Assignment III

Intro to Artificial Intelligence

CS 16:198:520

Kaustubh N. Jadhav
knj25

Pranav Shivkumar
ps1029

Sanyam Jain
sj770

Introduction

In this project, the idea is to create a landscape represented by a map of cells. These cells are of various terrain types which determines how difficult it is to search. A target is hidden in one of cells somewhere in the landscape and the goal is to find this target.

Representation of the Landscape

In order to represent the landscape, we have created a class, ProbabilisticHunting, which initializes the parameters like the size of the landscape, the probabilities of occurrence of each terrain type and false negative rates for each square based on its terrain type.

For the implementation of the landscape, each cell of the landscape is randomly assigned a terrain type according to the following probabilities of occurrence: **flat**: 0.2, **hilly**: 0.3, **forested**: 0.3, **caves**: 0.2.

The corresponding false negative rates (which represents how difficult it is for to search for the target in these cells) for these terrain probability types are set as follows: **flat**: 0.1, **hilly**: 0.3, **forested**: 0.7, **caves**: 0.9. The false positive rates for any cell is taken to be 0.

One cell of the landscape is randomly assigned as the target. The sample generated landscapes of size 10 and size 20 are shown below:

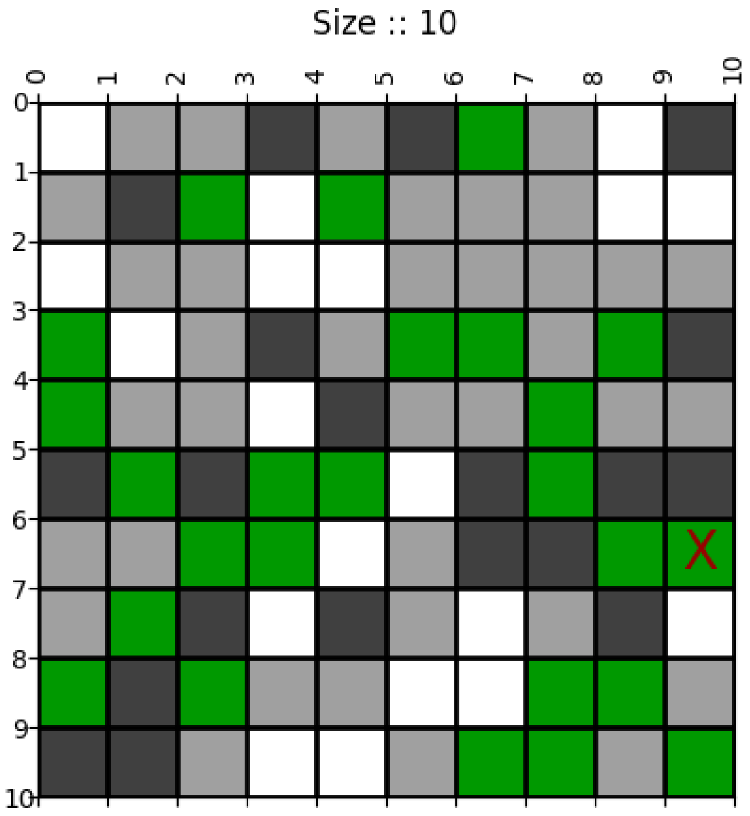

Figure 1: Sample Landscape of size 10x10

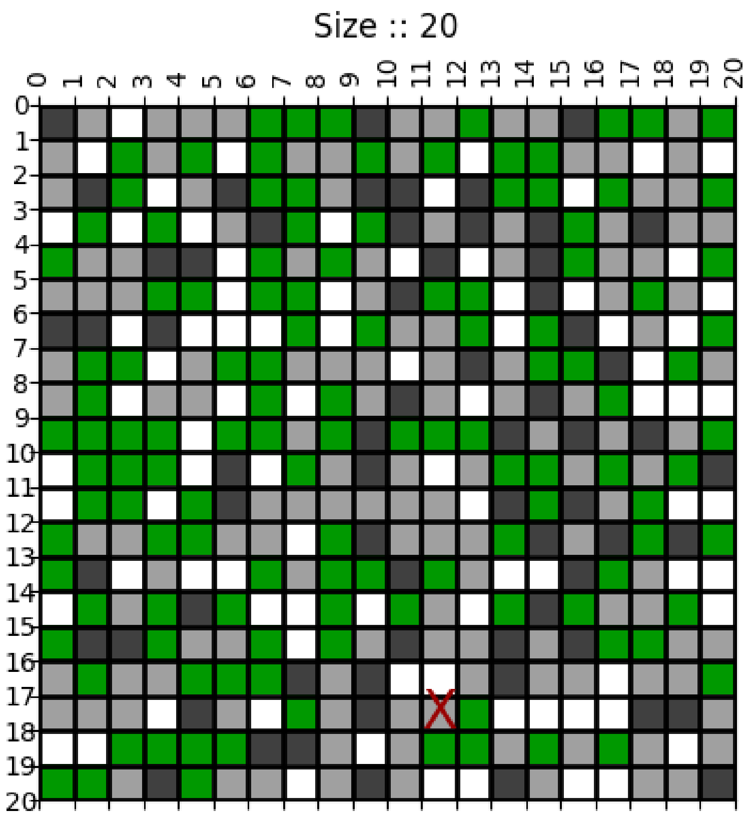

Figure 2: Sample Landscape of size 20x20

In the initial state, the target can be located anywhere in the landscape since we don't have any knowledge regarding the cells. Because of this, the location of the target in the landscape can be anywhere; as a result, the likelihood of any cell containing the target is represented by:

$$P(Target\,in\,Cell_i) = \frac{1}{\#\,of\,cells} \tag{1}$$

Starting with this information, the goal is to locate the target in as few searches as possible.

Part 1: A Stationary Target

Introduction

In this part, the target is stationary i.e. it remains in the exact same location as it was at time t=0, while the search for the target progresses.

As we search a cell for the target, there are two possible scenarios that can occur. The first of these is that the target is found and the search is terminated, the other is that the target is not found during the search and the search continues.
We are interested in the second observation since by using this observation, we can update the probability of finding locating target in the cells. For any cell:

$$P(Target\,not\,found\,in\,cell) = P(Target\,not\,in\,cell) + P(Target\,in\,cell\,and\,not\,found) \tag{2}$$

For all the cells, updated probabilities of locating the target can be calculated using this observation probability. Let $\mathbf{Cell_j}$ be the cell that was currently searched, then

$$\begin{gathered} P(Target\, not\, found\, in\, \mathbf{Cell_j}) = (1 - P(Target\, in\, \mathbf{Cell_j})) + \\ P(Target\, in\, \mathbf{Cell_j}) \times P(Target\, not\, found\, in\, \mathbf{Cell_j} \mid Target\, in\, \mathbf{Cell_j}) \end{gathered} \tag{3}$$

Also, for any $\mathbf{Cell_i}$ from the landscape excluding $\mathbf{Cell_j}$, updated probability of locating the target will be:

$$P(Target\, in\, \mathbf{Cell_i} \mid Target\, not\, found\, in\, \mathbf{Cell_j}) = \frac{P(Target\, in\, \mathbf{Cell_j})}{P(Target\, not\, found\, in\, \mathbf{Cell_j})} \tag{4}$$

For $\mathbf{Cell_j}$, updated probability of locating the target will be:

$$\begin{gathered} P(Target\, in\, \mathbf{Cell_j} \mid Target\, not\, found\, in\, \mathbf{Cell_j}) \\ = \frac{P(Target\, in\, \mathbf{Cell_j}) \times P(Target\, not\, found\, in\, \mathbf{Cell_j} \mid Target\, in\, \mathbf{Cell_j})}{P(Target\, not\, found\, in\, \mathbf{Cell_j})} \end{gathered} \tag{5}$$

In addition, for any cell $\mathbf{Cell_i}$ from the landscape (including the cell last search), the updated probability of finding the target in cell after the observation will be:

$$\begin{aligned} P(Target\, found\, in\, \mathbf{Cell_i} \mid Target\, not\, found\, in\, \mathbf{Cell_j}) = {} & P(Target\, in\, \mathbf{Cell_j} \mid Target\, not\, found\, in\, \mathbf{Cell_j}) \\ & \times (1 - P(Target\, not\, found\, in\, \mathbf{Cell_i} \mid Target\, in\, \mathbf{Cell_i})) \end{aligned} \tag{6}$$

Analysis

1. **Consider comparing the following two decision rules:**

   Rule 1: At any time, search the cell with the highest probability of containing the target.

   Rule 2: At any time, search the cell with the highest probability of finding the target.

   For either rule, in the case of ties between cells, consider breaking ties arbitrarily. How can these rules be interpreted / implemented in terms of the known probabilities and belief states?

   Ans:
   Rule 1 - As per this rule, the search is based on the probability of locating the target only. Once we search a cell, if the target is found, the search is terminated; otherwise, we use the equations 4 and 5 to update the probabilities of locating the target for all the cells and the cell with maximum probability of having the target is selected to search next. In case case of tie (which happens all the time) a cell is randomly chosen. We name the agent searching with this logic as Agent1

   Rule 2 - As per this rule, the search is based on the probability of finding the target in the cell. Once we search a cell, if the target is found, the search is terminated; otherwise, we use equations 4 and 5 to update the probabilities of locating the target for all the cells. From probabilities of locating the target for the cells with equation 6, we calculate the probabilities of finding the target for all the cells and the cell with maximum probability of finding the target is selected to search next. In case of tie (which happens all the time) a cell is randomly chosen. We name the agent searching with this logic as Agent2

   For a fixed map, consider repeatedly using each rule to locate the target (replacing the target at a new, uniformly chosen location each time it is discovered). On average, which performs better (i.e., requires less searches), Rule 1 or Rule 2? Why do you think that is?

   Ans: For this comparison we ran 1000 iteration with landscape of size 50 i.e. with 2500 cells.

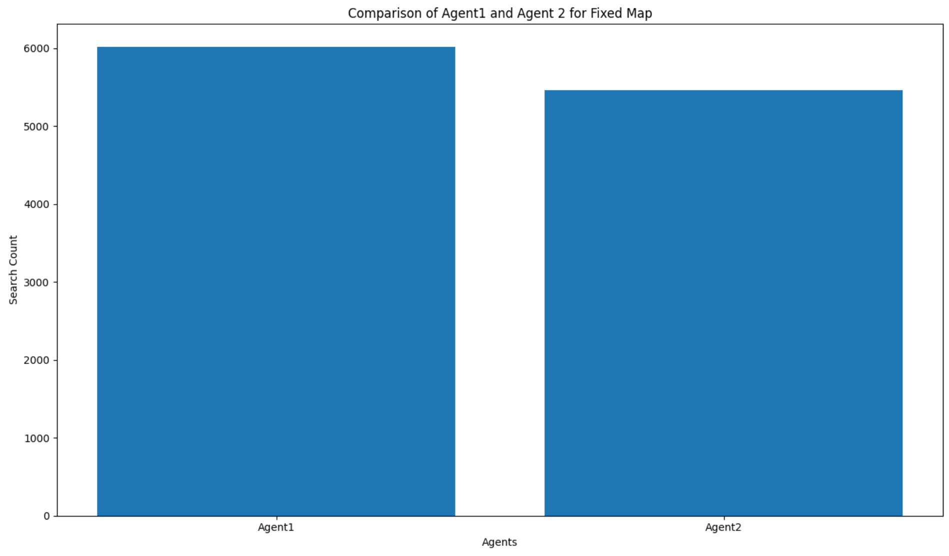

Figure 3: Comparison of Search Count for Agents for Fixed Map

From Figure 3, we can conclude that Agent 2 on average takes less number of searches than Agent 1. Figure 4 shows that Agent 2 performs way better than Agent 1 for Plain, Hilly and Forest terrain but Agent 1 outperforms Agent 2 for caves. This is expected as searching strategy for Agent 2 is based on searching all cells with high probability of finding the target, taking false negative rate of different terrain in account, Agent 2 will search plains first, then hills, then forest and caves at last. While Agents just take probability of locating target in account which is identical in many cells, Agent 1 ends up taking more random decision.

So, in case of plains, hills and forest, Agent2 will outperform Agent 1, In case target falls in cave, more randomness of Agent 1 and the fact that Agent will search all easy terrain first before reaching to caves makes Agent 1 search faster. But overall Agent 2 performs better than Agent 1.

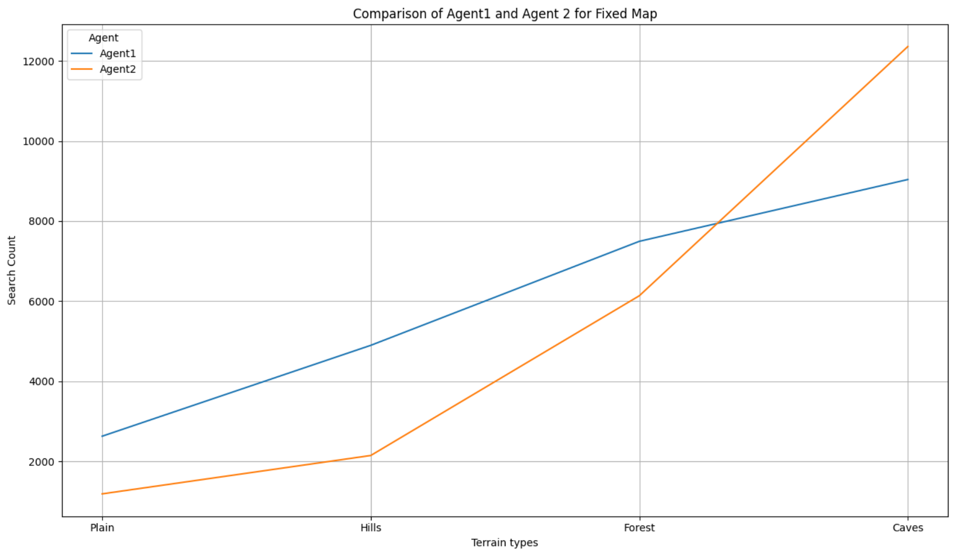

Figure 4: Search Count Comparison of Rules 1 and 2 for Terrain Types for Fixed Map

Does same conclusion holds across multiple maps?
Ans: Yes, same conclusion i.e. Agent 2 on average outperforms Agent 1. Agent 1's performs better than Agent 2 for caves. This conclusion holds true as there is not much difference w.r.t Agents in both scenario (Fixed or Multiple Maps). Every time we reset target in case of fixed map, we also reset probabilities and search start from basic initial probabilities only which like having a new map.

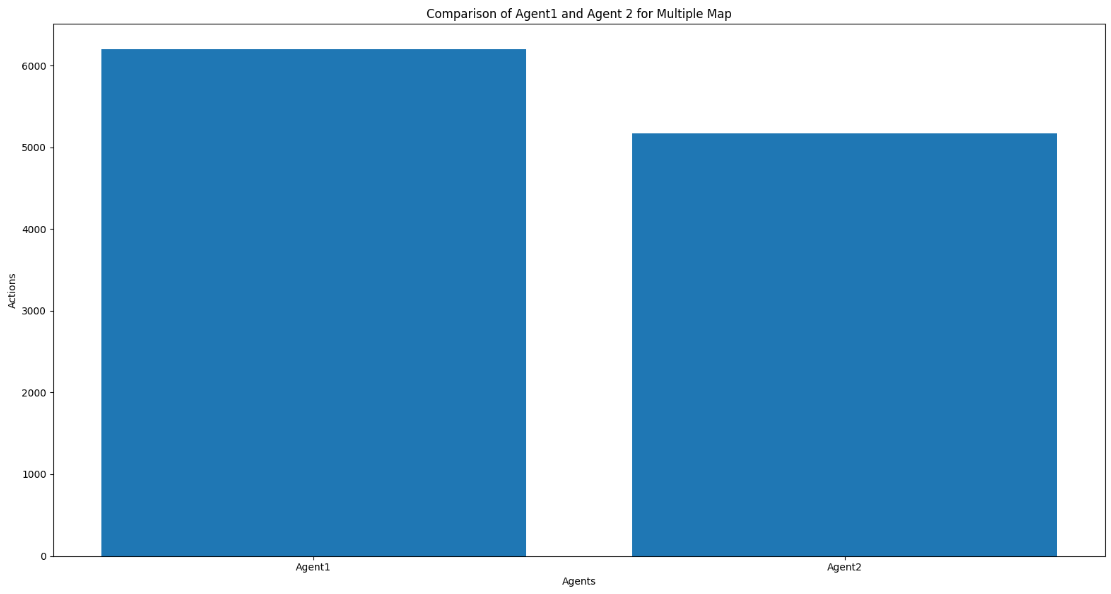

Figure 5: Comparison of Search Count for Agents for Multiple Map

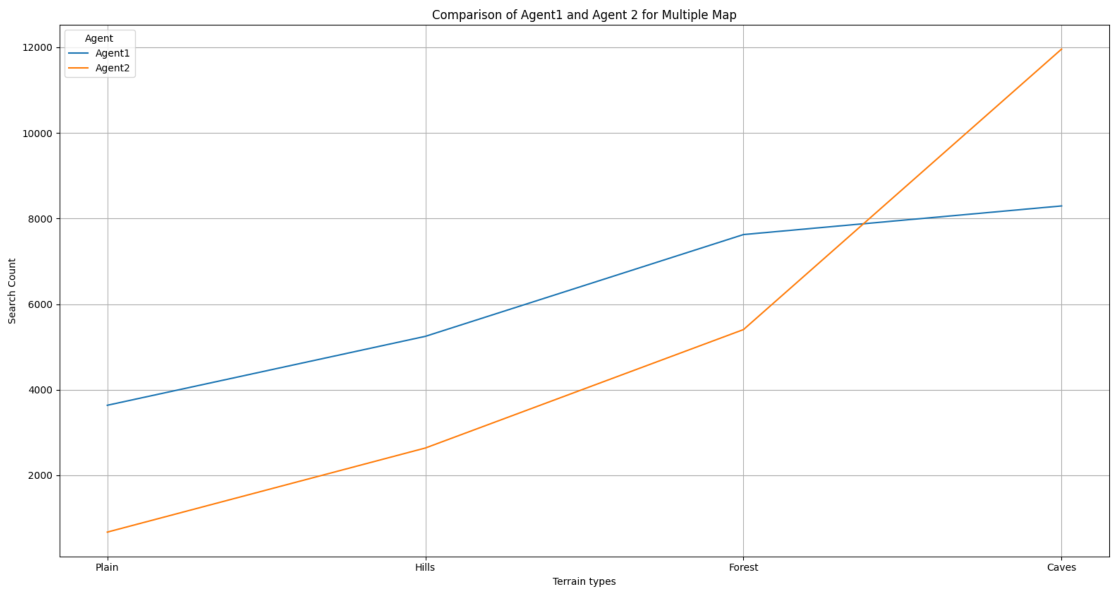

Figure 6: Search Count Comparison of Rules 1 and 2 for Terrain Types for Multiple Maps

2. **Consider modifying the problem in the following way: at any time, you may only search the cell at your current location, or move to a neighboring cell (up/down, left/right). Search or motion each constitute a single 'action'. In this case, the 'best' cell to search by the previous rules may be out of reach, and require travel. One possibility is to simply move to the cell indicated by the previous rules and search it, but this may incur a large cost in terms of required travel. How can you use the belief state and your current location to determine whether to search or move (and**

where to move), and minimize the total number of actions required? Implement and compare the following agents:
Basic Agent 1: Simply travel to the nearest cell chosen by Rule 1, and search it.
Basic Agent 2: Simply travel to the nearest cell chosen by Rule 2, and search it.
Basic Agent 3: At every time, score each cell with (Manhattan distance from current location)/(probability of finding target in that cell); identify the cell with minimal score, travel to it, and search it.
Improved Agent: Your own strategy, that tries to beat the above three agents.

Ans:
Basic Agent 1: Simply travel to the nearest cell chosen by Rule 1, and search it.
Basic Agent 2: Simply travel to the nearest cell chosen by Rule 2, and search it.
For these agents, search strategy remains same as discussed above. Actions are sum of search actions and travelling actions. Travelling actions are computed by calculating the Manhattan distance between the current location agent and next cell to search.

Basic Agent 3: At every time, score each cell with (Manhattan distance from current location)/(probability of finding target in that cell); identify the cell with minimal score, travel to it, and search it.
For this agent, we first compute probability of finding targets for all cells, then we sort the cells with maximum probability. Out of the cells which have maximum probability, we select a cell which is at least distance from the current cell. This cell is selected on the basis of score given by dividing Manhattan distance with the maximum probability. The cell with maximum probability which is at least distance from current cell will have minimum score. In case of tie, the cell is chosen at random. Actions are sum search actions and travelling actions. Travelling actions are computed by calculating the Manhattan distance between the current location agent and next cell to search.

Improved Agent: Your own strategy, that tries to beat the above three agents.
As part of improved agents, we decided to go for one step look ahead approach and we designed three agents based on different one step look ahead strategy(only two are used here for comparison). Actions are sum of search actions and travelling actions. Travelling actions are computed by calculating the Manhattan distance between the current location agent and next cell to search.

Agent 4: For this agent, first we compute probability of finding targets for all cells, then we sort the cells with maximum probability. For each of these cells, we calculate the probabilities assuming that if target is not found the cell. Cells for which future choice have maximum probability of finding the target are selected. For these selected cells, we calculate the score using the Manhattan distance between selected cells and their future choices. The cell whose future choice result in minimum score is selected. Actions are sum search actions and travelling actions. Travelling actions are computed by calculating the Manhattan distance between the current location agent and next cell to search.

Agent 5: For this agent, we first compute probability of finding targets for all cells, then we sort the cells with maximum probability. For each of these cells, we compute the score (as computed for agent 3). Cells which have minimum score are listed and for these cells. we calculate the score using the Manhattan distance between selected cells and corresponding future choices (assuming they return failure). Cell whose future choice result in minimum score is selected. Actions are sum search actions and travelling actions. Travelling actions are computed by calculating the Manhattan distance between the current location agent and next cell to search.

Agent 6: For this agent, first, we take a greedy approach. For each cell, we calculate one step look ahead score; calculated using the Manhattan distance between each cell and corresponding future choices (assuming it return failure). The cell whose future choice result in minimum score is selected. This approach is computation heavy thus it is not included in the comparison. we did test this approach for small landscape size and didn't notice any massive improvements. Actions are sum search actions and travelling actions. Travelling actions are computed by calculating the Manhattan distance between the current location agent and next cell to search.

The comparison of the agents is shown in the figure 7. For comparison we ran 500 iterations for landscape of size 30 i.e. 900 cells. Agent 3 shows a massive improvement from Agent 1 and Agent 2. Agent 4 does not outperform Agent 3, this may be because Agent 4's primary selection is just based on maximizing probabilities. Distance is only considered for future choices while calculating score for them.

Agent 5 slightly outperforms Agent 3 and is the best among agents. Agent 3 and Agent 5 also outperforms Agent 1 when target is located in the caves. As we move from Agent 1 to Agent 2 to Agent 3 to Agent 5, we are moving from taking more random decision to less random decisions and the decisions that is more aligned to our goal.

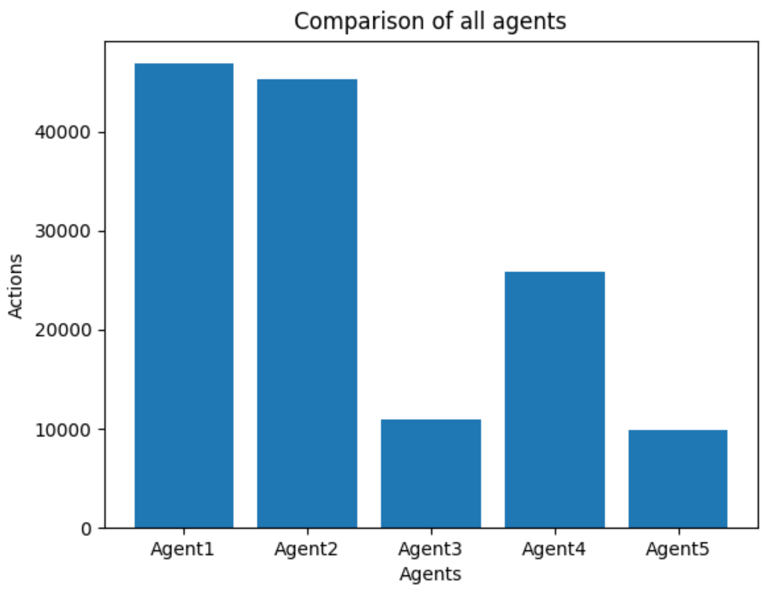

Figure 7: Comparison of all Agents

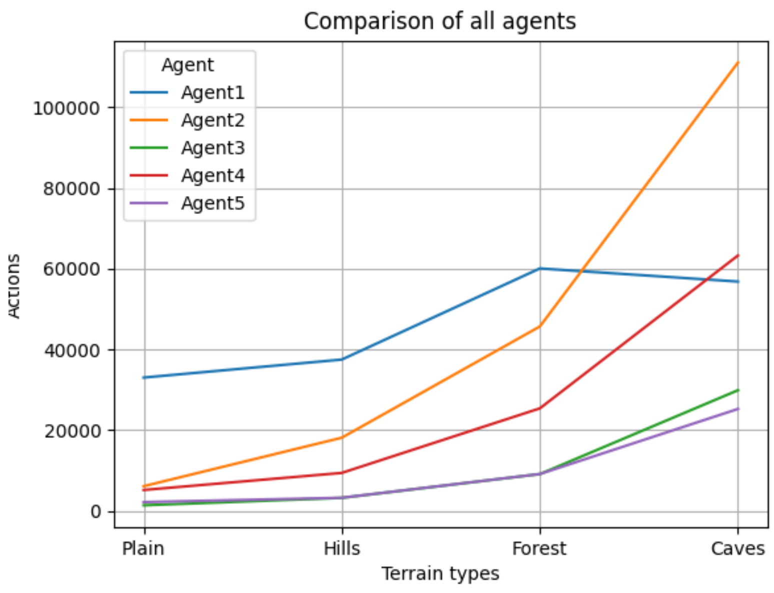

Figure 8: Bar Graph Comparison of All Agents

Part 2: A Moving Target

Introduction

In this section, the target is no longer stationary, and can move between neighboring cells. Each time we perform a search, and if we fail to find the target, the target will move to a neighboring cell. Every time we search a cell, we pass two pieces, first, whether or not our search was successful; and if the search was unsuccessful, we pass the Manhattan distance 5 from the current location.

Implementation of Moving Target and New Observations

As per this problem, at each search step which can result in failure, we move the target to one of its neighbors (we have included diagonal neighbors also) which is selected randomly. Also, we get to know whether the target is with in 5 Manhattan distance range of current location.
This observation divides search space into two parts, cells which are within 5 Manhattan distance of current location and cells which are outside 5 Manhattan distance of current location. And which part is relevant depends upon in which part the target exists. Since target is moving and relevant cells in the search space will also keep changing, there is no point in keeping track of probabilities for all previous observations.
At each step, if target lies within 5 Manhattan distance of current location, for all cells outside 5 Manhattan distance of current location, the probability of locating cells and thus finding cell becomes zero and visa-versa.
if target lies within 5 Manhattan distance of current location, all cells within 5 Manhattan distance of current location will have the updated probability of locating and thus finding the target based on the observation that target is not found in current location but is with in search space and the new cell is selected accordingly from these cells. On the other hand, if the target lies outside 5 Manhattan distance of current location, all cells outside the 5 Manhattan distance are equally likely to hold target and cell is selected accordingly from them.

Analysis

1. Factor this additional information into your search strategy in the case that you are able to search any point at any time. How does this additional information affect the number of searches needed to find the target? Compare to Rule 1 and Rule 2.

   Ans: We ran 100 iteration for a landscape of size 50x50 i.e 2500 cells, and as shown in Figure 9, Agent 2 outperforms Agent 1 in finding the moving target over here as well. On average, Agent 1 takes more search actions than Agent 2. The additional reduces the search space in which we have to locate the moving target.

2. Factor this additional information into your search strategy in the case that you are only able to move to your immediate neighbors in each time step. How does this additional information affect the number of actions (searches + motion) needed to find the target? Compare to Basic Agent 3 and your improved agent.

   Ans: We ran 100 iterations for a landscape of size 30x30 i.e 900 cells, and we compare all the agents that were compared for a stationary target. As shown in Figure 10, Agent 3 outperforms all other agents, including Agent 5, which was obtained as the best agent for a stationary target. We believe that the reason for the bad performance is the fact that due to moving target and new observations, relevant cells or cells which have high probability of having the target are continuously changing; thus one step look ahead probabilities or score from current situation does not provide very relevant information.
   For moving target, the improved agent will be the one with better scoring techniques or one step look ahead technique based on checking presence of target in 5 Manhattan distance zone for any cells

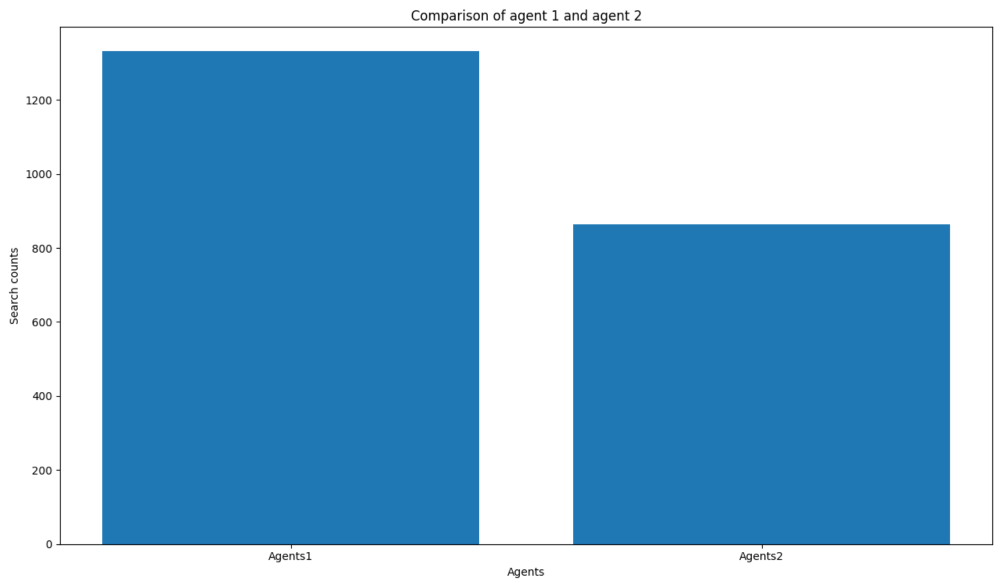

Figure 9: Comparison of all Agents

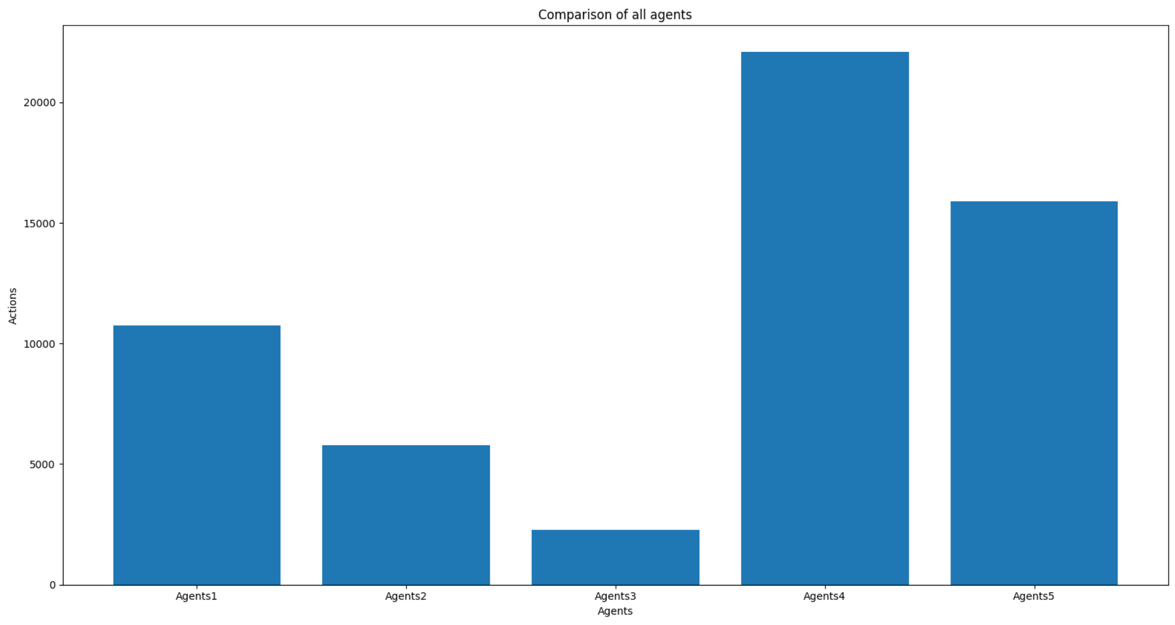

Figure 10: Comparison of all Agents

Code Overview

To implement this project, we created two Python classes, *ProbabilisticHunting.py*, which implements the landscape and the search agents, and *ProbabilisticAnalysis.py*, which compares agents and generates relevant graphs.

ProbabilisticHunting.py - This python script contains class with same name which implements landscape, all agents for stationary and moving targets, and all other functions required functions.

ProbabilisticAnalysis.py - This python script contains class with same name which implements functions for performance analysis for the agents in various scenario. Below are the brief about the functions in this class:

- disp_data1 - Normal plot graph is used. Average number of search count/actions are found for each terrain type and for each agent.
- disp_data2 - Bar graph plot is used. Creating axis on figure, average no of search counts/actions per agent.
- samemap - In this case, Agent 1 and Agent 2 to find target on same map with multiple targets.
- multiple map - In this case, both the landscape and target location change after each iteration
- compareall - Comparing all agents to find not moving target on multiple maps, changing target on each iteration
- compareallformovingtarget - Comparing all agents to find moving target on multiple maps, target is set to original position for each agent
- comparetwoagnetsformovingtarget - Comparing search counts for agent 1 and agent 2 to find moving target on multiple maps, target is set to original position for each agent

Responsibilities

GitHub Repository : Intro to AI - Project 3

For this project, all the team members shared equal responsibilities, but each one of us took lead in the below mentioned parts of the project:

Kaustubh N Jadhav (knj25) - Algorithms and Readability of Code
Pranav Shivkumar (ps1029) - Report, Data Visualization and Representation
Sanyam Jain (sj770) - Coding, Algorithms and Answers for the report.