CS 520 Project 1

Ghosts in the Machine

Submitted by

Chandhanu Mohan Kalamani – cm1573
Sai Chaitanya Chaganti – sc2482

Instructor

Dr. Charles Cowan

October 14, 2022

Contents

1. Project Statement

2. Implementation

    2.1 The Environment
    2.2 The Agent
    2.3 The Problem: Ghosts
        2.3.1 Implementation

3. Strategy

    3.1 Agent 1
        3.1.1 Implementation
    3.2 Agent 2
        3.2.1 Implementation
    3.3 Agent 3
        3.3.1 Implementation
    3.4 Agent 4
    3.5 Agent 5

4. Analysis

    4.1 Agent 1
    4.2 Agent 2
    4.3 Agent 3
        4.3.1 Comparing Agents 1,2 and 3
    4.4 Agent 4
    4.5 Agent 5

5. Conclusion

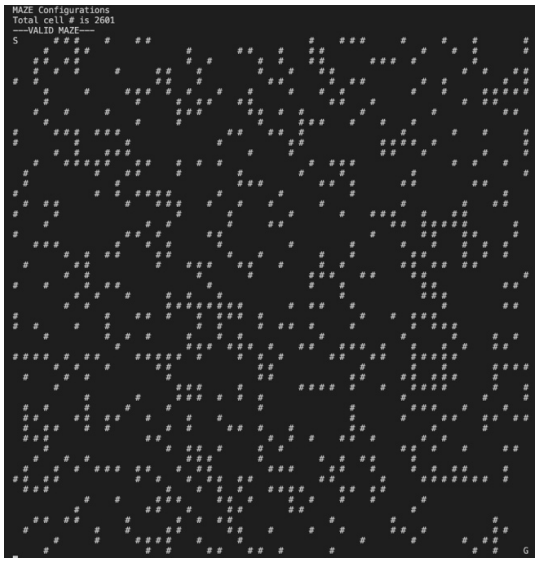
1 Project Statement

A few goals are served by this initiative. Initially, to offer you practice developing and applying search algorithms to problem-solving. Second, in order to emphasize the distinction between planning and execution, a plan may be founded on insufficient data and may require revision or adaptation as it is carried out. In this project, you'll create an environment, an agent that must interact with that environment to achieve a goal, and data that will be used to evaluate how well your agent performed in achieving that goal.

2 Implementation

2.1 The Environment

A square grid(maze) serves as the problem's environment. Some of the cells are unblocked and open, while others are blocked. The unblocked cells are open for an agent in the environment to move through and occupy, but the blocked cells are closed off. We created a randomized 2D array of size 51x51 to represent the maze while keeping the probability of a cell being blocked as 0.28. The blocked cells are characterized by '#' and the unblocked cells as empty spaces. The Start and Goal cells are characterized by 'S' and 'G' respectively. However, because the cells are dispersed at random, it's very conceivable that every cell will be blocked in these mazes, they might not be very good. Therefore, we evaluated the maze's quality and excluded any that are too obstructed by making sure there is a path between 'S' and 'G' (going up, down, left, and right), and removed any obstructions encountered.



Q. What algorithm is most useful for checking for the existence of these paths? Why?
A. We are using DFS for checking the existence of a path because of its low memory utilization, scalability and fast validation.

2.2 The Agent

Starting in the top left corner(S), the agent will try to get to the bottom right corner(G). The agent may only travel within unblocked cells within the 51x51 grid and only in the four cardinal directions (up, down, left, and right). The agent can always see the whole maze and utilize that knowledge to plot their course.

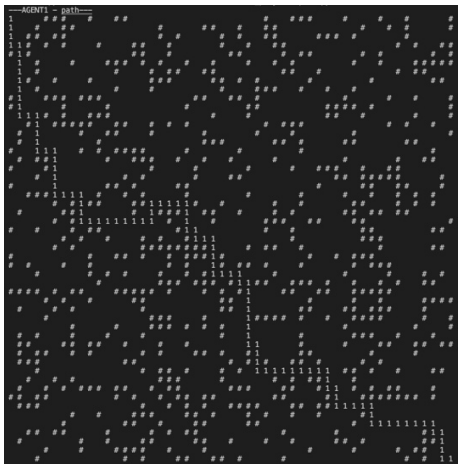


Image showing the path of Agent (represented by 1)

2.3 The Problem: Ghosts

Unfortunately for the agent, the maze is full of ghosts. Each ghost starts at a random location in the maze. The agent dies if they enter a cell where a ghost is present or if a ghost enters their cell. This should be prevented. The ghosts will follow the agent's every move. This implies that any strategy the agent used to navigate the maze may at any time become obstructed or useless. In order to try to avoid the ghosts, the agent may need to re-plan their way through the maze, and they may need to do it frequently as the ghosts move. A ghost selects one of its neighbors (up, down, left, or right) at each timestep; if the neighbor is unblocked, the ghost goes to that cell; if the neighbor is blocked, the ghost either stays in position with a probability of 0.5 or moves into the blocked cell with a probability of 0.5. Even if the ghost is now within a restricted cell, these guidelines still apply.) The ghosts follow the aforementioned rule whenever the agent travels. The agent dies if they make contact with a ghost.

2.3.1 Implementation

A ghost is characterized by 'X' in our maze. Ghosts are generated randomly throughout the maze. Cell values are updated to 'X' post generation in the maze. Ghost locations and the nature of the cell (Blocked/Unblocked) are stored using hashmap. The start and finish nodes are not populated with ghosts during the ghost filling process. The "advance ghosts ()" module to advance the ghost is called each time the agent travels one block. Each ghost's randomized

motions are generated by this module, which also alters the block's nature and its indices in the hash map. The ghost has a 50% chance of moving into a blocked cell and can enter any free cell. The ghost, however, cannot occupy a cell that is already occupied by another 'host. Based on the aforementioned requirements, this module makes sure that all ghosts are given random motions.

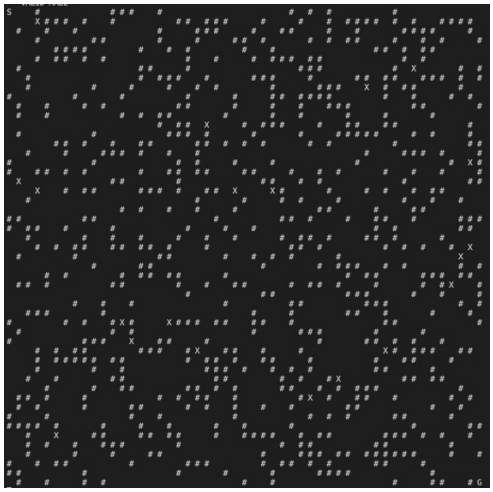


Image showing Ghost(X) locations

3 Strategy

3.1 Agent 1

Agent 1 determines the quickest route through the maze and follows it, oblivious to the ghosts. This agent is very effective since it only needs to plan a course once, but it doesn't update or change its behavior in response to changing environmental conditions.

3.1.1 Implementation

Agent 1 determines the shortest route using BFS between the Start cell Index (0,0) and the Goal cell Index (51,51). If it is impossible to find the path from Start to Goal, Agent 1 declares the mission a failure. Agent 1 simply chooses a path that looks good and pursues it. Agent 1 moves along its intended course, passing through each block unaware of the ghosts nearby or in its way. Agent 1 does nothing but go on if a ghost or ghosts enter the planned path. Agent 1 dies when a ghost enters the cell or when the Agent 1 visits the block where the ghost is located.

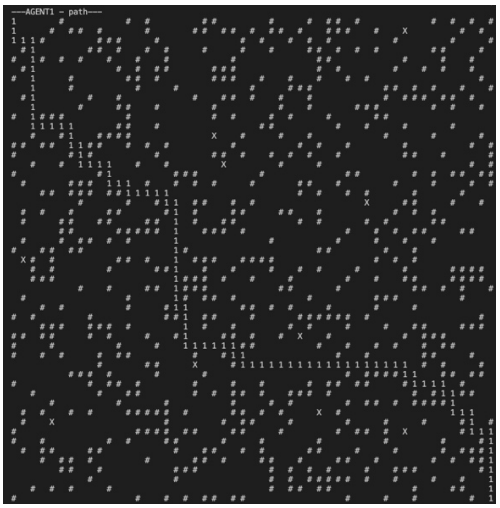


Image showing the path of Agent 1 with Ghosts

3.2 Agent 2

Agent 2 recalculates. Every time a time step occurs, Agent 2 recalculates a new path to the goal based on the available data and then performs the subsequent step on this new path. Agent 2 is continually revising and readjusting considering fresh ghost knowledge. Agent 2 does not predict the future. Agent 2 tries to get away from the closest visible ghost if all routes to the goal are currently blocked (not occupying a blocked cell).

3.2.1 Implementation

In contrast to Agent 1, Agent 2 uses the A* algorithm to replan its route from its current position to the goal node at every time step, employing the Manhattan distance as a heuristic. Furthermore, Agent 2 continuously tracks the ghosts' location and modifies its plans based on respective positions. Agent 2 is designed to plan best when the ghosts completely blocks all routes to the goal node. If every path to the goal node is blocked. Agent 2 then makes an effort to withdraw itself from the closest ghost by acquiring the ghost's indexes and determining which ghosts are closest to it by gauging each ghost's distance from it. Once the closest ghost and its distance from Agent 2 are obtained, Agent 2 will scan the neighboring nodes (up, down, left, and right) into which it may be able to enter and will determine the Manhattan distance from those nodes to the closest ghosts. Following this phase, Agent 2 moves to the neighbor node with the greatest determined distance. Agent 2 will continue this procedure until it reaches the goal node by replanning from its new position to the goal node.

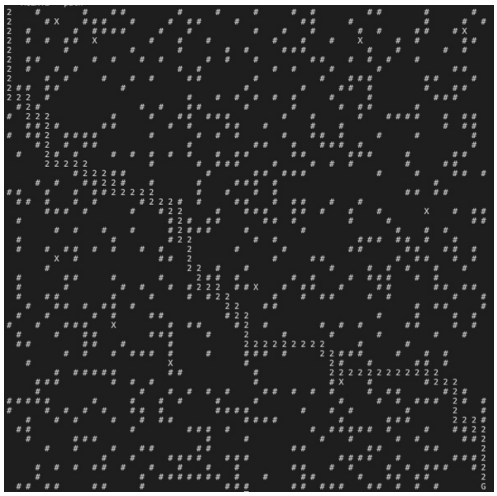


Image showing Agent 2's path with Ghosts

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?
A. We do not always need to replan. If the existing plan allows agent 2 to reach the goal, Agent doesn't need to calculate a new plan. In this case the new plan will be the same as the old plan. The agent only needs to strategize a new plan if the old one leads to failure.

3.3 Agent 3

Agent 3 projects. Agent 3 evaluates all potential actions (including remaining put) at every time – step and "simulates" the future based on Agent 2's rules thereafter. This future is projected a certain number of times for each potential action, and Agent 3 then selects the actions with the highest success rates in these simulations. Agent 3 can be compared to Agent 2 with the addition of future vision.

3.3.1 Implementation

Agent 3 is identical to Agent 2 but employs two layers of heuristics (the Manhattan distance and the survivability index), and in addition to the four actions of moving up, left, down, and right, Agent 3 may also execute a fifth operation called "Wait". Along with the aforementioned heuristics, Agent 3 employs the A* algorithm to determine the route from the beginning to the end. Agent 3 does a wait operation and rechecks the path if all paths are blocked. Agent 3 replans by calculating the Manhattan distance and retrieving the survivability index once the ghosts are relocated and the path has been opened. The agent 3's current position's prospective neighbors are listed as keys in the survivability index, and the survivability data are listed as values. The survivability from the agent 3's neighbors to the goal node is calculated by repetitively mimicking the agent 2 from the position of the nodes to the goal. The subsequent node for agent 3 to occupy is the neighbor node with the highest success rate. Agent 3 continues in this manner until the objective is attained. Agent 3 is designed to only replan when

ghosts obstruct the original path it computed. Agent 3 may be able to wait forever while using the wait operation until the path is opened or until agent 3 is killed. The number of waits it may complete at one time is constrained by the establishment of a threshold of 50.

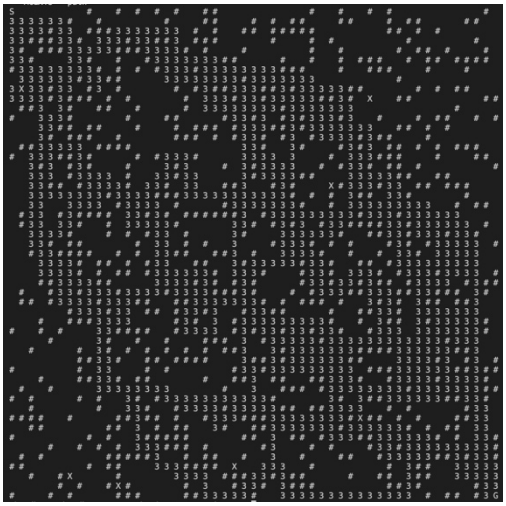


Image showing Agent 3's path with Ghosts

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?
A. There may be instances where all simulations have incredibly low success rates, showing that ghosts continue to obstruct the intended path. When an agent is faced with such uncertainty, it typically stays in its existing cell in the belief that the ghosts would continue, resulting in improved success rates in the subsequent time-step.
But success is not always impossible just because all the possible future routes are shut. We cannot be guaranteed that the ghosts will move in the fashion anticipated by the simulation of agent2 since their movement is randomized. This ambiguity in a dynamic environment shows that, logically, the likelihood of survival nearly never approaches zero.

3.4 Agent 4

Offering the same two layers of heuristics and the ability to execute "wait" operations as Agent 3, Agent 4 is a bootstrapped variant of Agent 3. Agent 4 is programmed to only replan when ghosts obstruct the path it calculated. Additionally, Agent 4 executes Agent 2 to produce heuristics and estimate the survival index of the neighboring nodes at its current position. Agent 2 is allowed to perform the wait with a threshold of 200 times in each of these scenarios. Furthermore, Agent 4 runs Agent 2 at least twice as quickly as Agent 3 in order to gather superior heuristics to Agent 3. Moreover, Agent 4 executes at least twice as many wait operations as Agent 3.

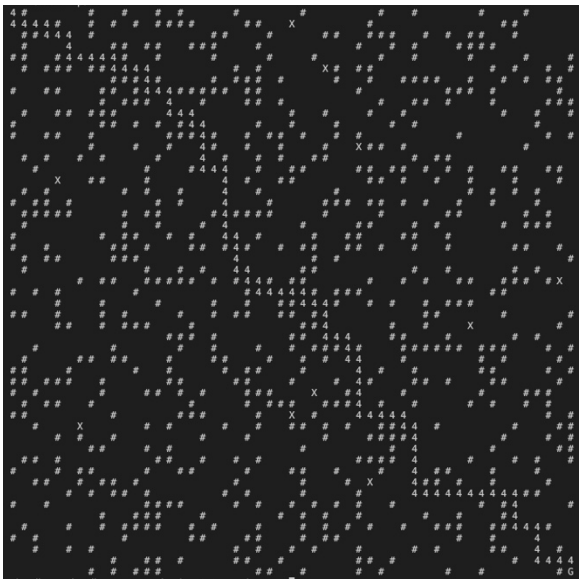


Image showing Agent 4’s path with Ghosts

3.5 Agent 5

In terms of implementation, Agent 5 is similar to Agent 4, however when a ghost occupies a block, the ghost indexes are not changed.

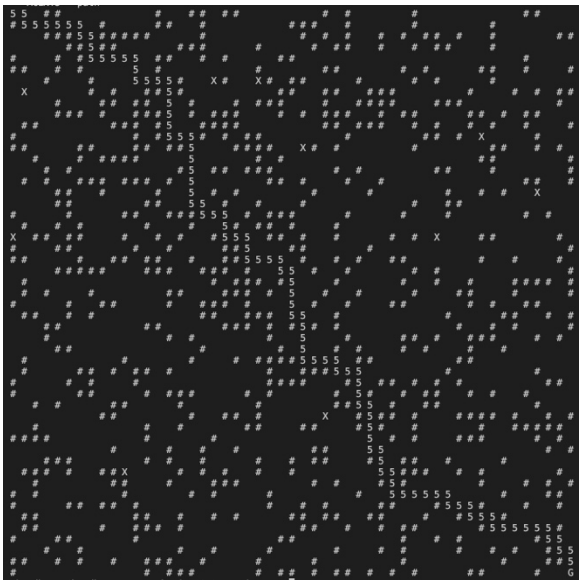
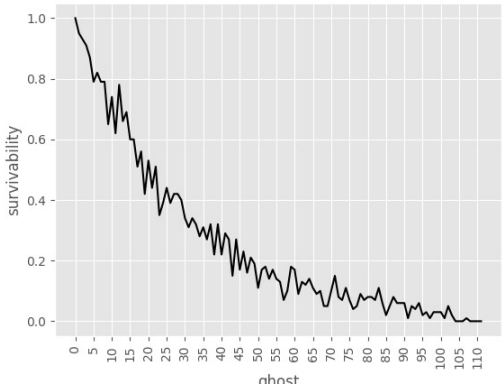


Image showing Agent 4’s path with Ghosts

4 Analysis

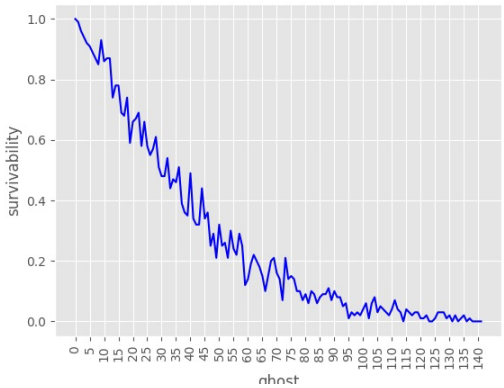
4.1 Agent 1

Agent 1 only chooses one course of action during the planning process. Consequently, this delivers the quickest and most effective outcomes when compared to other agents. Agent 1's lack of intelligence, however, is where Survivability fails. Agent 1's ability to survive in the simulations decreased considerably as the number of ghosts increased. Agent 1 almost dies when there are more than 100 ghosts.



4.2 Agent 2

In every simulation, agent 2's survival is rather better than agent 1's since it replans whenever the path is blocked. When agent 1 dies, agent 2 can survive in a maze with roughly twice as many ghosts.
Limitations: Because Agent 2 can walk away from the next ghost, in the event of obstructions at any point, Agent 2 may have to move away from ghosts endlessly. As a result, Agent 2 may not be able to go forward at all in some outlier labyrinth situations.
Additional adjustments: To address the constraint indicated above, a Threshold of 50 is created. When 50 wait procedures had been carried out, obstructions were still present. Agent 2 has been declared dead. Consequently, this is viewed as a failure.

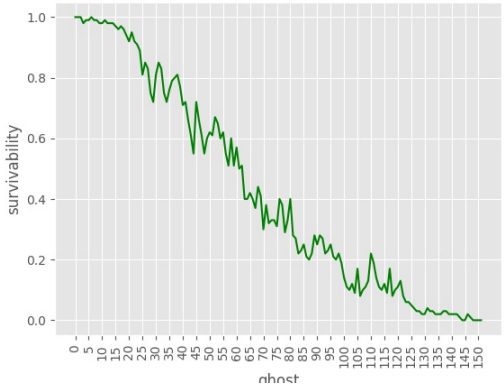


4.3 Agent 3

Agent 3: Based on the graphs, Agent 3's survival has greatly increased. Additionally, the agent may endure 150–170 ghosts. But the Agent's capacity to survive comes at a cost in terms of temporal complexity. Agent 3's ability to wait naturally increased the temporal complexity.

Limitations: Because Agent 3 may emulate Agent 2 and wait, it takes Agent 3 longer to clear even mazes with fewer ghosts. Additionally, the number of ghosts increased along with this time interval.

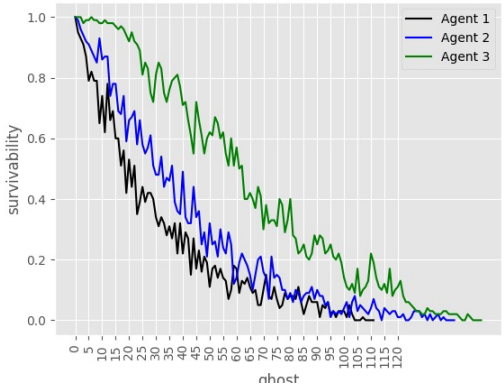
Threshold is added to prevent agent 3 from waiting endlessly, among other changes.



4.3.1 Comparing Agents 1,2 and 3

Q. How did Agents 1, 2, and 3, compare? Was one always better than others, at every number of ghosts?
A. The obvious drawback of Agent 1 is that it does not scan its run for ghosts. Agent 2 replans after checking for ghosts along the way. As a result, it outperforms Agent1. But it also has a

serious defect. Ghosts close to the path can attack it. Agent 2 is reprogrammed by Agent 3 to look for the direction Agent 2 would go in each cell. Whereas Agent 2 follows a clear strategy throughout its course, Agent 3 continuously replans and does not adhere to any single course. Therefore, Agent 3 works better than Agents 1 and 2 at every number of ghosts.

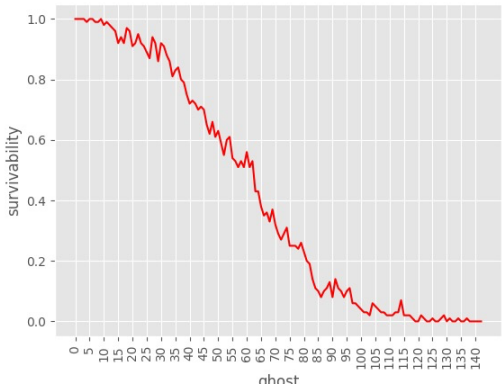


4.4 Agent 4

Agent 4: At first, agent 4 has considerably superior survival in fewer ghosts, but as the number of ghosts increases, agent 4's survivability approaches that of agent 3. Due to the threshold values specified for agent4, this has transpired. In environments with fewer ghosts, the threshold is hitting less frequently; in environments with far more ghosts, the threshold is hitting more frequently and induced rendering failure. Increased threshold numbers allow agent 4 to stay forever among previously visited nodes, considerably lengthening the time required for each simulation that agent 4 completes to complete the mazes. Because it can withstand even more ghosts than agent 3, agent 4's survivability is nearly as good as agent 3's on average. However, because it requires more computing and takes longer to fulfill tasks, the net computational cost is greater.

Limitation
It is expensive to compute.
Debugging gets more difficult after iteration.
Simulator data collection is challenging.

Threshold values are chosen and updated following several iterations, among other adjustments. Additionally, it often increases somewhat.

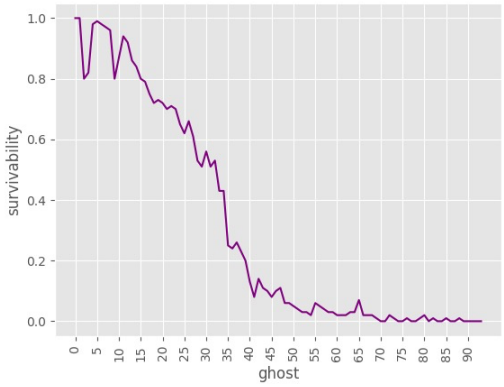


4.5 Agent 5

Agent 5:

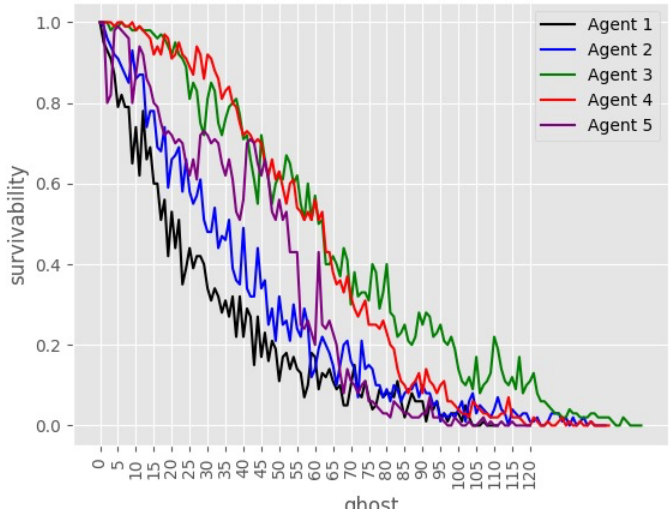
Agent 5 outperforms Agents 1 and 2, but not Agents 3 and 4. Agent 5 occasionally performs on par with agents 3 and 4, however in a few outlier labyrinth conditions, it also performs as poorly as agent 2.

Limitations: Agent 5's performance falls better within the performance range of Agent 3 and Agent 2 since Agent 5 cannot notice ghosts in the blocks unless the ghosts leave the blocks.



5 Conclusion

As we can see from the information above, the agents function effectively in terms of both survivability and effectiveness, despite the fact that they are by no means flawless. To further increase survivability, stronger agents can be developed which can bootstrap the aforementioned agents or calculate better heuristics.


Agent 1
Agent 2
Agent 3
Agent 4
Agent 5
survivability
1.0
0.8
0.6
0.4
0.2
0.0
0
5
10
15
20
25
30
35
40
45
50
55
60
65
70
75
80
85
90
95
100
105
110
115
120
ghost