Project 1: Ghosts in the Maze

Lab report

1.1 The Environment

To create the maze, we chose to use the data structure of a two-dimensional numpy array. This 2D array was chosen to represent the 51x51 grid.

The values in the maze are representing states of the cells in the maze. The states could either be blocked cell, unblocked cell, goal cell, ghost in a blocked cell, ghost in goal state, ghost in normal cell or 2 or more ghosts in a cell. We selected integers to represent these states in the 2D array and whatever value the cell contains tells us about the current state of the cell in the maze. Integers were selected for their ease of representation, however the numbers in their own were randomly selected.

Following integers represent the following states:

0: blocked cell
1: unblocked cell
2: goal cell
7: ghost on normal cell
8: multiple ghosts in a cell
10: ghost on a normal, unblocked cell
>10: multiple ghosts in a cell

We traverse each cell of the array and generate a random number between 0 and 1. If this number is less than 0.28, we make the cell blocked. If not, the cell remains unblocked and is represented by 1. Then the start and the goal cells are unblocked if they were initially blocked.

Checking quality of the maze

We check if the maze is “too blocked” by using BFS. If there is a path that can be followed to find the goal cell, then it is not too blocked. But, if there is no such path, that maze is discarded. If the maze is not too blocked it is accepted and we proceed.

Q. What algorithm is most useful for checking for the existence of these paths? Why?

The algorithm that is most useful for checking for the existence of these paths is DFS. This is because it returns if a path exists in the shortest time, making it the most efficient.

Our code runs BFS to check for “too blocked” since this is the algorithm that is used throughout the processes.

1.2 Ghosts

Our module in python, `floodfill()` returns cells that are reachable from the start node. This helps in generating cells that can contain ghosts. Ghosts will be spawned only at these cells that are accessible by the start node that is the (0,0) node and hence returned by `floodfill()`.
According to the number of ghosts, we choose random cells returned by `floodfill()`. If the chosen cell is the goal cell, we denote it as 8. Or else if it is an unblocked cell, we denote it as 10.

For moving ghosts, we access the list containing the coordinates of these ghosts. Then it randomly selects a move from the available moves. If the selected cell is unblocked, blocked, goal we denote it as 10,7, and 8 respectively. If the selected cell already has a ghost, then the value in the cell is added by 10. Accordingly the previous ghost cell is reset.

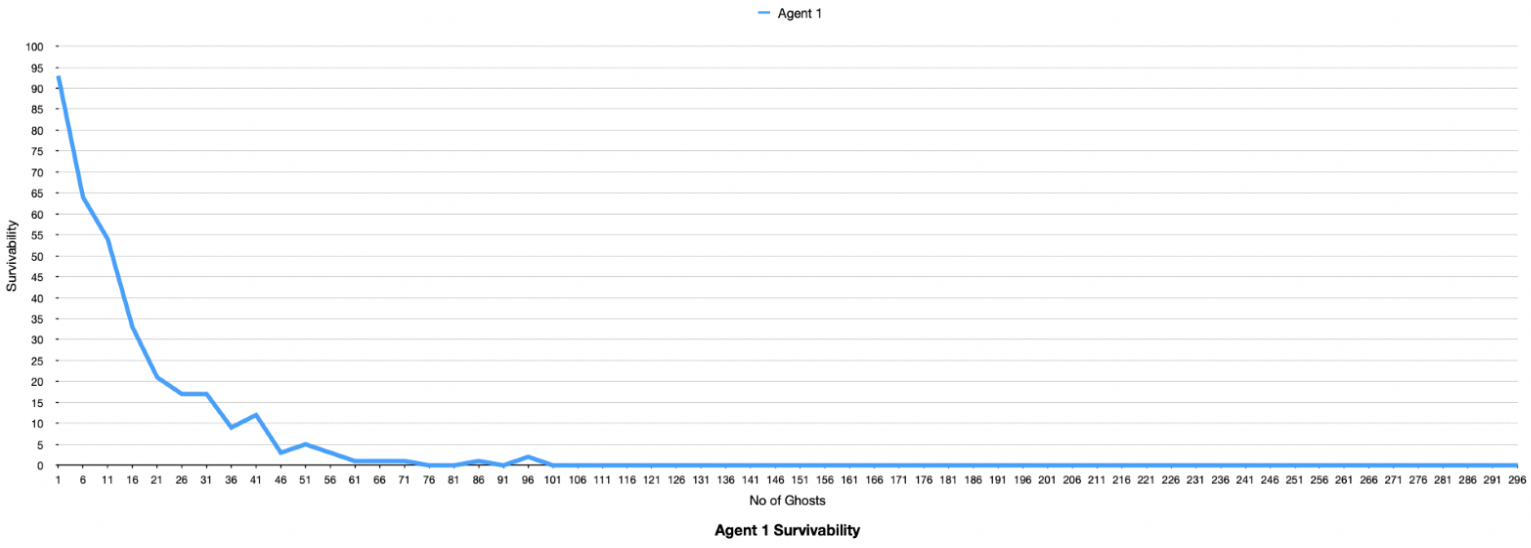
1.3 The Agents

1.3.1 Agent 1

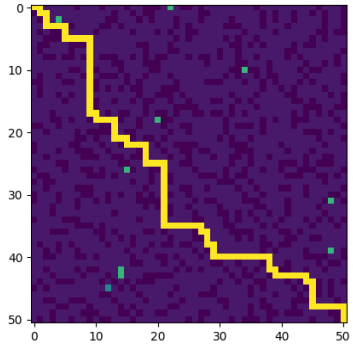
Number of mazes: 100
Ghosts: Total 300 ghosts, starting from one with increments of five at every randomly generated 100 mazes

This first agent sets to find the shortest path from initial cell to the goal cell. It does so by using BFS technique. It does not change its initial plan in the future, as it follows the exact path it set out to in the beginning. As it follows the path, it checks the cell for ghosts, if there is a ghost in the cell it dies, and if not, it proceeds ahead with the initial plan. The agent checks the state of the cells again after the ghosts move to make the decision to proceed or die according to the state.

Performance



Example of maze of Agent 1



1.3.2 Agent 2

Number of mazes: 100

Ghosts: Total 300 ghosts, starting from one with increments of five at every randomly generated 100 mazes

The second agent operates slightly differently from the first, where it chooses the shortest path to the goal cell but also checks that the path has no ghosts. The decision on what move to make and to what cell is made by the agent using the following criteria:

If there is no path with no ghosts, the agent has four choices to move in cells: top, down, left or right. Here, the agent finds the nearest ghost using the Euclidean distance formula:

$dist((x, y), (a, b)) = \sqrt{((x - a)^2 + (y - b)^2)}$

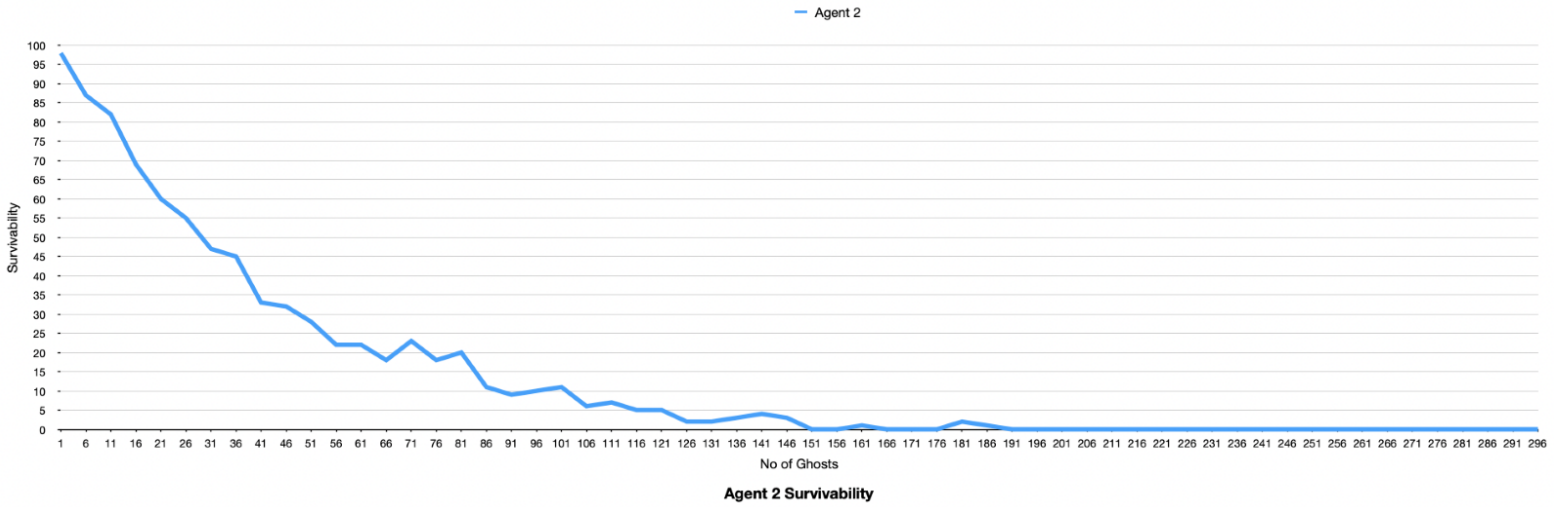
It then proceeds to run away from the nearest ghost as far as it can. This farthest distance from the nearest ghost is also defined by using the Euclidean distance.

Once the agent is far away from the nearest ghost, it checks the shortest distance with no ghosts to the goal cell using BFS. If it finds such a path, it proceeds or else repeats the process of running away from the nearest ghost again.

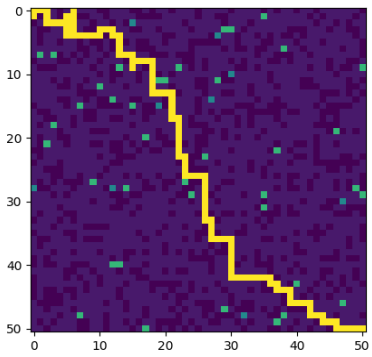
Q. Do you always need to replan? When will the new plan be the same as the old plan, and as such you won't need to recalculate?

We do not need to replan every time. This happens when the path found by Agent 2 has no ghosts in the path at all.

Performance



Example of maze of Agent 2



1.3.3 Agent 3

Number of mazes: 10

Ghosts: Total 80 ghosts, starting from one with increments of 5 at each randomly generated 10 mazes

The agent 3 works in a way that it calls agent 2 for all possible cells that it could take as a path that is, for all adjacent, top, bottom, left and right and current cell, 10 simulations each and whichever cell has the highest survivability, it chooses that cell and proceeds to repeat the process for the chosen cell till it reaches its goal state.

Q. 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?

When Agent 3 sees that there is no survivable path, then it behaves like Agent 2 and runs away from the nearest ghost, as far away as possible. These distances are described by the Euclidean

Distance formula. And even if the Agent 3 cannot see a survivable path right now, that does not mean success is impossible because the ghosts might move out of the way in the future.

Q. 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 fail to be helpful?

We are unable to beat agent 2 in performance even though theoretically agent 3 has more information than agent 2, this is because due to the time that it would take to run Agent 3 to the number of simulations that the data collected would be helpful in deciding a successful path, we have to bring down the number of simulations to just 5. This does not give enough data to Agent 3 to calculate a successful path.

Problems faced: As Agent 3 simulates Agent 2 in all directions before it makes a decision, this takes too much time to collect helpful data and so, we have to reduce the number of simulations to 5 and hence, Agent 3's survivability falls to 0 very quickly.

Reason for result: The number of simulations run for agent 3 is not enough for it to gather accurate information to get a good path.

1.3.4 Agent 4

Number of mazes: 100

Ghosts: Total 300 ghosts, starting from one with increments of five at every randomly generated 100 mazes

We worked towards making Agent 4 perform better than the preceding agents by filling the possible holes in the working idea of the agents.

The possible shortcomings of the agents 1, 2 and 3:

1. Agent 1 does not consider the ghosts into its decision to whether or not to take the path and this reduces its survivability by a lot.
2. While Agent 2 and 3 do take the ghosts into account before choosing the path, they do so by considering even those ghosts that do not affect them immediately, that is they are present in the path but might not be present later when they reach that point. This is an unnecessary step that reduces the possible efficiency of the agents.
3. Agents 2 and 3 have another possible flaw. They don't take into account their decreased chances of survival due to the ghosts that may be present in the neighboring cells of the path cells that they are choosing. This decreases the survivability of these agents as they might be killed by a neighboring cell ghost after it moves to the decided cell.

Q. Why did it succeed, or why did it fail?

By working on trying to resolve these issues in the approach taken by Agent 4, we increased the chances of survival of the agent. We did this by making the following modifications:

1. Agent 4 shields itself from all adjacent to the next possible cell ghosts whenever it possibly can, that is it checks the next possible cell for ghosts and it also checks the adjacent cells to it for the possibility of ghosts to protect it from possibly dying in the next steps.
2. Agent 4 doesn't let the ghosts far from it affect its current decision. This helps by letting it stay on the shortest possible path for a longer time and hence improving its efficiency.

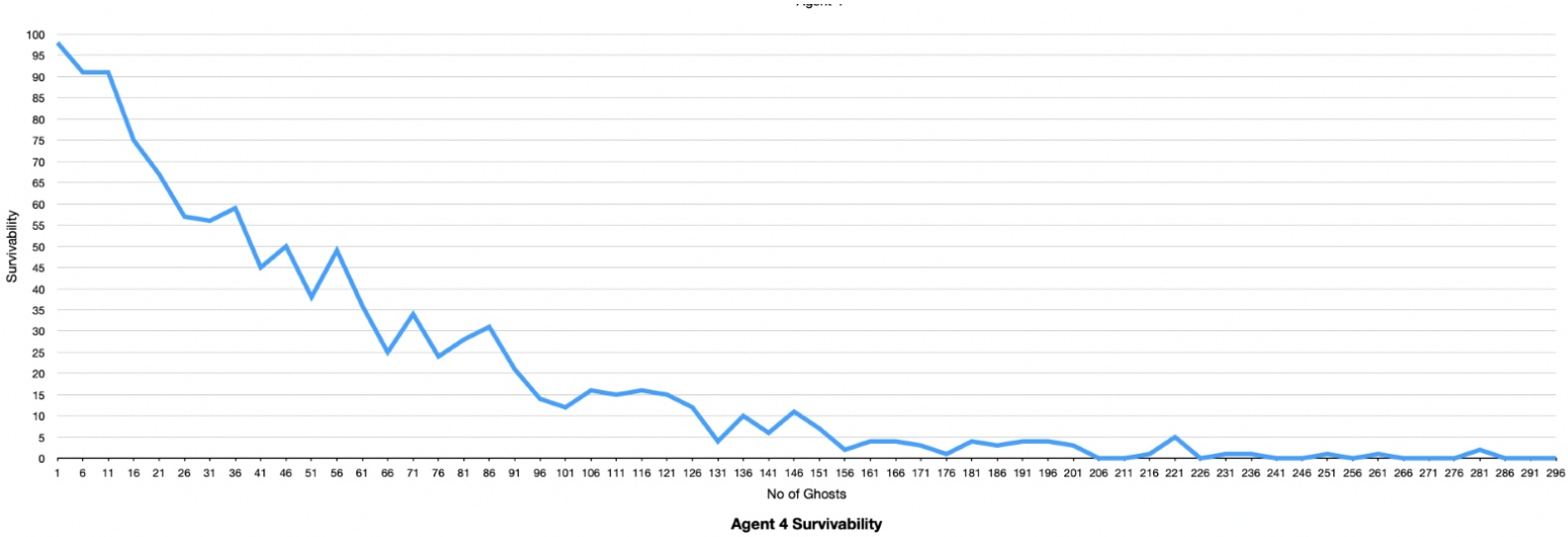
Agent 4 works in the following way:

It finds the shortest path to the goal cell using BFS. It checks the next possible cell's adjacent cells and their neighbors for ghosts. If it is able to find a cell here with no ghost, it decides to go to that cell. If there is no such cell with no ghost neighbors, it randomly chooses either the top, bottom, left or right cells. Then it proceeds to find the shortest path to the goal cell from this cell and repeats the above steps till it either dies or reaches its goal state.

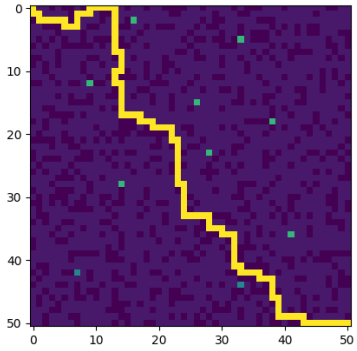
Q. How did your Agent 4 stack up against the others?

Agent 4 beats Agent 1 and 2 every time and hence is successfully a better agent with better survivability.

Performance



Example of maze of Agent 4



1.3.5 Low information environment

Number of mazes: 100

Ghosts: Total 300 ghosts, starting from one with increments of five at every randomly generated 100 mazes

Q. Redo the above, but assuming that the agent loses sight of ghosts when they are in the walls, and cannot make decisions based on the location of these ghosts. How does this affect the performance of each agent?

In this low information environment where the agents cannot see the ghosts in the blocked cells, we realized that this would not impact the Agents 1, 2 and 3 in anyway because:

1. Agent 1 does not check for ghosts in its path. This means agent 1 wouldn't be affected with the lack of knowledge about the blocked cell ghosts. It will function in the way it normally would, that is it would find the shortest path using BFS and take that path.
2. Agent 2 checks if a possible path has ghosts but the path that it selects will never have a blocked cell and since agent 2 does not check adjacent cells for ghosts, agent 2 will not be affected either by this lack of knowledge.
3. Agent 3 is based on agent 2 hence would also not be affected by the lack of knowledge and will get unaffected results in this environment, similar to the previous environment.
4. Agent 4 will be affected by this low information environment as it uses the information of the ghosts in adjacent cells to decide which path to take. If it does not know if a ghost is present in a blocked cell, its decision will be less involved and its survivability will decrease.

1.3.6 Agent 5

Number of mazes: 100

Ghosts: Total 300 ghosts, starting from one with increments of five at every randomly generated 100 mazes

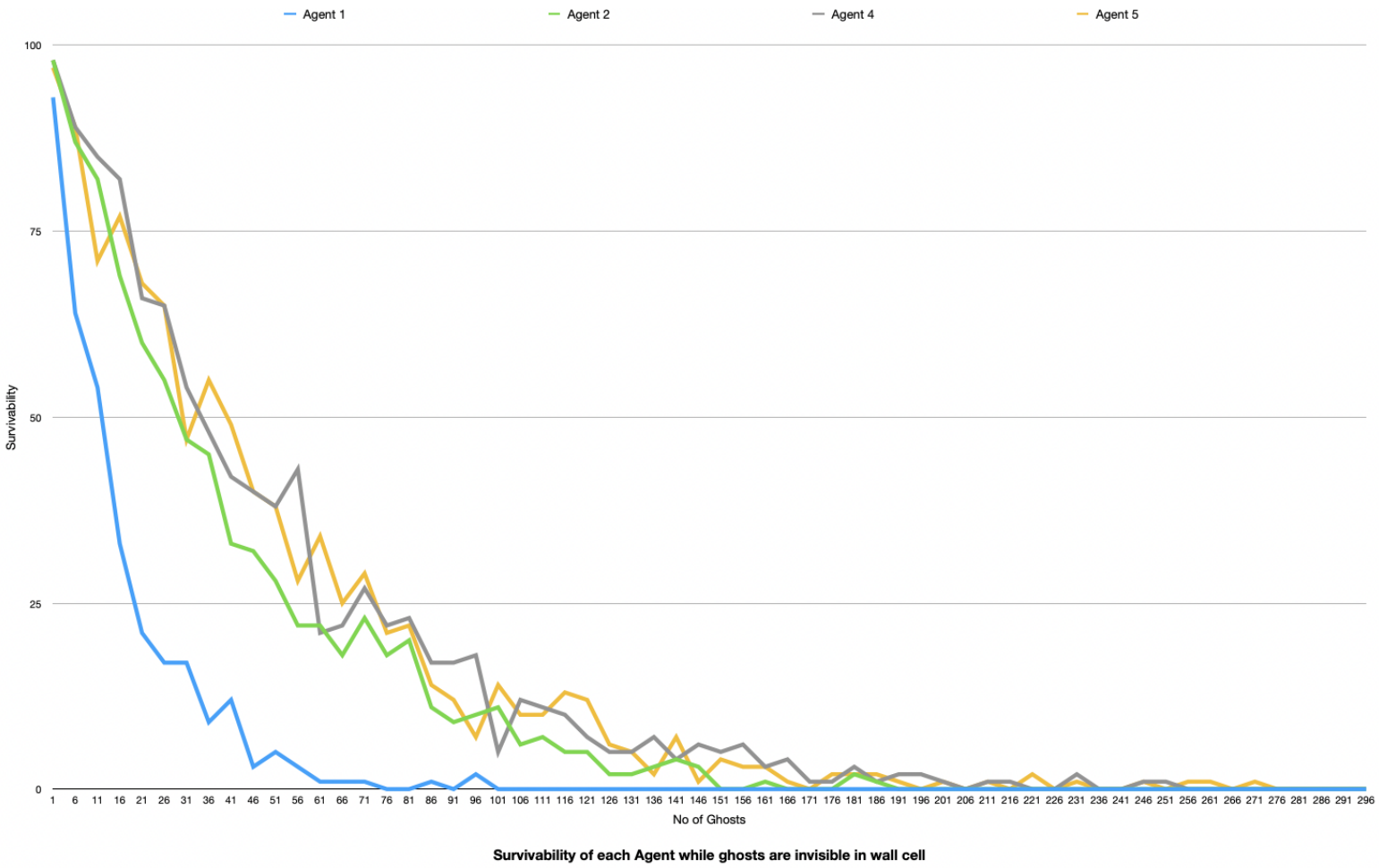
Q. Build an Agent 5 better suited to this lower-information environment. What changes do you have to make in order to accomplish this?

An Agent 5 that is better suited to the low information environment required certain changes to be made in the existing Agent 4. The Agent5 works like Agent 4, that is it shields itself from all the ghosts by staying away from cells whose neighbors also might have ghosts. But since in this low information environment we did not know if a particular blocked cell has a ghost or not, we changed it to check not only if there are ghosts in the adjacent cells or not but also if the adjacent cells are blocked. If any of the two are true it will prefer not taking the cell with such neighbors and hence even with low information, it is able to avoid the blocked cell ghosts.

Performance of Agents in low information environment

Ghosts_invisible_results

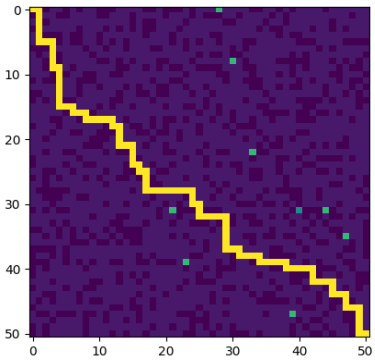
No of Ghosts	Agent 1	Agent 2	Agent 4	Agent 5
1	93	98	98	97
6	64	87	89	89
11	54	82	85	71
16	33	69	82	77
21	21	60	66	68
26	17	55	65	65
31	17	47	54	47



Survivability of each Agent while ghosts are invisible in wall cell

Here, we can conclude that the performance of 1,2, and 3 is unaffected while Agent 4's survivability has decreased.

Example of maze of Agent 5



1.4 Computational problems

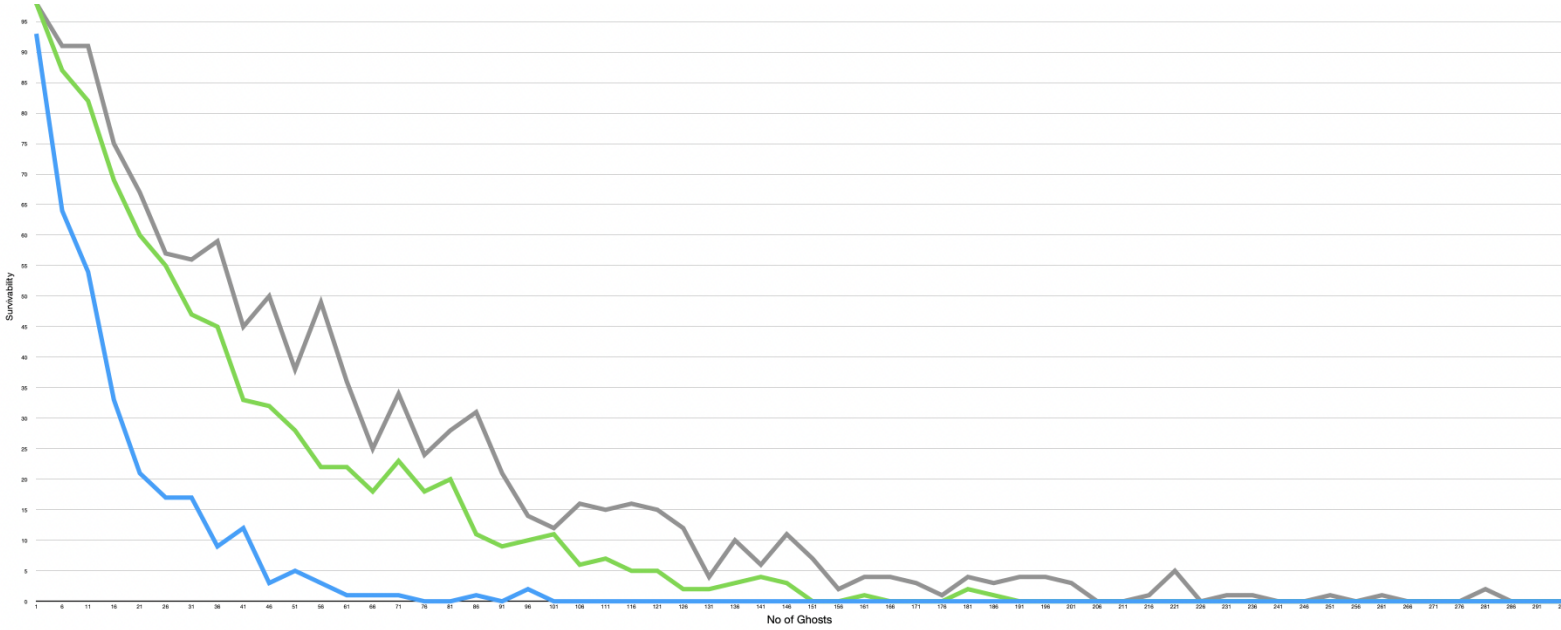
We faced computational problems while working on the agents and trying to collect the data in the given amount of time. Since agent 3 after every step runs an agent 2 simulation in every possible direction, we faced the issue of collecting data as every simulation would take a great amount of time.
To resolve this issue, we first decided to run our processes using multi-threading. But this too, was taking a long time as it just engages only one CPU of the laptop at a time. To overcome this problem and reduce the time it would take to collect data from agent 3, we decided to use multi-processing which uses 4 CPUs of the laptop and runs several processes simultaneously to reduce the time it took to collect data.
We also decided to reduce the number of simulations for data collection from agent 3 to 5 simulations to reduce the time.

1.5 Analysis: Performance Comparison

The following is the comparative graph of agents 1, 2 and 4's survivability run for 100 mazes.

results

No of Ghosts	Agent 1	Agent 2	Agent 4
1	93	98	98
6	64	87	91
11	54	82	91
16	33	69	75
21	21	60	67
26	17	55	57
31	17	47	56



Comparisons of every agent survivability in 100 maze

We can see from the graph that Agent 4 beats Agent 1 and 2 every time and hence is successfully a better agent with better survivability.
Agent 2 beats Agent 1 every time as it has better survivability due to the fact that it takes ghosts into consideration at each step.

Q. When does the survivability fall to 0?
Agent 1's survivability falls to 0 at 101 ghosts
Agent 2's survivability falls to 0 at 151 ghosts
Agent 4's survivability falls to 0 at 206 ghosts
Agent 5's survivability falls to 0 at 171 ghosts

The following is the comparative graph of agents 1, 2, 3 and 4's survivability run for 10 mazes.

10_maze_results

No of Ghosts	Agent 1	Agent 2	Agent 3	Agent 4
1	9	10	2	10
6	7	7	3	10
11	7	7	5	9
16	2	4	4	7
21	1	3	2	6
26	5	5	1	8
31	1	5	0	5