CS 520: Exploring Search

Project 1: Ghosts in Maze

Prepared by:

dm1639: Darshee Machhar
Rutgers University

pn266: Pankti Nanavati
Rutgers University

14th October, 2022

Contents

1 Introduction

1.1 Problem Statement

A maze is an n*n grid of cells that are either blocked or unblocked. There are different types of agents that are attempting to traverse the maze by starting at cell (0,0) and finishing the maze at cell (n, n); each deploying their own strategies. Apart from navigating the maze, the agents also need to look out for ghosts that are spawned randomly in the maze. Ghosts are free entities that roam the maze and they can pass through walls, unlike our agents. If an agent and a ghost find themselves in the same cell, the ghost will kill the agent. The agents can move in the four main compass directions (east, west, north, south) to any adjacent unblocked cell. The agents have complete visibility to which cells are blocked and which aren't and where the ghosts currently reside. However, it is not aware of the ghosts' future positions and must gain that knowledge or intuition using some heuristic. By exploring the maze and collecting information on ghost positions, and integrating that into the whole context of traversal, the agents must reach the target as efficiently as possible, and more importantly, they shouldn't die whilst doing so.

1.2 Problem Environment

The problem environment (the maze) depends on the state of each cell in the Maze and the ghosts that inhabit it. Each cell is either blocked or unblocked depending on the given value of probability density for that maze, which in this case was 0.28 probability for making a blocked cell, and 0.72 probability for making the cell unblocked.

1.2.1 Maze Validity

A maze is valid if there exists a path from the start node till the end node. The maze that we generate using the probability density might be formed in such a way that the end node is walled off from the start node. Hence, before initiating planning and path traversal, we need to make sure there exists atleast one path from start to end in the maze.

1.2.2 State of Cells

- Unblocked: Cell which is "open" i.e. where the agent can move.
- Blocked: Cells that are "walled off" i.e. where the agent cannot move.
- Safe: Cells where there are no ghosts.
- Unsafe: Cells where one/more ghosts are currently present.

1.2.3 Ghost Movement

Ghosts move through the maze randomly taking at most one step at every timestep. There is an equal probability to ghosts going in any one of the cardinal directions (up, down, left, right) if the neighbouring cells are all unblocked. If the ghost is up against a wall there is a 50-50% chance that the ghost stays in its current cell or moves into the wall/blocked cell. If the is ghost already within a wall, the same probability applied to whether it moves out or stays put. Ghosts move after every time an agent makes a move.

1.2.4 Maze Properties

The Maze is generated on the basis of the following parameters:

- Dimension: The dimension of the maze is N*N. (For our project N=51)
- Density: Determines the probability of a cell being blocked. (For our project, p=0.28)
- Number of Ghosts: The number of ghosts roaming the maze can be increased or decreased to measure how difficult it gets to traverse the maze as the number of ghosts vary.

1.2.5 Terminologies

- Path- The route that the agent follows to reach from the start node to goal node.
- Temporary Path- Path that agent plans before actually implementing it, not considering the changing environment.
- Heuristic- The estimated cost to move from the current node to the goal node.
- Ghost future states/probabilities- The probability of the ghost occurring in the adjacent nodes of the current ghost position.
- Planning- Finding the path from the start node to the goal node considering the current states and the future movements of ghosts.
- Re-planning- Finding the new path from the current node to the goal node depending on the changing environment.
- Visibility- The sight that the agent has of the current states(blocked/unblocked nodes, ghost positions) in the maze.
- Survivability- The number of times the agent is successfully able to reach the goal node without being killed by ghost to the total number of times the agent is simulated.

2 Algorithm and Implementation

Throughout the problem we used three search algorithms according to the use case we had to fulfill.

1. DFS:
   Question: What algorithm is most useful for checking for the existence of these paths? Why?
   In order to check the maze validity we have to make sure there exists at least one path from start to end. Since we don't care about that path being the most efficient or the shortest path, we just want to make sure that one exists, we have used the DFS algorithm. DFS allows us to quickly explore and validate the maze.
2. BFS:
   Since Agent One plans the shortest path through the maze and executes it, we used the BFS algorithm because it guarantees to return the shortest path.
3. A Star:
   For Agent Four, we want to increase its survivability and hence increase its ability to dodge ghosts. Hence for this informed search, we took the probabilities of the presence of ghosts along with the distance from the goal node as the heuristic and applied the A star algorithm for an optimal path i.e. a path that is successfully traversed whilst also avoiding ghosts as much as possible.

2.1 DFS Algorithm

Depth First Search algorithm is a traversal algorithm that searches along each branch as far as possible before backtracking. It uses a stack data structure to keep track of the nodes that are discovered and uses this stack to backtrack when needed.

Pseudocode:

1. Initialise S=stack
2. Push the initial node to S
3. Repeat steps 4-7 till the S is not empty
4. v=Pop node from S
5. Mark v as visited
6. If v is the goal node, return True
7. For each adjacent node(i) of v
   If i not visited, then push i to the stack and mark i as visited
8. Return False

2.2 BFS Algorithm

Breadth First Search algorithm is a traversal algorithm that explores the nodes of the current level first before moving to the next level. It uses a queue data structure to keep track of the nodes that were encountered but are not yet explored.

Pseudocode:

1. Intialise Q=queue
2. Push the initial node to Q
3. Mark the initial node as visited
4. Repeat steps 5-7 till the Q is not empty
5. v=Dequeue node from S
6. If v is the goal node, return True
7. For each adjacent node(i) of v
   If i not visited, then push i to the Q and mark i as visited
8. Return False

2.3 A* Algorithm

A* algorithm is a best first search algorithm. It is different from other traversal algorithms since, it takes into consideration the estimated cost f(n) of each node. f(n)= g(n)+h(n), where g(n) is the cost so far to reach n and h(n) is the estimated cost to reach the goal node from n.

Pseudocode:

1. Initialise open list(O) and closed list(L)
2. Add start node to the O
3. Repeat Step 4-8 until O is empty
4. Initialise the currentNode with the node with least f(n)
5. Remove currentNode from O
6. Add currentNode to C
7. If currentNode equals the goal node, return True
8. For each adjacent node of the currentNode
   (a) If adjacent node is not already in C, then find f(n)

(b) If adjacent node is already in O and adjacent node's g(n) ¿ g(n) in O, then continue

(c) Add adjacent node to O

3 Agent One

Agent One only plans once and then executes it without paying heed to ghosts. Agent one always seeks out the shortest path through the maze. Hence our strategy for Agent one was BFS search as described in section 2.0.2

3.1 Path Planning

Agent one plans the shortest path through the maze and since we ensure that our mazes are valid i.e. solvable, agent one always does find the shortest path to take. Once it has the path, agent one starts executing it. The path agent one takes has taken into account all the blocked and unblocked cells but not the safe/unsafe ones.

3.2 Survivability

Since Agent doesn't factor ghosts into its planning, the agent does come face to face with ghosts a lot and thus dies in the process before ever reaching the goal node. As a result, the survivability of agent one is pretty low. It makes no effort to modify its path when a ghost comes in its way.
As it can be seen from figure 1, agent one has good survivability with less number of ghosts but as the number of ghosts increases the survivability steeply drops closer to 0 and then 0. This is because it does not take the changing environment into consideration.

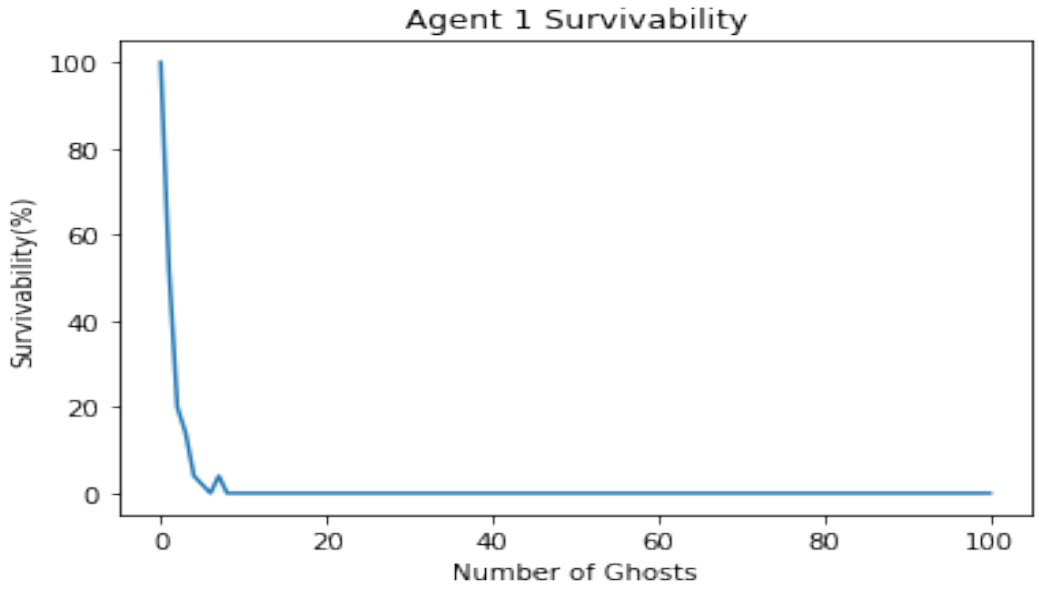

Figure 1: Agent One Simulations for 100 ghosts(50 simulations for each ghost)

3.3 Performance

Since the agent barely finishes the maze at all, because it dies so much, the performance of Agent one is generally poor. But if we were to talk about the path it takes for the cases it does complete its maze, that is pretty efficient and agent one is therefore the quickest maze runner of all the agents (for when it does survive).

4 Agent Two

Agent 2 plans a path but it also always re-plans at every timestep. Agent 2 is cognizant of and pays attention to the ghosts in the maze. Hence Agent 2 recalculates a new path to the goal based on the current positions of the ghosts and executes the next step accordingly.

4.1 Path Planning and Re-planning

We have modelled our Agent two around agent one but with some tweaks. Agent two calls agent one every time it needs to plan a new path i.e. it always plans the shortest path to the end goal. As you can recall Agent One doesn't consider ghosts while planning its path, but since Agent two does, we have made a little improvement in agent one. If Agent two is calling agent one to re-plan the path, agent one will consider the ghosts and not include any unsafe cells in its planned path.

4.2 Survivability

Because of our change in agent one, every path that agent two plans/re-plans is devoid of any "ghost cells" at that given timestep, but of course, since Agent two doesn't look into the future, or rather it makes no predictions of the ghosts' movements, there is still a possibility that agent two will encounter a ghost while traversing the newest re-planned path.
Therefore, though not perfect, agent two's survivability is higher than that of agent one's.
As it can be seen from Figure 2 that survivability is pretty better than agent one but as the number of ghosts keep increasing there is a gradual decrease and eventually the survivability becomes 0.

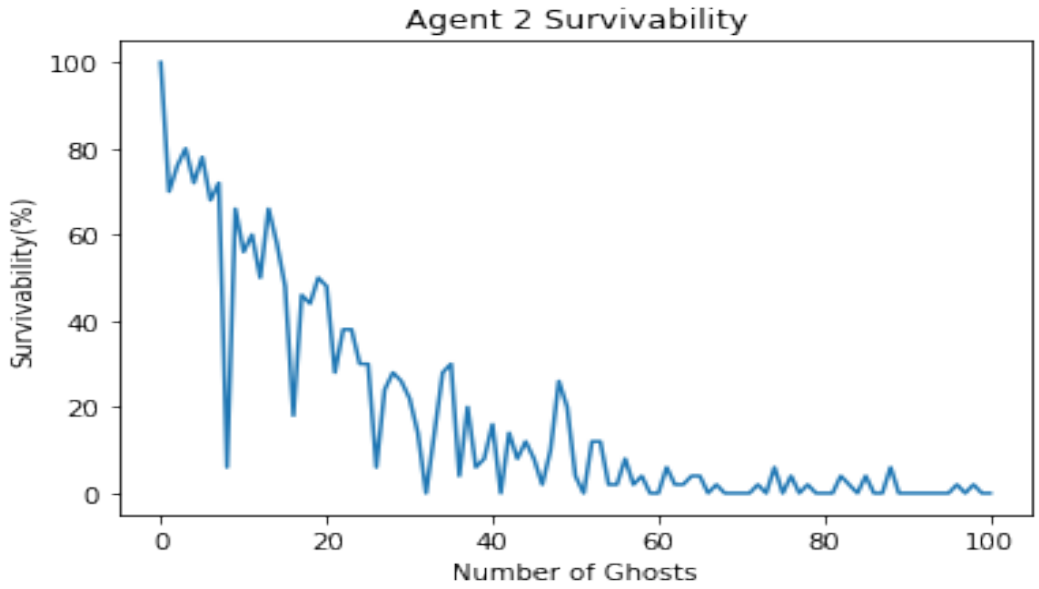

Figure 2: Agent Two Simulations for 100 ghosts(50 simulations for each ghost)

4.3 Performance

Agent two shows a lower rate of dying and its paths are also efficient since we continue to use BFS under the hood. However, performance only falters when it comes to speed. Because of re-planning at every time step, we are utilizing agent one every time agent two moves, so therefore the time agent two takes is several times that of Agent one's.

4.4 Improvement

Question: 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?

As mentioned above, the speed that agent two delivers leaves much to be desired. I believe, agent two doesn't need to re-plan all the time. We need to re-plan only when the previous plan is not going to work out. In other words, if the previously planned path is going to encounter a ghost, we can re-plan it in order to dodge the danger.

Version 2: So we designed version 2 for Agent two, where we were still using agent one but this time we only re-planned if upon traversing the maze, agent

one encounters a ghost i.e. agent one dies. When we measured its survivability and efficiency against version 1, there were no discernable differences, but the speed had improved by a bit. But then in class, Dr. Cowan presented us with a question, if there is a ghost 10 steps away, does it make sense for us to run away from it then? The ghosts can't move at super speed anyways, like agents they move through the maze, one cell at a time. So after further analysis, we designed version 3 for Agent two.

Version 3: We were re-planning when we encountered danger. So we thought of shifting the way we think of danger. We now only look at immediate danger. If the very next cell that we are about to take might have a ghost there, we re-plan our path. This drastically improved time for us.

For comparison, here is the table showing the frequency of replanning in a 51 x 51 maze.

Version	**Number of Re-plans**
1.	More than 100 times
2.	Between 80 to 90 times
3.	Below 20 times

Table 1: Agent 2 versions vs number of re-plans

We were going to use agent two to run simulations for agent three, hence time efficiency was paramount, hence the final version of Agent Two we went ahead with was the version 3.

5 Agent Three

Unlike Agent two Agent three forecasts. At every timestep, Agent three considers each possible move it might take (including staying in place), and 'simulates' the future based on the rules of Agent 2 past that point. For each possible move, this future is simulated some number of times, and then Agent 3 chooses among the moves with the greatest success rates in these simulations.

5.1 Path Planning and Simulations

As agent three traverses the maze, we call agent two recursively for every possible next node. We run the simulation 5 times in each direction i.e. we run agent two with every possible next node/unblocked neighbour of the current cell agent three resides in as the start node and the goal node as the end node. If agent two can traverse the maze successfully we increase the survivability score for that particular node. Agent three then goes ahead with the node with the highest survivability score.

5.2 Predicting the Future

Apart from running simulations, we also base the selection of nodes on the positions of ghosts, one timestep into the future. After we find the most optimal node based on its survivability score, we crosscheck for further safety that there will be no ghost in that position in the future (in the very next timestep).
We find these future ghost states using the probabilities of ghost movement given in the question, to get the next 'y' cells for the each of 'x' ghosts present in the maze. We store this collection of 'y' cells for each ghost and make sure the next node is not among these. If it is, we go to the next optimal node from among the neighbors of Agent 3. If we run out of neighbors (this includes the option to backtrack), agent three stays put, stays in the same place and waits for the ghosts to change their position.

High-Level Psuedo Code:

1. Get Current Node N
2. Find All valid neighbors: Nx = N1, N2, N3, N4
3. For each neighbor:
   (a) Run agent 2 with start node = Nx and end node = goal node
   (b) Return a True/False value for the path traversal status
   (c) If status = True: Increase survivability score of the node Nx
   (d) Repeat steps (a to d) 4 more times.
4. Sort Nx in descending order of survivability

5. Select N1 (Nx with the highest survivability score)
6. Check if N1 is in the future ghost states
   (a) If yes: Choose Nx+1 and repeat step 6
   (b) If no: Append to the path and continue to step 1
7. If no "next node" is safe, the agent chooses to stay where it is

5.3 Survivability

Agent three has very high survivability. Since it is basically agent two plus the ability to look into the future, we are able to dodge the ghosts, run away from immediately nearby ghosts and also stay away from cells where a ghost might pop up in the near future. All of this put together makes our agent three pretty immune.
As seen from figure 3, the agent three's survivability has improved compared to agent two. A gradual decrease in the survivability is seen as the number of ghosts keep increasing. It can be seen that 0 survivability has been reduced.

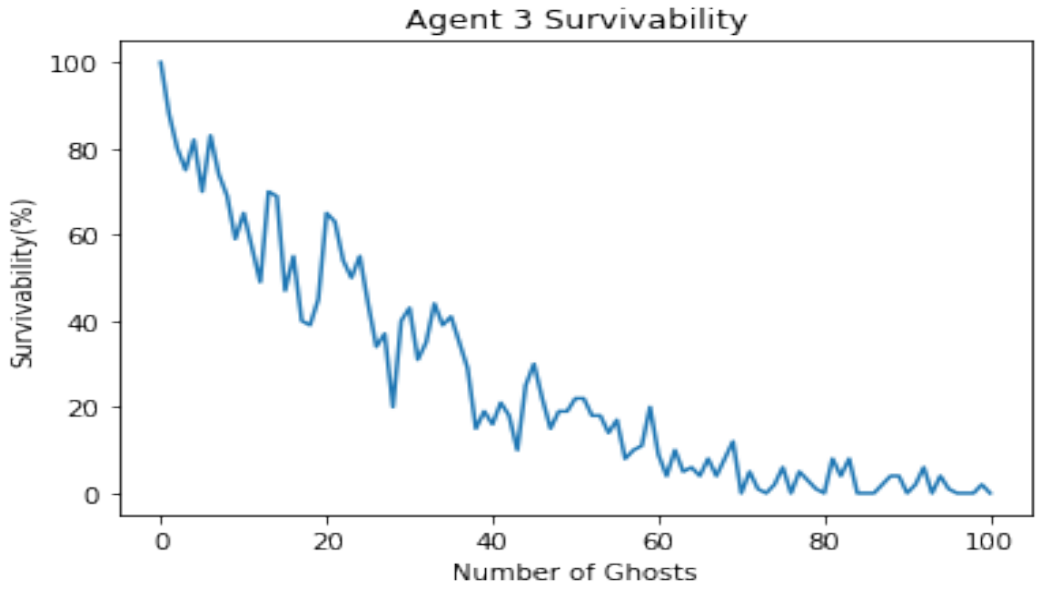

Figure 3: Agent Three Simulations for 100 ghosts(50 simulations for each ghost)

5.4 Performance

Agent three is a slow agent and does take a lot of time to run. Its survivability is better than that of agent two's but its speed is much worse.

5.5 Challenges we faced

(a) The Time Issue-
Agent three is not a fast agent, and understandably so, it is running Agent two repeatedly before each step it takes. So it was getting very tough to collect data for the agent for a maze as big as 51 x 51 for varying numbers of ghosts multiple times. That is when we made improvements for Agent two and made sure Agent two doesn't re-plan at every step. As mentioned in section 4.4 we reduced the running time of agent two drastically and that helped us a lot when we ran agent three as well.

(b) The Wiggle Effect-
Question: Agent 3 requires multiple searches - you'll want to ensure that your searches are efficient as possible so they don't take much time.
It would happen very often that the agent would go from Cell A to Cell B and then at Cell B, it would realize that the most optimal next node is Cell A itself. So the agent would be stuck in a sort of loop of going from A to B to A and back to B and so on until the ghosts around them would get close and either force the agent to go to another node or the ghosts would kill the agent as it kept wiggling between two cells without making any progress.
In order to fix the wiggling issue, we wrote a piece of code to first detect if the Agent is wiggling between two cells, if yes, we would hold a tie-breaker between the two. The tie-breaker was decided by two things:

- Of course, it should be safe and away from any immediate danger from ghosts
- It should allow the agent to make progress, hence we calculated the Euclidean distance to the goal node. Whichever cell was closer to the goal, won the tie-breaker.

Further, we made sure the tie-breaker node doesn't go back to the "wiggle node" again.
Solving the wiggle effect also helped us drastically reduce the time agent three was taking to get executed.

(c) Resetting the Ghost Position
We have made ghost positions a global variable.
When we called agent two from within agent three to run simulations, the code for agent two would move the ghosts as was necessary for the simulations (we had to check if agent two survives or not) but when the

command came back to agent three, we weren't resetting the ghosts position and agent three would take a step and move the ghosts again.
When we saw the output, the ghosts were moving randomly it seemed because we couldn't see the movement happening under the hood (done by agent two).
It took some debugging to realize we must reset the ghosts' position back to their original after running the simulations.

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?
As mentioned in section 5.2, if agent three finds no successful path in the predicted future, our agent three stays where it is. This could mean waiting until the ghosts move and clear the path or it could also mean the ghost will get cornered and killed. So this doesn't guarantee that success is impossible. It will depend on the movement of ghosts.

6 Agent Four

We had to design a free agent, that has better survivability than all the preceding agents. Since we didn't have to ensure the shortest path in order to increase survivability we thought that we don't need to bootstrap any of the previous agents and we can start afresh. We thought if we planned a path in such a way that it stays away from ghosts and their future states all the time whilst also slowly moving toward the goal node, we would be able to ensure higher survivability. Therefore we used the A star algorithm and designed a heuristic that takes into account, not just the distances but also the current and future presence of ghosts.

6.1 Predicting the Future

In Agent Three we were predicting the positions of ghosts after one time step. Thus:

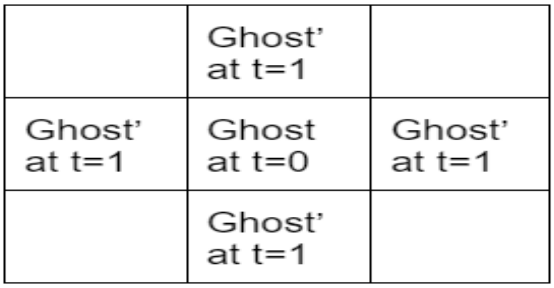

Figure 4: Future Ghost Positions after one timestep

For agent 4 we thought that we should look more into the future. So we found the ghost future states up to 4 timesteps into the future and also attached probabilities to these future possible states, like this (the figure shows the future probabilities up to two timesteps):

		7% chance Ghost" at t = 2		
	14% chance Ghost" at t = 2	25% chance Ghost' at t = 1	14% chance Ghost" at t = 2	
7% chance Ghost" at t = 2	25% chance Ghost' at t = 1	Ghost at t=0	25% chance Ghost" at t = 1	7% chance Ghost" at t = 2
	14% chance Ghost" at t = 2	25% chance Ghost' at t = 1	14% chance Ghost" at t = 2	
		7% chance Ghost" at t = 2		

Figure 5: Future Ghost Positions after two timestep

We would then use these probabilities to inform our heuristic.

6.2 Heuristics and Path Planning

In A star algorithm, f(n) = h(n) + g(n)
Where h(n) is the heuristic (normally it is an estimate of the cost of getting to the goal node from the current node) and g(n) is the cost incurred so far.
In our case, this is how we estimated the heuristic: H = Manhattan Distance between the current node and goal node + some weight * (probability of a ghost being present at that cell)
We multiplied some weight to the probabilities to give more impetus to that aspect of the heuristic. As the agent went along traversing the maze, it would calculate the heuristic for every cell for each timestep and then choose the one with the lowest cost.

Question: How did your Agent 4 stack up against the others? Why did it succeed, or why did it fail?

6.3 Survivability

Agent four's survivability was higher than all of the other agents. This is because it would not just avoid nearby ghosts or run away from them, it would avoid going near ghosts altogether. Since we prioritized ghost probabilities over just the path cost, our agent intuitively focused more on surviving than quickly

reaching the goal.
As seen from Figure 6, the agent four's survivability has increased further. There is a less gradual decrease in the line with increased survivabilty for more ghosts.

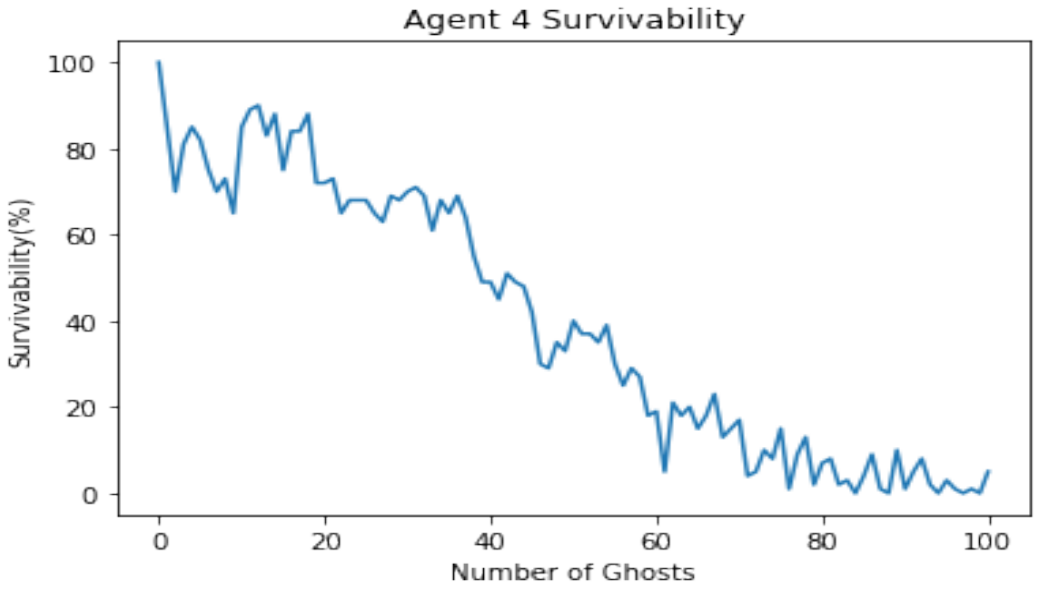

Figure 6: Agent Four Simulations for 100 ghosts(50 simulations for each ghost)

6.4 Performance

Agent four didn't meander as much. Its path was still pretty efficient and agent four was much faster than agent 3.

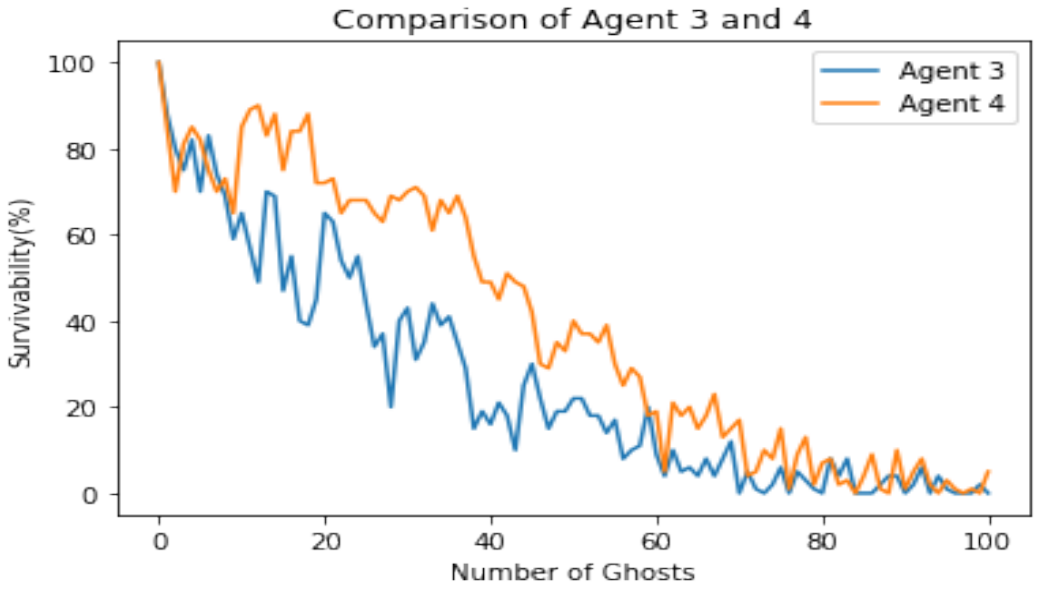

Figure 7: Comparing simulations for Agent 3 and Agent 4

7 Agent Five

Q. Redo the above, but assume 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? Build an Agent 5 better suited to this lower-information environment. What changes do you have to make in order to accomplish this?
So far all of our agents had complete visibility into the maze and all of the maze's components- status of cells and positions of ghosts. So a situation where the agent doesn't have visibility into the existence of ghosts when they are in walls poses a problem of whether or not to avoid a particular cell while traversing. The chances of "surprise attacks" i.e. the agent encountering unseen ghosts within walls that have just come out increase.

7.1 Path Planning

We have modeled our agent five on agent four. Therefore we will be following the A-star algorithm but with certain changes in the heuristics and visibility.

7.2 Modified Visibility for Ghosts in walls

Since agent five can not see ghosts when they are in walls, the future movements of those ghosts also become unknown to the agent. Therefore while calculating the future probabilities of the ghosts for the heuristic, we need to omit any and all ghosts currently in a wall i.e. occupying a blocked cell.

7.3 Modified Heuristics

The modified visibility handles only the representation of the lower information environment but it doesn't help us with how to navigate it. For our agent to successfully manage in such an environment, we added another aspect to our heuristic.
We negatively penalize all unblocked cells that are next to blocked cells. This ensures that the agent will intuitively choose to take a path that stays away from walls and thus reduces chances of any surprise attacks.
In this case, this is how we estimated the modified heuristic: H = Manhattan Distance between the current node and goal node + some weight * (future probability of unwalled ghosts being present at that cell) + penalty for cells in the vicinity of walls

7.4 Survivability and Performance

Agent four and five are similar when it comes to survivability and performance. Also since the walls in the maze are pretty sparse, there were not many opportunities for any surprise attacks by ghosts in the walls. Therefore there weren't

any discernible differences between the two agents.
As seen from figure 7, the agent 5 shows similar trend in line to the agent four.

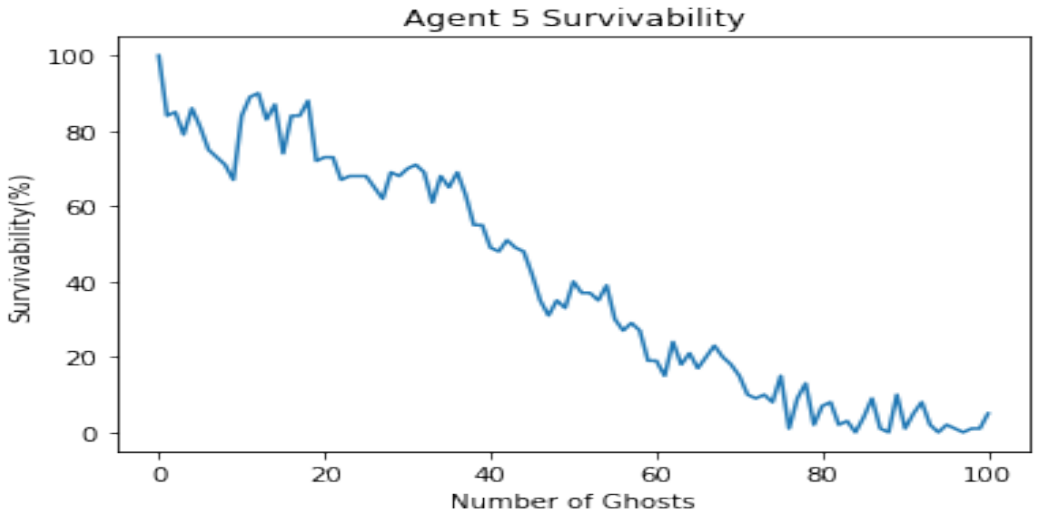

Figure 8: Agent Five Simulations for 100 ghosts(50 simulations for each ghost)

Figure 9: Comparing simulations for Agent 4 and Agent 5)

8 Comparison

Question: Graphs comparing the performance of each agent for different numbers of ghosts. When does survivability go to 0?–Note that to get a decent graph, you'll have to repeatedly run experiments for each agent, with different mazes and each number of ghosts, and average the results together. The more times you can do this, the more accurate your estimates of success rates will be.

All agents performed well when the ghosts were fewer. But as the ghosts increased: Agent 1 had the lowest survivability. Agent 3 had the highest survivability, but it was not an efficient agent, it took too long to compute the optimal path to take. Agent two wasn't as successful as agent three but it took drastically less time. So each agent has their pros and cons, no one agent is the best. But speaking with respect to survivability, agent three always trumped the other two.

Figure 10: Comparing simulations for Agent 1, Agent 2 and Agent 3