Ghosts in the Machine!

Priyanshu Shrivastava (ps1276), Nikhil Tammina (nt444)

1 Introduction

This project empathizes with how different agents with distinct features search for a solution in the best optimal path. The entire project starts with making a maze which consists of 51 X 51 blocks. Although there are ghosts that are moving randomly, the search algorithm we used, A*, is an informed search algorithm that is actually the best algorithm in this project because it finds the shortest path based on past information and heuristic information to make the best decisions, indeed "intelligent decisions" dodging all the ghosts. Every model in the world today works like these agents, which deploy models in environments with hazy knowledge and ask them to find the best possible outcomes, eventually finding a robust solution. The ghosts in this project represent possible hurdles that we face in daily life and agents are like the partial information that we have and we need to deploy a model to solve real-life problems with the most optimal solution.

2 Design Choices

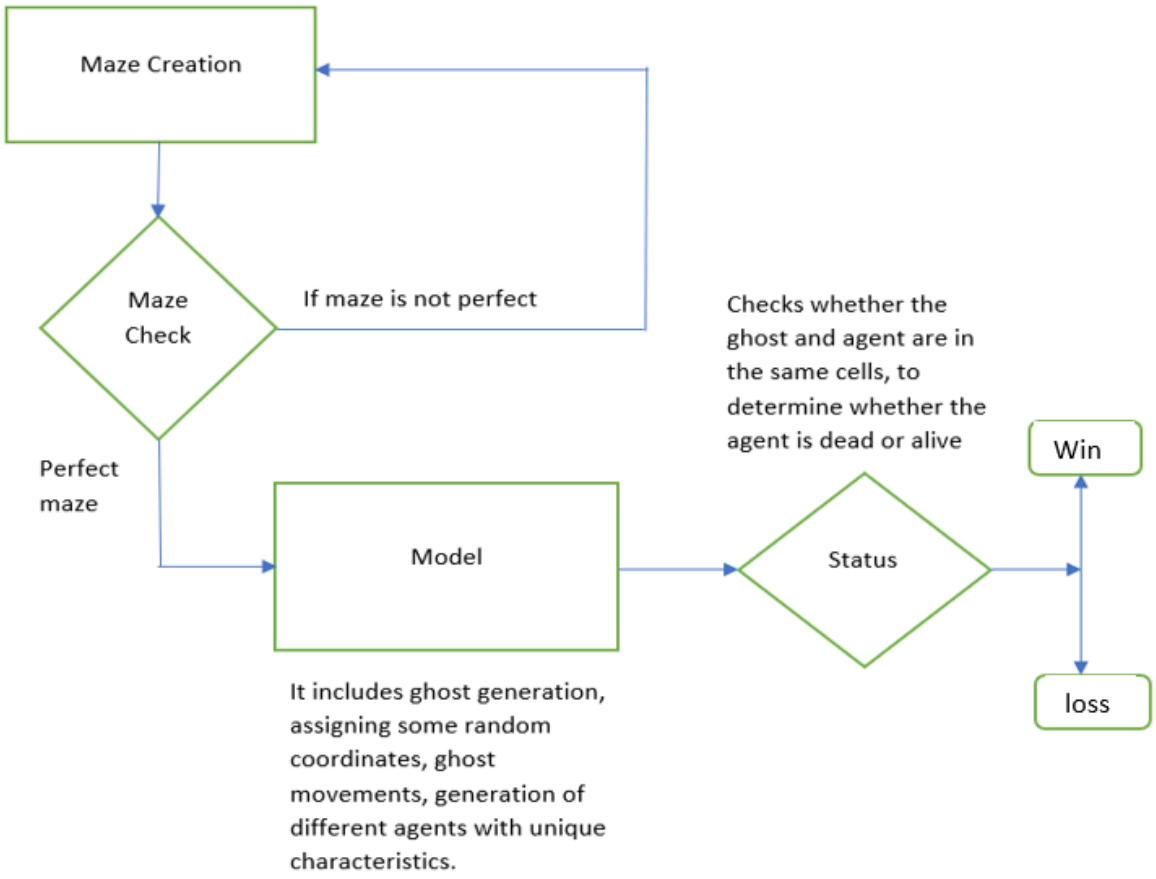

Figure - 1: Model Design

To begin with, we started the project by creating a 53 X 53 maze by putting an Iron wall around the border of the maze which prevents agents or ghosts from entering or passing through it. So technically we have a 51 X 51 grid world, in which we allotted blocks of probability 0.28. The next important thing after the creation of the maze is checking whether the maze is perfect or not from the start node to the goal node (from extreme top left to top right). In order to achieve the above criteria, we used the A* algorithm for backtracking the path from the goal node to the start node, if the maze is perfect i.e if it contains an empty path without blocks in the grid from the start node to the goal node, then the algorithm continues to next steps, or else the algorithm prints that the maze is not perfect and the agent stops searching the path. Ghosts have been randomly spawned throughout the maze with the restriction that they only spawn where there are no blocks. Even though

ghosts can only move in a few directions (top, bottom, left, and right), we added another layer of complexity by giving ghosts a probability of 0.5 to either stay in the same place or move in the direction of a block if one is there. In order to update the movement of the ghost, we copy the original maze, add the coordinate of the new position and discard the old maze and this loop is repeated.

Q1 What algorithm is most useful for checking for the existence of these paths? Why?

A* is the most useful algorithm for our project to check whether a path exists from the top left to the bottom right. The main reason we have opted for this algorithm over BFS, DFS, or Dijkstra's is that - firstly, BFS takes a lot of time and space in order to find the path from the start node to the goal node, hence it can be safely eliminated. DFS has the capability to find the shortest path, but it is not optimal since we may not get the goal node all the time, so it is eliminated. It is somewhat difficult to choose which algorithm is better especially when it comes to A* and Dijkstra's because both will give the same optimal path by choosing the least weighted graphs. However, A* contains a heuristic function (estimate of the path to the goal node) along with the cost of the path to the present node from the start node, while Dijkstra's on the other hand doesn't have a heuristic function and solely focus on cost it has so far, hence it is less effective as it explores more nodes than what A*does. Therefore, we have used the A* algorithm.

Agent 1

Agent 1 is done without any knowledge of ghosts and hence its survivability completely relies on the number of ghosts and luck. It dies once the both agent and the ghost land in the same cell.

A scenario of how our code performs:

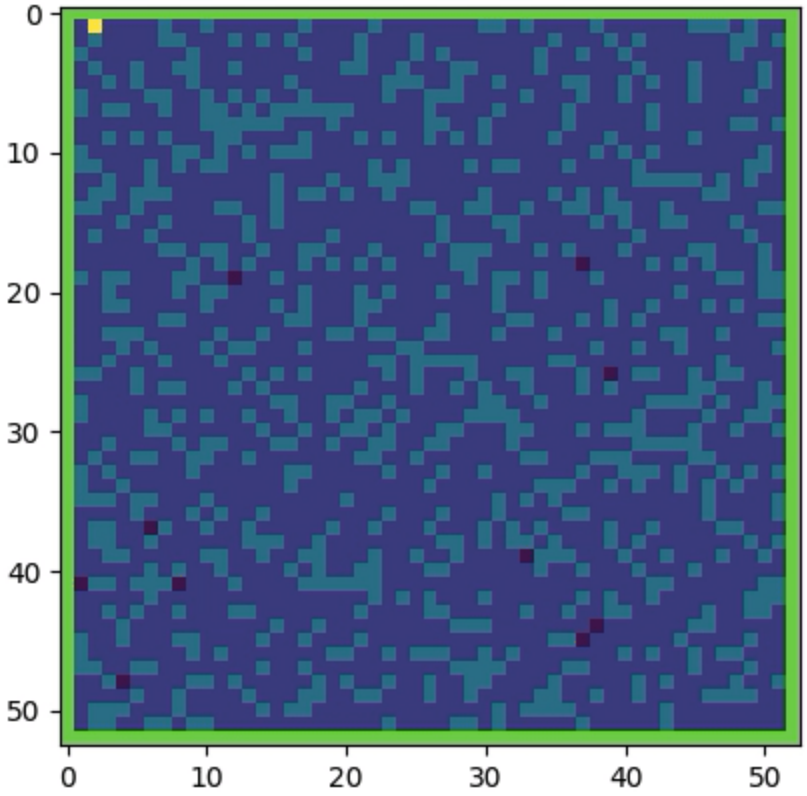

Figure - 2: GIF of Agent-1 Simulation

For creating agent 1, we used the A* star algorithm for our agent irrespective of the ghost's spawned locations or movements of the ghost. Hence, the agent reaches the goal node from the start node without the knowledge of the ghost's movement. However, it dies when the agent and ghost both settle in a single cell.

Agent 2

Agent 2 is more effective than agent 1 as it keeps in mind the ghost's movements and replans its path accordingly. It calls the A* algorithm every time it finds a ghost in its path only when the manhattan distance between the ghost and agent is less than or equal to 15 cells. Furthermore, it chooses its neighboring cells (top, left, bottom, right) based on which neighboring cell is farthest away from the ghost and the agent occupies that cell. For example, let's consider that our agent is lying on (1,1) and the ghost is lying somewhere in (2,3) (less than 15 magnitudes away), so the next neighboring cell the agent chooses is the farthest cell from the ghost cell i.e it will choose a random coordinate from (0,1) or (1,0) as both are equally far away from the ghost.

Q2 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?

In order to keep agent 2's search efficiency from being compromised, we coded the agent in such a way that it need not have to replan every time step, instead, it will replan only when the manhattan distance between the cell in which the agent is lying and the ghost is less than 15 magnitudes away. Hence the new plan will be the same as the old plan only when the manhattan distance between the ghost cell and agent cell is more than 15. However, when it is less than 15, the agent checks for the neighbor (top, bottom, left, right) that is farthest from the nearest ghost and occupies that cell and this goes on in a loop until it finds a new path to the goal node.

Agent 3

Agent 3 is the improved version of agent 2 as it contains all the features that agent 2 has with the additional feature of simulating the future. In order to make this agent, we iterated agent 2's heuristic function 30 times on the neighboring and current cell of agent 3 whenever agent 3 has the information that there is a ghost in its path. If there is a case where the survivability rate is similar for two neighbors, we have coded the agent in such a way that it will choose the shortest path to the goal node by assigning weights (by multiplying with 0.7) to the neighbors whose manhattan distance is farthest from the goal node (top, left). Since we are assigning weights to the two neighbors, we are forcing the agent to go in a certain direction or stay in the same place. Though we call this agent the better agent it has its own drawbacks which I have mentioned in the following questions.

Q3 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?

Agent 3 checks the path to the goal node initially and checks whether there is any ghost in the path. If there is any ghost in the path, then it will check its neighboring cell with Agent 2's heuristic function and it will iterate it 30 times from each direction from the neighbor's cell with the condition that the manhattan distance between the ghost cell and the present agent cell is less than 15. Even though Agent 3 utilizes Agent 2's information, it has lower success rates than Agent 2 overall. Agent 2 simply replans its path based on the manhattan distance between the ghost and its current cell, hence its working mechanism is simple. Agent 2 helps agent 3 to decide the best neighbor that has a good probability of success rate, but, that being said, since it uses the same agent 2's heuristic function on every neighbor when it finds a ghost in its path to the goal node, there won't be any significant difference in success rates. For example, let us assume that agent 3 is in (2,1) coordinate, and is using agent 2's heuristic function on (1,1), (2,2), and (3,1) they will definitely have similar success rates as the number of ghosts in the maze remain same and the agent 2 functionality is also the same. In order to prevent the to-and-fro motion, or loop, in the same cells, we are assigning weights to the top

and left neighbors from agent 3 to prioritize going to the goal. Hence, it more often uses bottom and right neighbors, so its accuracy compromises. Moreover, let's say agent 2's heuristic function on a neighbor has the highest success rate after doing 30 iterations, but what will be the probability that its success rate is gonna stay the same after increasing the number of iterations, this debate continues with an increase in a number of iterations. This is due to the factor that ghosts' movements are random as the number of ghosts increases, randomness increases.

Q4 Agent 3 requires multiple searches - you'll want to ensure that your searches are efficient as possible so they don't take much time. 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 way agent 3 works in our code is that whenever the agent decides that there is no successful path in its projected future, it will check its neighboring cells (top, left, bottom, right) with Agent 2's heuristic function and will iterate Agent 2 30 times in those neighboring cells in order to check the best optimal path and this will stay in a loop until it finds the path to the goal node. If agent 3 doesn't have a successful path in its projected future, then it is highly possible that it is unlikely going to be successful. It doesn't guarantee that success will be possible or impossible, because it entirely depends on the situation in which the ghost or agent is facing. The factors may include the coordinates in which ghosts and agents lie, and the number of ghosts.

Agent 4

We initially thought to code the agent in such a way that it also ignores the ghost's bubble (Outer square boundary from the ghost's current cell including 9 cells) cells, but because of the limited path to the goal node with all the ghost's bubbles blocking the path, we changed our decision. So, we have taken the neighbors of the ghost (top, left, bottom, right) instead of the entire bubble so the agent won't pass through the path of the neighbors of the ghost. In scenario 1 of this agent, the agent considers the ghost's location when it goes into the block, hence the agent will avoid the path of the neighbors of the ghost even when the ghosts are in the block, which is more advantageous than the previous agents as they don't contain the exact location of a ghost when the ghosts enter the block. In this particular agent, we tried a different approach to checking whether the agent works better when the weight of the heuristic is 1 or when it is increased. We found that when the heuristic is multiplied by 10, it less often avoids the ghosts that are not its path, and more often tries to reach the goal node efficiently.

Q5 What possible ways can you improve on the previous agents? Does each planned path really need to be the shortest possible path? How could you factor in distance to ghosts? What do they do well? What do they do poorly? Can you do it better? Challenge yourself!

Agent 4 is better than the other agents that we have done so far. We have taken the neighbors of the ghost (top, left, bottom, right) instead of the entire bubble so the agent won't pass through the path of the neighbors of the ghost. It is efficient since the scope of the agent being killed is low. Each path need not really be the shortest path for the agent because there is a possibility that the agent moves from the upper coordinates of the maze if the ghosts are blocking the shortest path to the goal node. Moreover, for agent 4, the agent only considers a new path if the manhattan distance from the current cell and ghost cell is less than or equal to 10, hence it has a better advantage than agent 2 since it only reroutes the path if the distance is less than or equal to 10 instead of 15 or less than 15. The disadvantage is that when this agent avoids the bubble of the ghost, we are not considering the possibility that one neighbor of the ghost is the shortest possible path and is not visited by the ghost.

Scenario 2

Scenario 2 focuses solely on whether the agent is considering the ghost's path when the ghost is inside the block. Agent 1 uses the A* algorithm and goes in the path which reaches the goal node irrespective of the ghosts it is facing along the path. Hence it is unaffected by Scenario 2. Agent 2 replans its path based on the minimum distance of the ghost, but when the ghost is inside a block, agent 2 will be unaware of the ghost, and hence even this agent is not affected by Scenario 2. Agent 3 uses its neighbor cells by iterating agent 2's heuristic function 30 times in order to determine the best possible path. However, even agent 3 is not aware of the whereabouts of the ghost when the ghost is in the block. Agent 4 replans whenever its path coincides with the bubble of the ghost living but it doesn't have the information about ghosts when the ghost is in the block. However, in order to increase the performance rate of agent 4, we have introduced agent 5 in which the agent only replans whenever its path coincides with only the previous location of the ghost entering the block as this agent knows the location of the ghost entering the block.

Agent 5

Agent 5 only replans when the manhattan distance between the agent cell, and the ghost cell is less than or equal to 10 (similar to agent 4). The major advantage of agent 5 with respect to agent 4's first scenario is that, while agent 4 will ignore the bubble of the ghost even when it is in the block, it will not consider the possibility that one neighbor of the ghost is the shortest possible path and is not visited by the ghost. However, in agent 5, it will remember the previous location of the ghost cell, when it is entering the block cell and only ignores that cell in its path to the goal node.

Agent 5's scenario :

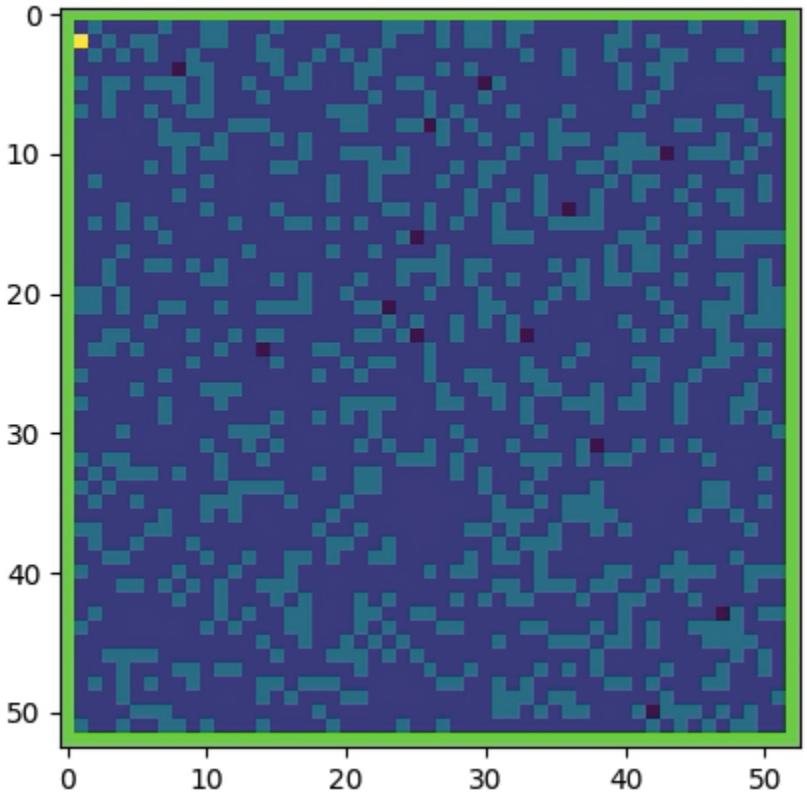

Figure - 3: GIF of Agent 5 Simulation

We can see the ghosts disappearing when they are inside a blocked cell (wall)

Issues faced

The major issue we encountered in agent 5 is that initially when we considered the previous location of the ghost entering the block we thought to consider the entire bubble of the ghost's previous location in which the agent ignores the path to the goal node. However, its performance rates are not as anticipated, hence we have coded the agent in such a way that it will replan only when the path to the goal node contains the ghost's previous location and not the bubble (top, bottom, left, right) before entering into the block.

Observations

Agent 1

Figure - 4: Agent-1 analysis over thousand simulations

It is abundantly evident from the graph above that agent's survivability declines as the number of ghosts increases.

Agent 2

Figure - 5: Agent-2 analysis over thousand simulations

The win-to-loss ratio(survivability) of agent 2 is high as long as the number of ghosts is approximately 28. However, the agent dies with 0 probability when the number of ghosts is more than 100.

Agent 3

Figure - 6: Agent-3 analysis of its simulations

Agent 3 has the least success rate as it is evident from the graph above. The survivability linearly decreases till the number of ghosts becomes 50.

Agent 4

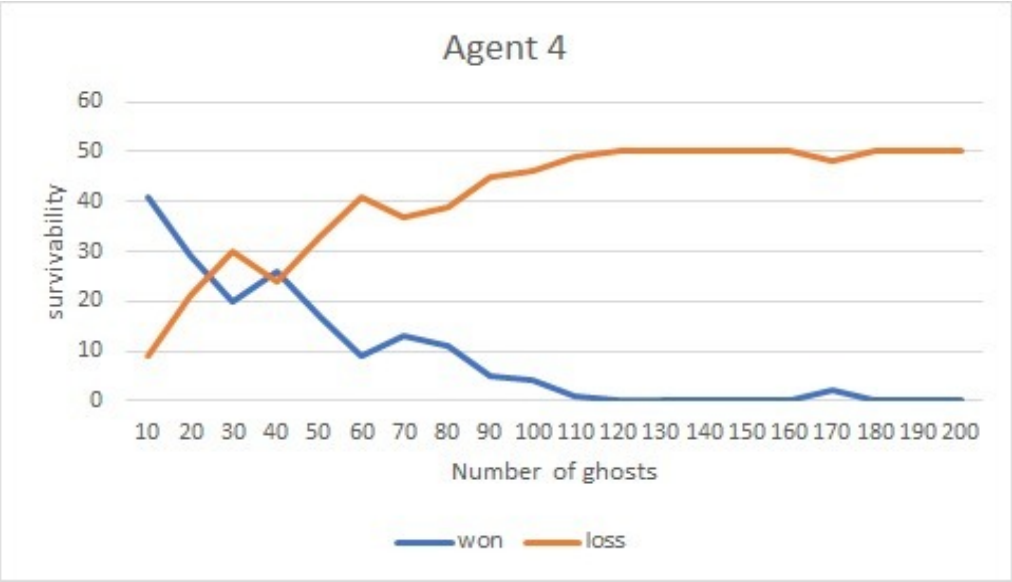

Figure - 7: Agent-4 analysis over thousand simulations

Agent 4's survivability is the highest among all the other agents that we have faced so far. There is still a possibility of the agent being alive even after the number of ghosts is 110.

Overall analysis of the 4 agents

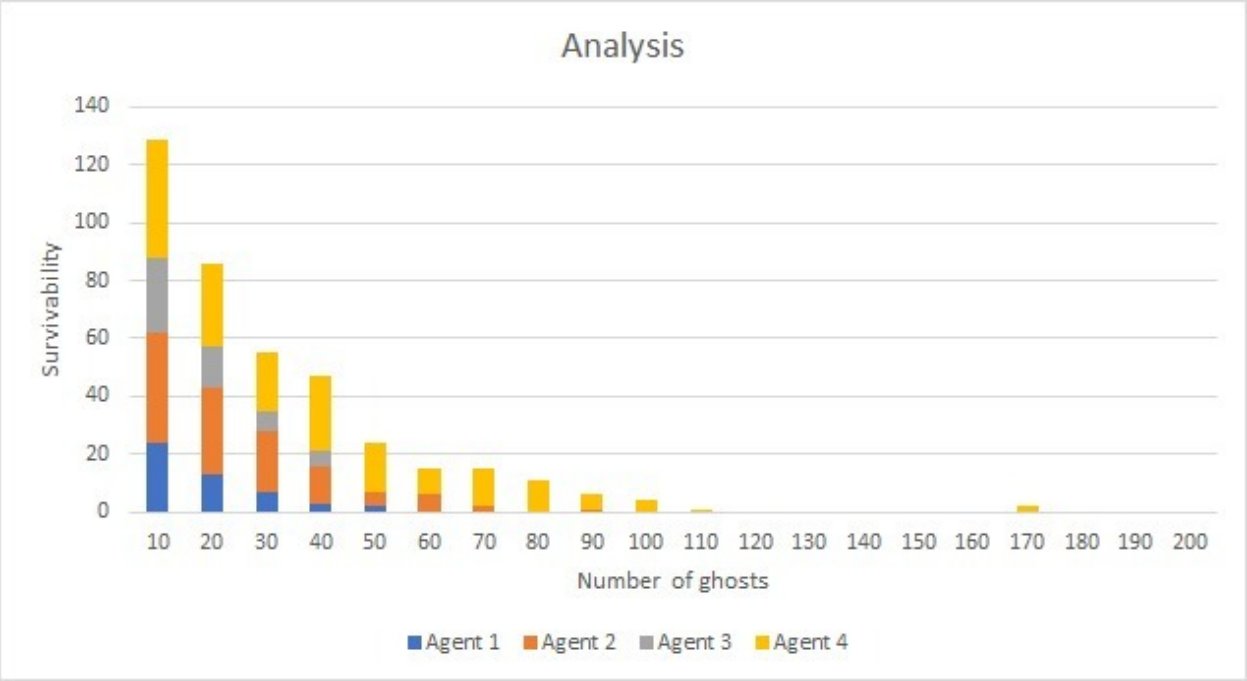

Figure - 7: Analysis ALL AGENTS over thousand simulations

Analyzing all 4 agents from the bar graph above, it is explicitly clear that agent 4 is the best agent especially when it comes to survivability. Agent 3 is best compared to the other agents as long as the number of ghosts is below 40. Agent 4, agent 2, and agent 1 surpass agent 3 when the number of ghosts is increased from 40 to 200. It is obvious that agent 2 is always better than agent 1 as it keeps in mind the whereabouts of the ghost.

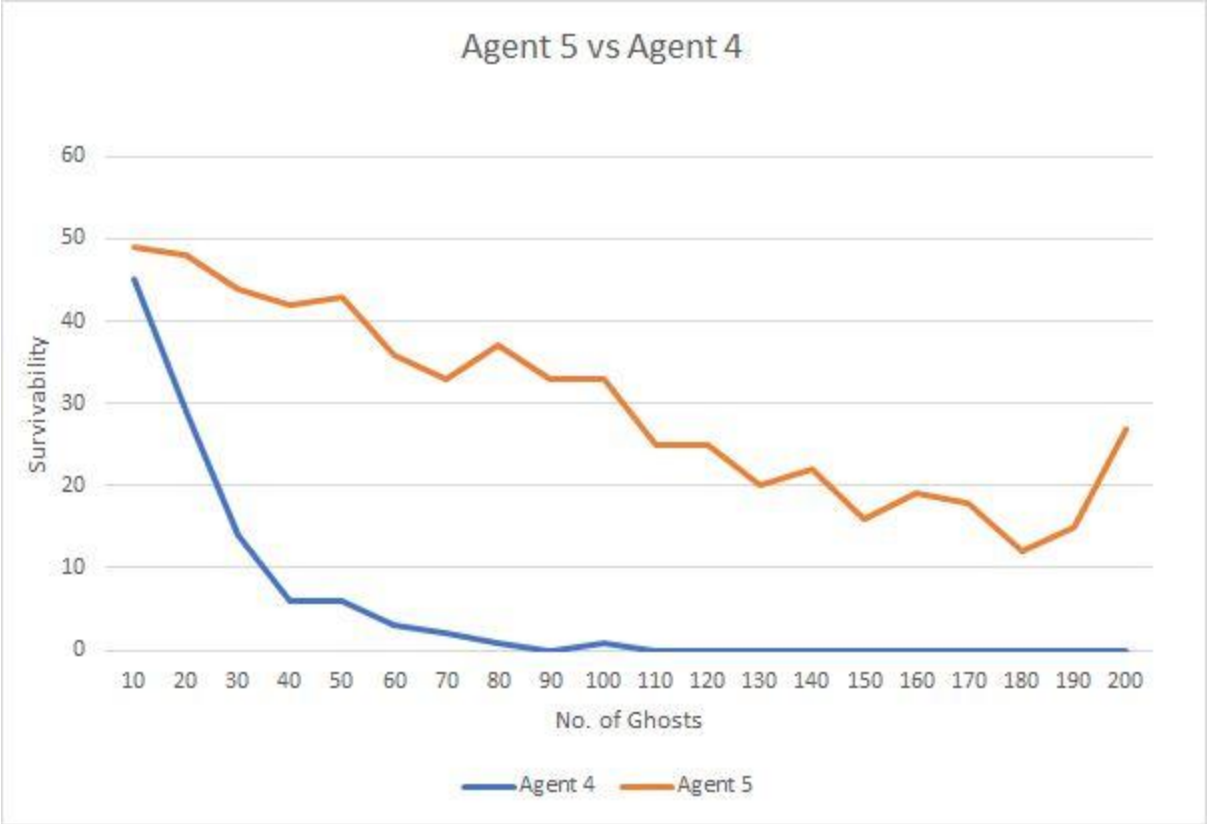

Figure - 8: Agent-4 and Agent-5 analysis over thousand simulations in scenario 2

We can see that when Agent 4 is in the second scenario it fails to achieve the efficiency that it had in original scenario. This is due to the fact that Agents are unable to see ghosts when they are inside the wall. Agent 4 fails here as it is unable to avoid ghosts that are inside the wall and can't create bubble around it which lowers its efficiency of avoiding path with ghosts. Agent 5 overcomes this dilemma by remembering the last known position of the ghost before it enters the blocked cell (wall) and only avoiding that cell when it is in its path. Agent 5 performs better as it does not considers bubble for the ghost that entered the block cell and only considers the previous cell giving it more room to find a path to the goal node.