Ghosts in the Maze Lab Report

Tej Shah
Rutgers University
`tej.shah@rutgers.edu`

Srinandini Marpaka
Rutgers University
`sm2237@rutgers.edu`

Abstract

We build a `ghosts-in-the-maze` simulator and develop 5 different intelligent agents for High Information (all ghost locations are known) and partially observable Low Information environments (only visible ghost locations are known). Agent 1 plans an optimal path once, ignoring ghosts, and executes actions in its path until it reaches a terminal state. Agent 2 re-plans paths whenever a ghost is obstructing its current path to the end of the maze and it maximally moves away from the nearest ghost if no path to the end of the maze with the ghosts exists. Agent 3 forecasts the future survivability by running Agent 2 for each possible action it can take, and then processes that survivability with a curiosity heuristic and distance to end goal heuristic; then, Agent 3 uses these processed results for greedy hill climbing local search. Agent 4 forecasts the future by computing the expected utility of action using a game tree (Player 1: Agent, Player 2: Ensemble of Ghosts) which tries to minimize the distance to the k nearest ghosts; those statistics are then processed to curiosity and distance to end goal heuristics; then, Agent 4 uses the processed results for greedy hill climbing local search. We build Agent 5 to work for a lower information environment by augmenting Agent 4 with a "memory" of the most recent ghosts that have been seen. We run our agents on High Information (HI) and Low Information (LI) environments for $5 \times 5, 10 \times 10, 15 \times 15, 20 \times 20, 30 \times 30$, and 51×51 grids. Our experimental results indicate that the survival of the agents rank generally as follows for HI environments: $A1 < A2 < A3 \leq A4$. Our experimental results indicate that the survival of the agents rank generally as follows for LI environments: $A1 < A2 < A3 \leq A4 \leq A5$. We observe similar performance curves in different-sized grids, suggesting that the agents we developed are generalizable and agnostic to grid size in terms of success rate.

1 Introduction

Consider a $n \times n = 51 \times 51$ maze-like square grid where some cells are open (unblocked) and some are obstructed (blocked). An agent in the environment moves among/enters unblocked cells but cannot enter blocked cells. Cells are blocked with some probability $p_c = 0.28$. The goal of the agent is to start from the top left corner of the maze and reach the bottom right corner of the maze by moving around the unblocked squares in the maze without dying. Unfortunately for the agents, there are n_g ghosts that are running around in the maze initialized to random locations on the board. The ghost moves according to the following rules: at every time step, a ghost pick one of its neighbors (up/down/left/right); if the picked neighbor is unblocked, the ghost moves to that cell; if the picked neighbor is blocked, the ghost either stays in place with $p_g = 0.5$ or moves into a blocked cell with $p_b = 0.5$. These rules apply even if a ghost is currently in a blocked cell. Each time the agent moves, the ghost moves. If the agent enters a cell with a ghost, then it dies and does not reach the goal.

We implement 4 Agents $\{A1, A2, A3, A4\}$ with perfect information about the state of the board: the maze, blocked and unblocked cells in the maze, and the location of all ghosts (even in walls). We then test these 4 Agents $\{A1, A2, A3, A4\}$ in a high and lower information environment, where these agents only have access to visible ghosts at each time steps (in other words, these agents no longer have visibility to ghosts inside blocked walls). We then build another agent $A5$ better suited for this lower information environment. We run our agents on High Information (HI) and Low Information (LI) environments for $5 \times 5, 10 \times 10, 15 \times 15, 20 \times 20, 30 \times 30$,and 51×51 grids for experiments.

2 Agents

The terminal state for all agents is either a success or a failure: (1) an agent runs into a cell occupying a ghost (failure state) or (2) an agent reaches the end of the maze without dying (success state).

2.1 Agent 1

Agent 1, $A1$, plans an optimal path from the start of the maze to the end of the maze, ignoring ghosts, with Breadth-First Search (BFS) when it is first initialized. At every time-step, $A1$ runs the next action in the initially planned path until it runs into a ghost or reaches the end of the maze. We choose to use BFS, at a higher space complexity cost, instead of Depth-First Search (DFS) since the path we find to the end of the maze is guaranteed to be the shortest path, whereas that's not the case with DFS.

2.2 Agent 2

Agent 2, $A2$, uses information about the locations of the ghosts to inform its planning strategy. $A2$ treats ghosts as blocked cells overlayed on the current maze as an `effective-maze`. Naively, $A2$ plans a path to the end of the `effective-maze` at each time-step, always executing the next action in the new plan. Efficiently, $A2$ instead chooses to re-plan a path that reaches the end of the `effective-maze` only when the next action in its previously planned path at time-step $t + 1$ will run into a ghost at time-step t. If no path to the goal state from the agent's current position in `effective-maze` exists after re-planning, Agent 2 chooses an action in it's action space that maximally moves away from the nearest ghost (as measured by manhattan distance). In implementation, after discussion with TA, we decided to run BFS instead of DFS for path-planning, since the difference in wall-clock time is marginal, even at a higher space complexity cost.

2.3 Agent 3

Agent 3, $A3$, "predicts the future" by bootstrapping $A2$. At every time step, Agent 3 considers all possible neighbors that it can select. For each of those possible actions x that Agent 3 takes, Agent 3 simulates and stores the survivability $\widehat{s(x)}$ of $A2$ k times after after l timesteps of $A2$ in the environment. After empirical analysis and observation, to trade-off between intelligence and efficiency, we set $k = 2$ and $l = 3$. After $A3$ stores the percentage survival for k simulations for each action in its action space, we can naively choose to greedily select the action that maximizes the survival rate. Through empirical result and observation, we realized that this greedy local hill climbing approach gets stuck in local minima, "wiggling around" states that it has seen before at the origin, never reaching the goal. Hence, we input survival rates to a curiosity-heuristic function:

$$c(\widehat{s(x)}) = \begin{cases} \widehat{s(x)} & \text{if } x \notin \texttt{visited \{coordinate : frequency\}} \\ \widehat{s(x)} \cdot a^{\texttt{visited[x]}} & \text{if } x \in \texttt{visited \{coordinate : frequency\}} \end{cases}$$

Through empirical analysis and observation, we found $a = 0.6$ to work well for the parameter a in the curiosity function $c(\widehat{s(x)})$. This function rewards the agent for exploring novel states (every time we re-visit a state, $c(\widehat{s(x)})$ exponentially decreases the likelihood of visiting that state). Empirically we noted that curiosity alone is not enough for the agent to reach the goal state. While an agent might be exploring the maze, it does not necessarily reach the goal state, which is the end of the maze. Hence, we process $c(\widehat{s(x)})$ into a distance heuristic d that rewards the agent for choosing actions closer to the end goal of the maze based on manhattan distance. We define the function d as follows:

$$d(c(\widehat{s(x)})) = \frac{c(\widehat{s(x)}) + 1}{(\texttt{manhattandistance}(x, \text{goal}) + 1)^2}$$

We use Laplacian Smoothing (add $+1$ to numerator/denominator) to avoid using zero probability for an action. The action Agent 3 takes is $\arg\max_x d(c(s(x))$. Adding this distance heuristic in addition to the curiosity heuristic enables Agent 3 to successfully reach the end of the maze more often.

2.4 Agent 4

Agent 4 also plans in the future. Instead of planning the future based on the survivability of Agent 2, we make Agent 4 plan the future based on an evaluation or utility heuristic function u that measures how close an action in the future will be to the k nearest ghosts. Through empirical analysis and observation, we found $k = 10$ to be a good value. The way that we plan for the future is using a variant of "minimax", where we assume that the agent is a rational player and the Ensemble of Ghosts are an adversary that move according to the rules of ghosts movement, instead of assuming ghosts are rational as is the case of minimax. The agent wants to maximize the negative manhattan distance to the k nearest ghosts every time the ghost moves.

After running the game tree for d steps, we propagate the utility up the tree that maximizes the negative manhattan distance to the k nearest ghosts every time the agents moves to a location x. We found $d = 7$ to be a good value through observation and empirical analysis, trading off optimality for efficiency. As in the case of Agent 3, we process the utility values for each possible action through a curiosity function c and distance heuristic function d. The heuristics of distance to ghost, curiosity, and distance to goal enables Agent 4 to reach strong performance in our experiments.

The functions and the parameters we used based on empirical observations are shown below. Based on our observations and empirical results, we found $z = 5, a = 0.6, b = 15$ to be good values.

$$u(x) = \frac{1}{(\texttt{utility of } x \texttt{ per game tree} + 1)^z}$$

$$c(u(x)) = \begin{cases} u(x) & \text{if } x \notin \texttt{visited \{coordinate : frequency\}} \\ u(x) \cdot a^{\texttt{visited[x]}} & \text{if } x \in \texttt{visited \{coordinate : frequency\}} \end{cases}$$

$$d(c(u(x))) = \frac{c(u(x)) + 1}{(\texttt{manhattandistance}(x, \text{goal}) + 1)} + \left| \frac{1}{u(x) + 1} \right| \cdot b$$

As is the case for $A4$, we use Laplacian Smoothing to ensure non-zero probability for u and d.

2.5 Agent 5

Agent 5 only has access to the visible ghosts at time-step t. We boot-strap Agent 4 running in a low information environment by augmenting it with memory, to consider all ghosts in memory as opposed to all the ghosts in the environment as it had known previously in the high information setting. At every time-step t, we store all visible ghosts and their location in memory with priority 1. At every time-step t, we increase the priority of all ghosts in memory that are currently not visible by 1. We prune any ghosts in memory that have priority n, so for example, in a $n \times n = 51 \times 51$ grid, any ghost with priority 51 would be removed from memory. At every time-step t, the agent considers all visible ghosts and all ghosts currently in memory to compute utility values in the game tree, before those evaluations are passed through the A4's curiosity heuristic c and distance to goal heuristic d.

3 Lab Report Questions

3.1 When generating mazes, you want to make sure that a path is "not too blocked'. What algorithm is most useful for checking that a path exists from start to end of a maze? Why?

DFS from the start location of the maze to the end location of the maze is most efficient for verifying the existence of a path as opposed to BFS. We do not need to find an optimal path from the start to the end - we just need to verify that a path exists. In the worst case, BFS and DFS have the same time complexity, but DFS has better space complexity than BFS, which is why we choose to go with DFS for verifying existence of a path. A maze is valid if at has ≥ 1 path from start to end. In practice, we had initially implemented DFS with recursion. We ran into challenges with verifying the existence of mazes when we scaled it up to larger grid sizes, since in the worst case, we could be adding n^2 recursive calls to the call-stack but `python` only supports 999 items on the recursive call stack. Hence, we re-implemented DFS iteratively, which ended up working well.

3.2 Agent 2 requires multiple searches - how do you make the searches as efficient as possible? Do you always need to re-plan? When will the new plan be the same as the old plan?

We reiterate our methodology for $A2$ from section **2.2**. Agent 2 considers all the ghosts in the environment at each time-step to inform its strategy. Agent 2 does not need to always naively re-plan at each time-step t. Instead, Agent 2 only re-plans a path only when a planned action at time-step $t+1$ will run into a cell occupying a ghost at time-step t; since we use BFS for search, which guarantees an optimal path if it exists, until then, any new plan will effectively be the same as the old plan.

3.3 Agent 3 requires multiple searches - how do you make the searches as efficient as possible? 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?

We reiterate our methodology for $A3$ from section **2.3**. We run Monte Carlo simulations of $A2$ for each action x in $A3$'s action space. Through observation, we noted that knowing $A2$'s survivability in the near $l = 3$ timesteps is most relevant for $A3$ making a decision about an action since the near future is most relevant. This was a huge speed boost, because, previously, we were running $A2$ until it reached a goal state or ran into a ghost. We ran the simulations of $A2$ for each action x in $A3$'s action space $k = 2$ times, which was sufficient for planning for the near future. This was also a large speed boost since previously, we were running around $k = 10$ simulations; but, since the performance with more simulations was marginally different, we decided to stick with $k = 2$. Finding low values for k and l made Agent 3 efficient in search as Agent 3 runs through the maze.

If Agent 3 decides there is no successful path in its projected future, $A3$ selects an action x that is $\arg\max_x d(c(\widehat{s(x)}))$. In simple terms, if all actions have $\widehat{s(x)} = 0$, we greedily select the maximum value of the composition of a distance-to-heuristic function d and curiosity heuristic function c. In simple terms, we prioritize locations that are least frequently visited and closer to the goal.

Having a success rate $\widehat{s(x)} = 0, \forall x \in \texttt{AgentActionSpace}$, does not guarantee that success is impossible. All it means is that the average rate of success in k simulations is 0. As k increases, $\widehat{s(x)} \approx s(x)$. We have two arguments for why success is not necessarily impossible: (1) we use a low value of k for speed reasons, so $\widehat{s(x)}$ might not be the best approximation for the true survival rate $s(x)$; (2) while Monte Carlo simulations gives an expected value for some kind of input, it is not unreasonable for an outlier sample to have survived in a very large number of simulations. While it is not guaranteed that success is impossible, we are confident that success is unlikely.

4 Experiments

4.1 Overview

For both high-information and low-information environments, we analyzed the performance of the agents by running 30 simulations per agent per increment of ghosts. The results for the 5×5, 10×10, and 15×15 grids were collected by keeping the environment consistent between agents for each increment of ghosts. In contrast, the results for the 20×20, 30×30, and 51×51 grids were collected by running each agent in a different process, meaning that there is a low chance that the environments were the same. The agents' survival rates are plotted against the number of ghosts for both environments in Figures 1 - 12 for both the HI and LI environments on varying grid sizes. For each grid, we define the agent that is able to survive with the longest with the largest number of ghosts to have the best performance.

Note that we were not able to run Agent 3 on a 51×51 grid for either the high or low information experiments due to significant wall-clock time and computational complexity, further discussed in section **5.2**. For that reason, we will generalize the performance of $A3$ on the 51×51 grid based on its performance on the smaller grids. We also added a timeout for Agents 3, 4, and 5 in an effort to speed up data collection. Based on the amount of time it took for one simulation, we set the timeout per simulation to be 4 minutes for $A3$ and 3 minutes for $A4$ and $A5$ on the 20×20, 30×30 and

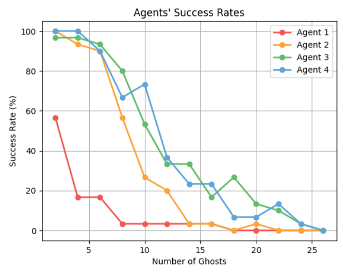

Figure 1: HI 5×5 Grid

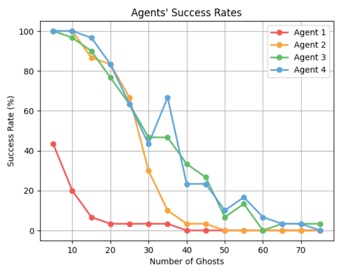

Figure 2: HI 10×10 Grid

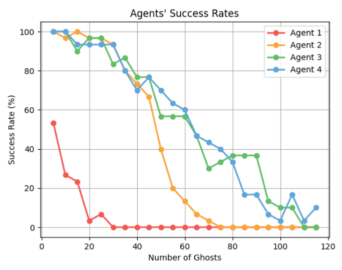

Figure 3: HI 15×15 Grid

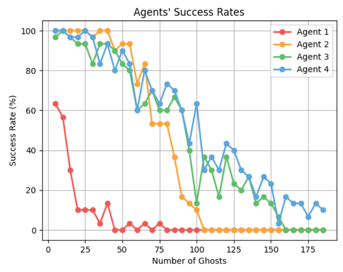

Figure 4: HI 20×20 Grid

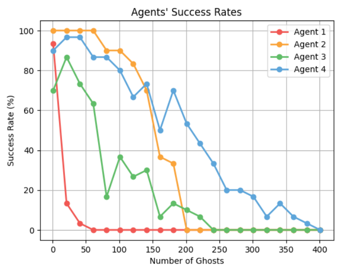

Figure 5: HI 30×30 Grid

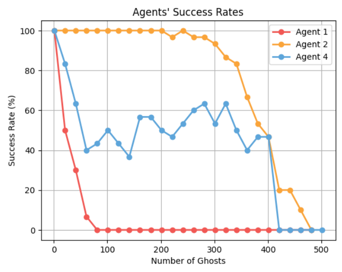

Figure 6: HI 51×51 Grid

51×51 grids. We treated a timeout as a loss for the purpose of determining our success rate, and in addition, separately recorded the number of terminations per number of ghosts. The termination rate is plotted against the number of ghosts for both environments in Figures 13 - 18. Note that once an agent reaches 0% survival rate, we plot the agent to have 0% survival rate for the rest of the visualization so that it's easy to compare performance between agents as ghosts increase in count.

4.2 High Information Experiments

Figures 1 - 6 illustrate the results of running Agents 1 - 4 in high-information environments.

- 5×5: $A3$ and $A4$ perform best, surviving up to 26 ghosts.
  - Rankings: $A1 < A2 < A3 \leq A4$
- 10×10: $A4$ performs best, surviving up to 73 ghosts.
  - Rankings: $A1 < A2 < A3 \leq A4$
- 15×15: $A4$ performs best, surviving up to 118 ghosts.
  - Rankings: $A1 < A2 < A3 \leq A4$
- 20×20: $A4$ performs best, surviving up to 186 ghosts.
  - Rankings: $A1 < A2 < A3 \leq A4$
- 30×30: $A4$ performs best, surviving up to 382 ghosts.
  - Rankings: $A1 < A2 < A3 < A4$
- 51×51: $A2$ performs best, surviving up to 469 ghosts.
  - Rankings: $A1 < A3 < A2$

The data helps us conclude that, on average, the performance of $A1 < A2 < A3 \leq A4$ in the high information environments. The data collected on the 51×51 grid is inconsistent with the data of the other grids since $A4$ times out in 30.15% of its simulations due to larger grid size whereas $A2$ does not time out at all. We note that $A4$ is significantly more computationally intensive than $A2$ because $A4$ has to plan for the future whereas $A2$ does not. Hence, because of the timeouts, we are not surprised that $A2$ outperforms $A4$ in the simulations we ran for the 51×51 grid.

$A2$'s better performance than $A4$ on the grid can be attributed to the difference in the agents' runtimes. $A2$ does not plan for the future. $A4$, on the other hand, forecasts 4 steps into the future. Between

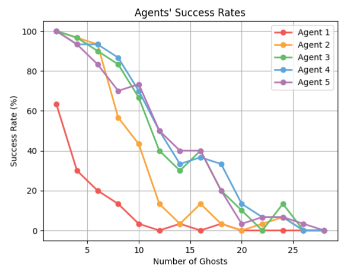

Figure 7: LI 5×5 Grid

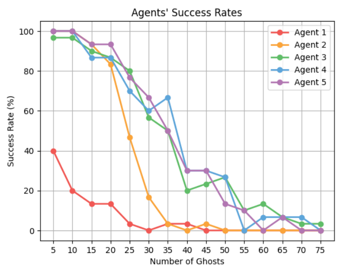

Figure 8: LI 10×10 Grid

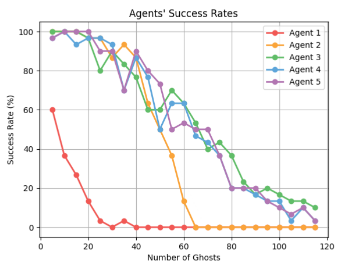

Figure 9: LI 15×15 Grid

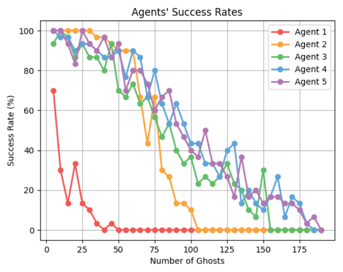

Figure 10: LI 20×20 Grid

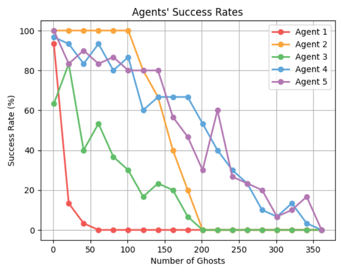

Figure 11: LI 30×30 Grid

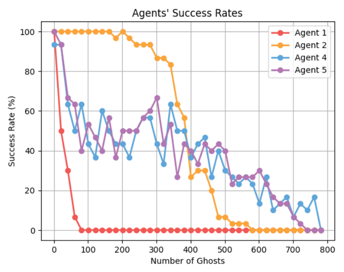

Figure 12: LI 51×51 Grid

each of the steps, it must also simulate the ghosts' movement and, therefore, it can be said it forecasts $d = 7$ steps into the future before each step it takes. All 30 simulations time out at 4 minutes for $A4$ at 441 ghosts. Note that $A2$ still survives as number of ghosts increase since it does not timeout.

4.3 Low Information Experiments

Figures 7 - 12 illustrate the results of running Agents 1 - 5 in low-information environments.

- 5×5: $A5$ performs best, surviving up to 28 ghosts.
  - Rankings: $A1 < A2 < A3 \leq A4 \leq A5$
- 10×10: $A3$ performs best, surviving up to 76 ghosts.
  - Rankings: $A1 < A2 < A3 \leq A4 \leq A5$
- 15×15: $A3$, $A4$, $A5$ performs best, surviving up to 116 ghosts.
  - Rankings: $A1 < A2 < A3 \leq A4 \leq A5$
- 20×20: $A5$ performs best, surviving up to 186 ghosts.
  - Rankings: $A1 < A2 < A3 \leq A4 \leq A5$
- 30×30: $A5$ performs best, surviving up to 342 ghosts.
  - Rankings: $A1 < A3 \leq A2 \leq A4 \leq A5$
- 51×51: $A4$ performs best, surviving up to 767 ghosts.
  - Rankings: ≤ 450 ghosts: $A1 < (A4 \leq A5) < A2$
  - Rankings: > 450 ghosts: $A1 < A2 < A5 \leq A4$

The data helps us conclude that, on average, the performance of $A1 < A2 < A3 \leq A4 \leq A5$ in the low information environments. The data collected on the 51×51 grid is inconsistent with the data of the other grids. On the 51×51 grid, $A2$ outperforms the others in environments with up to 450 ghosts while $A4$ outperforms the others in environments with over 450 ghosts. The former can be attributed to the fact that, unlike the others, $A2$ does not plan for the future and, therefore, just takes the next best step. For a relatively small amount of ghosts ($\approx 17\%$ of the grid), this proves to be successful for $A2$. However, as the number of ghosts increases, it is imperative that agents forecast the location of the ghosts to ensure that they will not find themselves in an irreversible situation. $A4$ and $A5$ do plan for the future with respect to ghosts and are, therefore, more successful than $A2$ as the number of ghosts increase; as a result, more planing is required which results in more timeouts.

	Agent 3		Agent 4		Agent 5	
	Timeouts	%	Timeouts	%	Timeouts	%
HI 20 x 20	1 out of 930	0.11%	0 out of 1110	0.00%	N/A	N/A
HI 30 x 30	25 out of 420	5.95%	8 out of 630	1.27%	N/A	N/A
HI 51 x 51	N/A	N/A	199 out of 660	30.15%	N/A	N/A
LI 20 x 20	1 out of 900	0.11%	6 out of 1080	0.56%	7 out of 1110	0.63%
LI 30 x 30	31 out of 330	9.39%	14 out of 570	2.46%	17 out of 570	2.98%
LI 51 x 51	N/A	N/A	252 out of 1200	21.00%	250 out of 1130	22.12%

$A4$'s better performance than $A5$ on the 51×51 grid can be attributed to the agents' difference in information and, consequently, run times. $A5$ times out in 22.12% of its simulations as, on average, it must account for more information per step in comparison to $A4$. $A4$ decides its next step based on the k-closest visible ghosts, which is a smaller subset of the overall ghosts. $A5$, on the other hand, stores a ghost's last seen location in memory as long it has been seen within the last 51 steps. That means that $A5$ generally stores more ghosts than $A4$, which in turn means that $A5$ must account for a larger number of ghosts when making its next move and, therefore, will take a longer amount of time per step. All 30 simulations timeout at 4 minutes for $A5$ at 722 ghosts. Hence, because of the timeouts, we are not surprised that $A4$ outperforms $A5$ in the simulations we ran for the 51×51 grid.

4.4 Hung Simulations

For $5 \times 5, 10 \times 10, 15 \times 15$ grids in the HI and LI environments, there are no timeouts for any of the agents. Note the following key observations: (1) in LI environments, $A5$ has similar timeout rates to $A4$ since it bootstraps off $A4$; (2) as the grid sizes increase, $A4$ and $A5$'s plan for the future is more efficient than $A3$'s plan for the future.

For a 20×20 grid in a HI environment, $A3$ timeouts 0.11% of the time and $A4$ never timeouts, suggesting that $A3$'s timeout was an outlier. For a 20×20 grid in a LI environment, $A3$ timeouts 0.11% of the time, $A4$ times out 0.56% of the time, and $A5$ times out 0.64% of the time. Just like the HI case, we argue that the $A3$ timeout is an outlier; we argue that $A4$ and $A5$ timeout more than $A3$ because these agents plan in the future based on the number of ghosts and not just survivability; hence, since the environment is partially observable, $A4$ and $A5$ take longer wall-clock time.

For a 30×30 grid in a high-information environment, $A3$ times out 5.95% of the time whereas $A4$ times out 1.27% of the time. For a 30×30 grid in a low-information environment, $A3$ times out 9.39% of the time whereas $A4$ times out 2.46% of the time and $A5$ times out 2.98% of the time. This suggests that $A3$ spends more time planning the future than $A4$ and $A5$ as the grid size increases.

For a 51×51 grid in a high information environment, $A4$ timeouts 30.15% of the time. For a 51×51 grid in a low information environment, $A4$ timeouts 21.00% of the time and $A5$ timesout 22.12% of the time. This suggests that as the grid size scales, the wall-clock time of the $A4$ and $A5$ increase.

For a 51×51 grid in LI and HI settings, figures 15 and 18 show that termination rate percentage decreases almost linearly as the ghosts in the maze increase for $A4$ and $A5$. We hypothesize that there are fewer terminations as the number of ghosts increases since the survivability of the agent decreases, meaning that the agent dies before enough time has been expensed to warrant a timeout or hung simulation. For a 30×30 grid in LI and HI settings, per Figures 14 and 17, $A3$ timed-out more than $A4$ or $A5$ when $A3$ had non-zero survival, suggesting that $A3$ plans for the future less efficiently than $A4$ or $A5$. For a 20×20 grid in LI and HI settings, per Figures 13 and 16, we observe few if any terminations for all of the agents, indicating that our agents can handle a small grid like 20×20 without much problem in wall-clock time.

4.5 Confidence Intervals

Figures 1 - 12 initially included a 95% confidence interval for each data point. The upper and lower bounds of these intervals are not visible as the Y-axis of the graphs was scaled from 0% - 100%. In other words, the confidence intervals did not impact the overall trend lines or deviate from the conclusions we reached. Therefore, we removed them from the graphs.

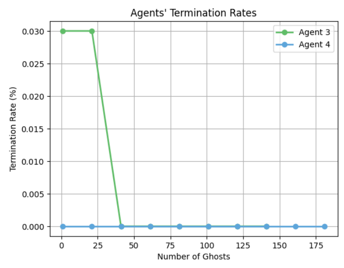
Figure 13: HI 20 × 20 Grid

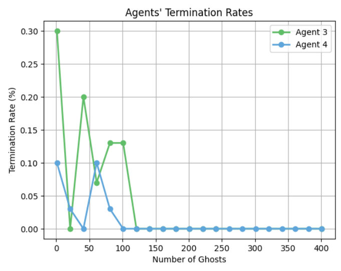
Figure 14: HI 30 × 30 Grid

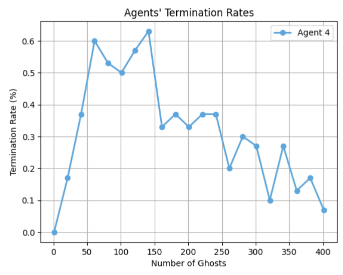
Figure 15: HI 51 × 51 Grid

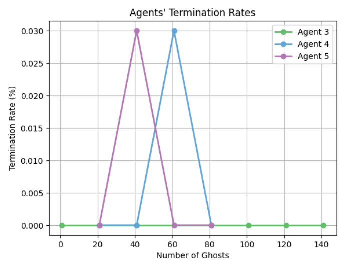
Figure 16: LI 20 × 20 Grid

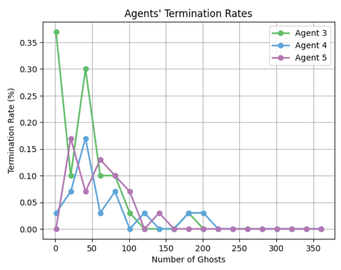
Figure 17: LI 30 × 30 Grid

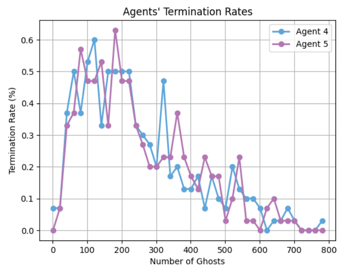
Figure 18: LI 51 × 51 Grid

5 Discussion

5.1 Challenges in Scaling up Agents to Work on Larger Grids

One of the initial computational bottlenecks we ran into was developing a suitable representation of our `ghost-in-the-maze` simulator. We created a `Block` class and had mazes composed of `Block` objects on which `Agents` and `Ghosts` interacted with. Quickly, we found that $A1$ took an intractable time to run even once on a 51×51 maze. Hence, we revised our representation and refactored our entire codebase to store `numpy` matrices of the grid, where 0 is unblocked and 1 is unblocked. Similarly, we stored a ghost grid, where 0 is no ghost present, and 1 is a ghost present. With fast numerical processing, we quickly were able to run $A1$ and $A2$ on a 51×51 grid. The second challenge with scaling up Agents was determining how many simulations to run in Agent 3 and how far to forecast in the future, which our discussion in section **3.3** covers. The third challenge was determining how far to forecast $A4$ into the future and by consequence $A5$, which is covered by discussion in section **2.4**. As we scale up, we note trade-offs between intelligence and efficiency.

5.2 Challenges in Data Collection for Experiments

For the 20×20, 30×30, 51×51 grids, we ran the simulations on `ilab` machines. In an effort to speed up data collection, we ran each agent on a different `tmux` session. We collected the wins, losses, and terminations of each agent at increments of 20 ghosts. The process was further split on the `tmux` sessions, such that each session would run a maximum of 10 increments of ghosts or 300 total simulations. In turn, we were not able to run to guarantee that all the agents would start on the same initial environment with each new simulation. However, the data is considered to represent the average success rate so we thought it would still be a valid measure of overall survivability.

For the 51×51 grid, we were unable to collect data for $A3$ on the High Information and Low Information environments since the time for data collection would be more than several days. While we did try to run these experiments for $A3$, our sessions were terminated on `ilab` machines post-24 hours. Then, we tried to run multiple `tmux` sessions to see if we could run multiple simulations on each session and then aggregate the results later on - Agent 3 simply took too long to run and our sessions were terminated again. Hence, we decided to run experiments on several grids of smaller sizes. Fortunately, across all grids, we see similar trends in relative performance for the agents.

Another limitation of our results is that, for larger grids, $A3$, $A4$, and $A5$ timed out. For the purpose of our graphs, we treated time-outs as failure states. As you can observe from our figures, the timeouts increased as the maze size increased for computationally intensive agents like $A3$ and occasionally for $A4$ and $A5$. If we had more time for this project, we'd run these experiments without time-out limits and collect data that way to be most precise with our results. Alternatively, we would also look into ways to further optimize our agents to gain as large of a speed boost as possible in terms of both wall-clock time and computational complexity.

In addition, we worked under the assumption that once an agent has a 0.00% survival rate with a certain number of ghosts, it will maintain the 0.00% survival rate for all increasing numbers of ghosts. We worked under this assumption for the purpose of our graphs as we wanted to maintain a continuous line with each agent until all the agents were not able to survive. It seemed unnecessary to run the agents past their survival as the results of the smaller grids indicate that there are minimal cases in which an agent is able to sustain a higher number of ghosts after its initial death.

5.3 Limitations

For $A3$, $A4$, and $A5$, we manually determined the values of the parameters for the heuristic functions, but they might not necessarily have been the best parameters. If we had more time on the project, we would try to use grid-search to find better parameters for the heuristic functions. Another strategy we could do is to make all of our heuristics functions differentiable and use a gradient-based learning approach to "learn" the parameters of the function that maximizes a relevant objective.

In a similar vein, a limitation of all our agents, and classical artificial intelligence in general, is that we hard-code sophisticated heuristics and logic into our agents. But, it is not necessarily the case that the logic and heuristics we encode are the most relevant. Quite possibly, there are strategies for agents that might be better than the ones we encoded manually. That motivates a gradient-based learning approach for more sophisticated agents. If we had more time, we would look into building a "deep" reinforcement learning agent that maximizes the "reward" given by an action at every timestep through a differentiable function $Q(s, a)$ that takes as input the current state of the board s as well as some action a that we are evaluating. We would also have to look into how to best shape the reward.

6 Conclusion

We developed a simulator that hosts our 5 agents in both high and low-information environments. We categorize the environment as HI or LI based on whether the agent has access to all the ghosts or just the ones that are visible on the grid. Agent 1 naively plans a path from the start to the end using BFS, ignoring the ghosts. Agent 2 builds off of Agent 1 and plans another path when there is a ghost blocking its move or, in the worst case, moves away from the nearest ghost. Agent 3 runs Monte Carlo simulations of Agent 2 for each of its potential actions and, from there, chooses the next step based on a heuristic that accounts for the success of those simulations, amount of times the location has been previously visited, and the distance from the goal. Agent 4 uses a variant of minimax and considers the distance from k closest ghosts in addition to curiosity and distance to the end goal for each step it takes. Agent 5 builds off of Agent 4, additionally storing all the ghosts that were seen in the last n moves, where n is the width of the grid. We ran these agents on 6 different size grids and were able to generalize that in the HI environments $A1 < A2 < A3 \leq A4$, while $A1 < A2 < A3 \leq A4 \leq A5$ in the LI environments.