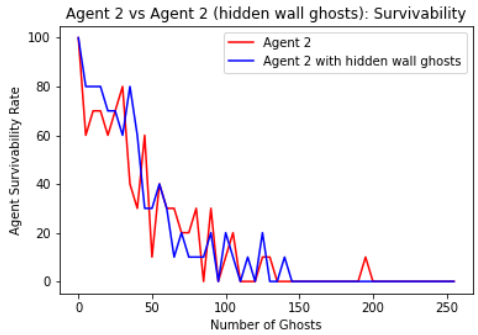
In the above graph, we can see the performances of all the five agents together. We can see that Agent 4 outforms all the other agents. Agent 5 fails to perform better than Agent 2 and Agent 4.

3.7 Redoing Agent 1/2/3/4 with Ghosts in Blocked Cell invisible to Agent Condition

3.7.1 Agent 1

Agent 1 does not get affected by the condition of ghosts in blocked cells not being visible to the Agent. As the agent finds a path around the blocked cells and ignores the ghosts, the performance of the agent remains unaffected.

3.7.2 Agent 2

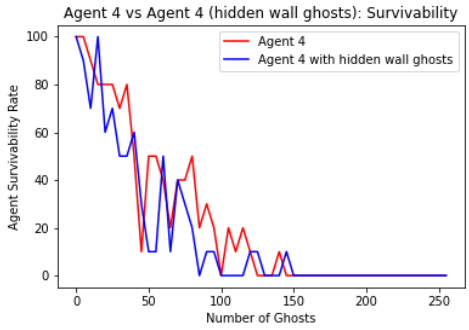


Agent 2 has a feature to move away from the nearest ghost if a path to the goal node does not exist. If the ghost in the blocked cell is close to the agent, the agent can't see it and the agent may die in the next steps.

3.7.2 Agent 3

As agent 3 works by running multiple simulations of agent 2, it gets affected by the same amount that agent 2 get affected by the new condition.

3.7.2 Agent 4



As Agent 4 checks the neighbourhood of the planned path for the presence of ghosts, Agent 4 also gets affected by not being able to see the ghosts in the blocked cells. From the graph, we can see that the performance of the agent comes down once this new condition is introduced.

4. Conclusion

We have applied and implemented multiple path searching algorithms such as Depth First Search, Breadth First Search, and the A* Algorithm. We have implemented forecasting capabilities for Agent 3 so that decisions can be made based on the success rate. From all the agents that are implemented, Agent 4 gives out the best performance followed by Agent 2. Incorporating information such as distances between agents and ghosts combined with algorithms designed to find a good path helped improve the performance of the agent.