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:

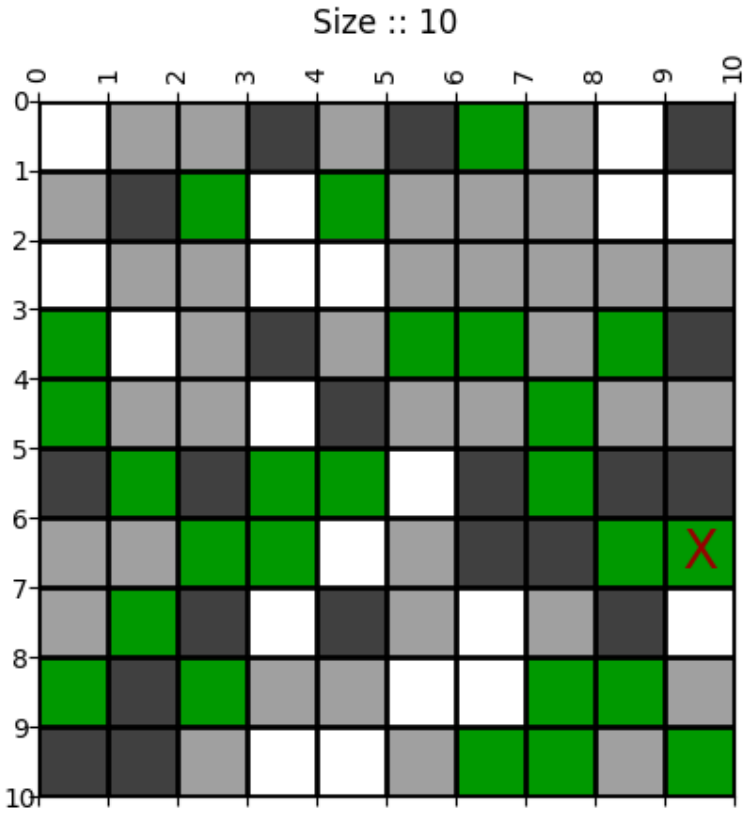

Figure 1: Sample Landscape of size 10x10

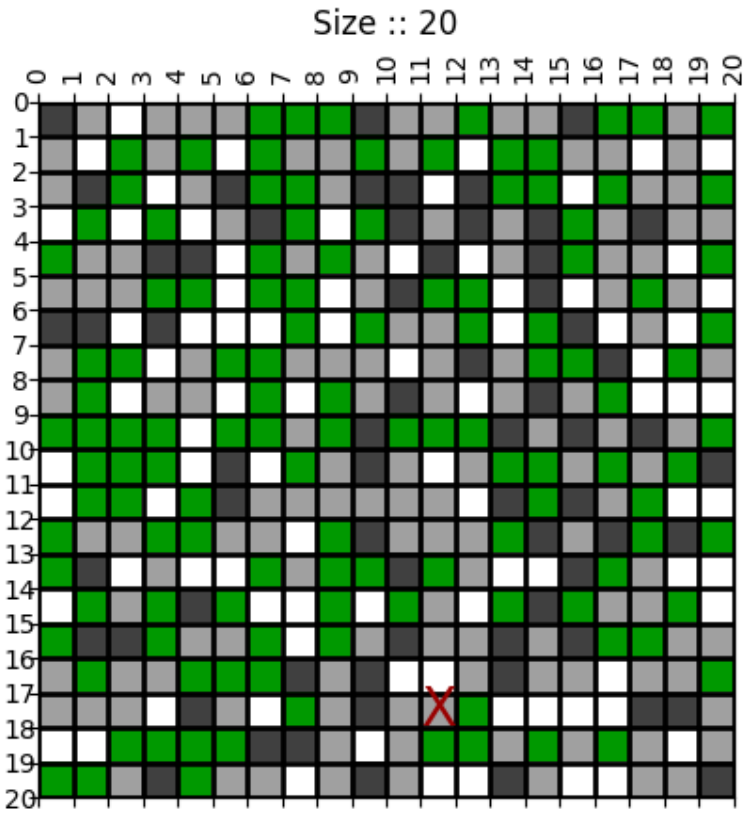

Figure 2: Sample Landscape of size 20x20

In the initial state, the target can be located anywhere in the landscape since we don't have any knowledge regarding the cells, 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

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

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

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

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

   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:

   Does that hold across multiple maps?
   Ans:

   In order to compare the performance of both rules, we plotted a graph between the terrain type vs the number of searches. This plot is shown in the graph below:

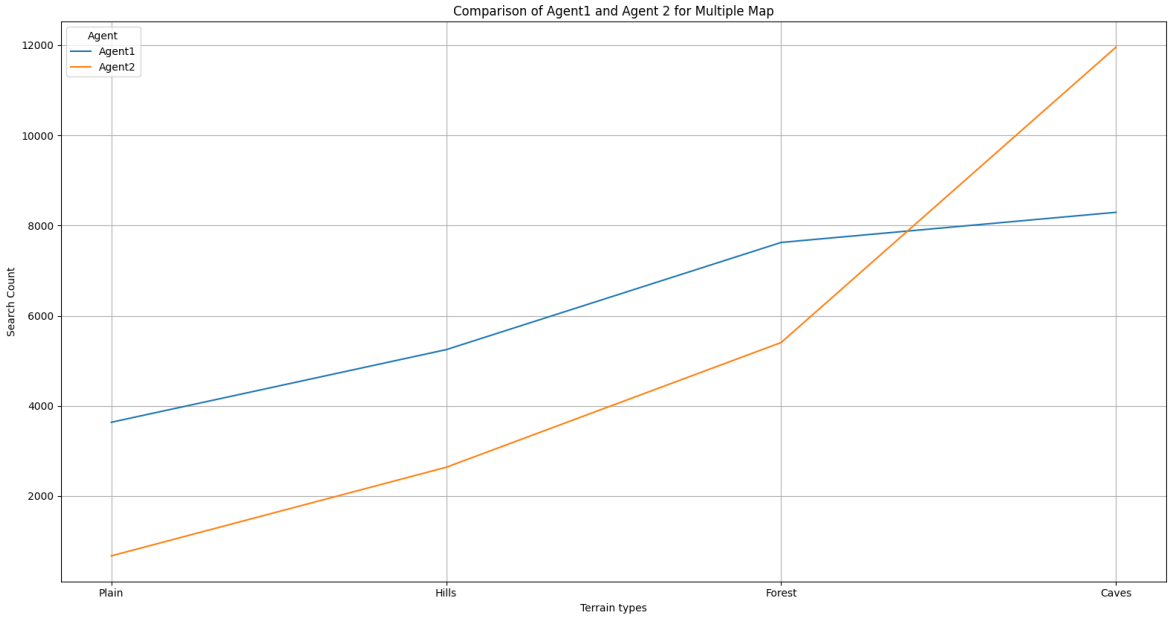

Figure 3: Search Count Comparison of Rules 1 and 2 for Terrain Types

From this graph, we can see that the number of searches to find the target using Rule 1 is more than the number of searches taken by Rule 2, for the flat, hilly and forested terrain types. However, the number of searches by Rule 2 is more than that for Rule 1 in the caves terrain.

Rule 2 shows better performance in terms of the searches compared to Rule 1 in most of the terrains, but Rule 1 shows a better performance than Rule 2 for a region with caves because the false negative rate for the caves region is the greatest among all the terrain types, so it is more difficult to search for the target in those cells. Since Rule 1 considers the probability that the cell being searched has the target, it doesn't take the false negative rates into account, so it takes less number of searches than Rule 2, which takes the false negative rate into account.

The below graph shows the scenario for multiple maps:

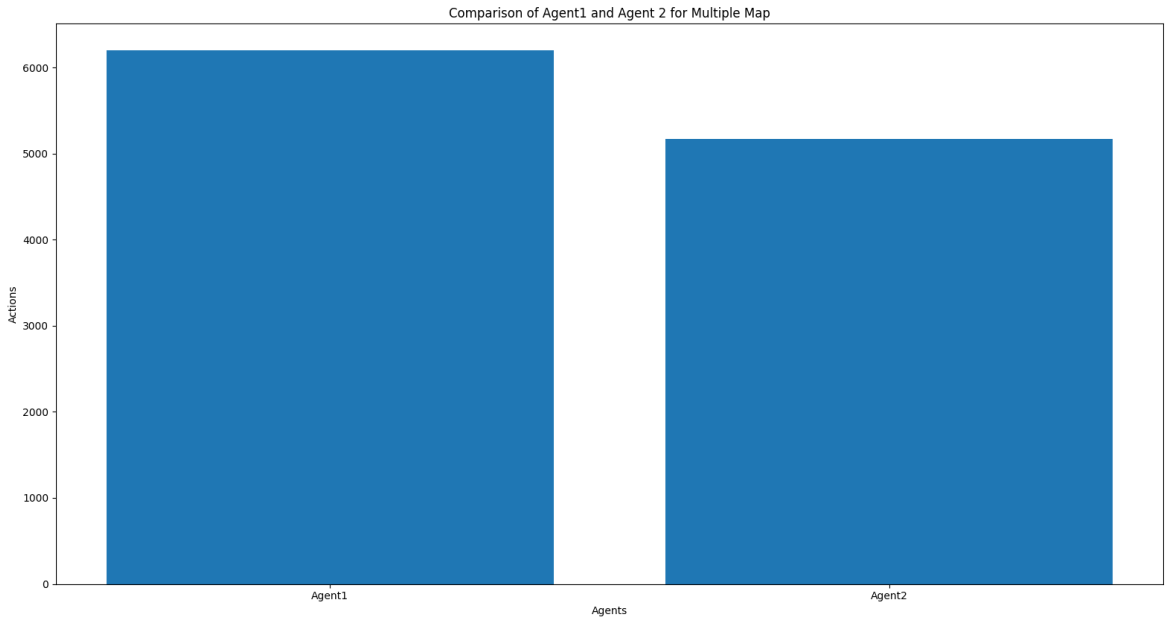

Figure 4: Comparison of Rules for Multiple Map

In the case of multiple maps, the location of the target and the landscape map is modified after each iteration i.e. after running with both the rules. As seen from the bar graph, Rule 1 shows a higher number of searches compared to Rule 2, which shows that the above idea holds true for multiple maps as well.

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: The belief state shows what we know about the landscape as the search progresses. AS we search the cells in the landscape, based on any of the rules, the probabilities in the belief state get updated until the target is found. As far as the basic agents 1 and 2 are considered, while the target will be found eventually, there will be a large cost incurred due to the travelling between the cells in the landscape. For Basic Agent 3, we decide how to proceed with the search using a score for each cell, defined based on the Manhattan distance from the current location. The cell with the minimum score is chosen as the next cell to search, which eliminates the travel cost to an extent.

   The comparison of the agents is shown in the plots below:

Among all the agents, the improved agent (Agent 5) shows the best performance, in terns

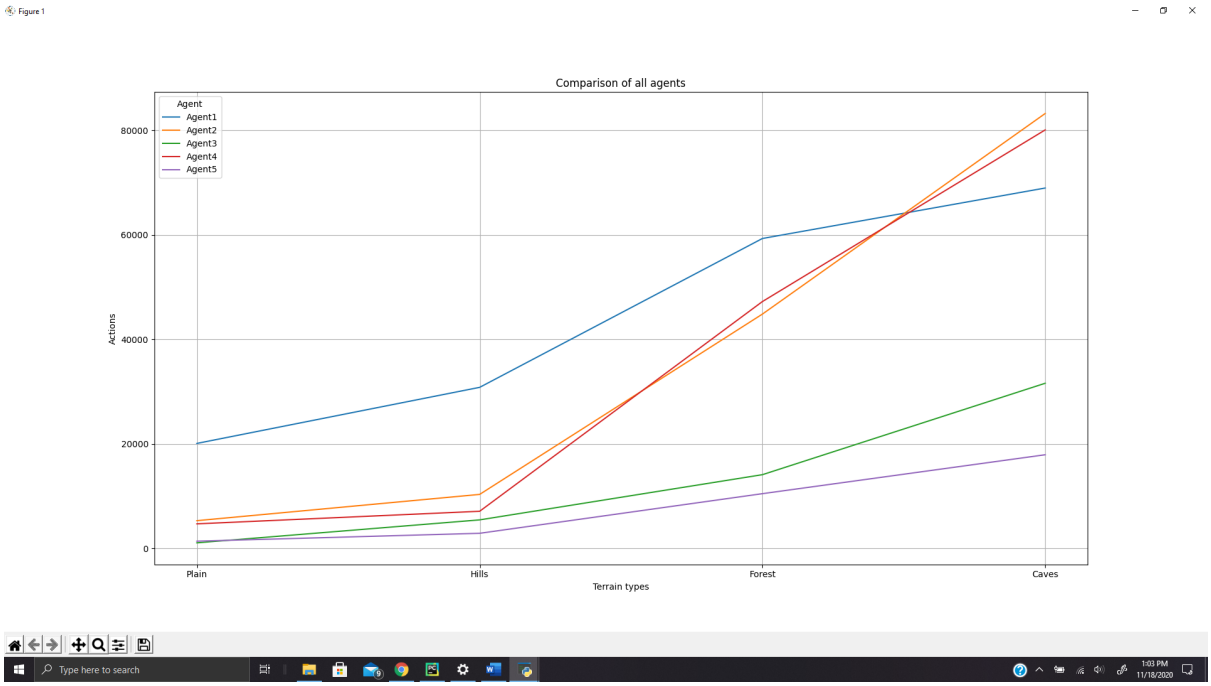

Figure 5: Comparison of all Agents

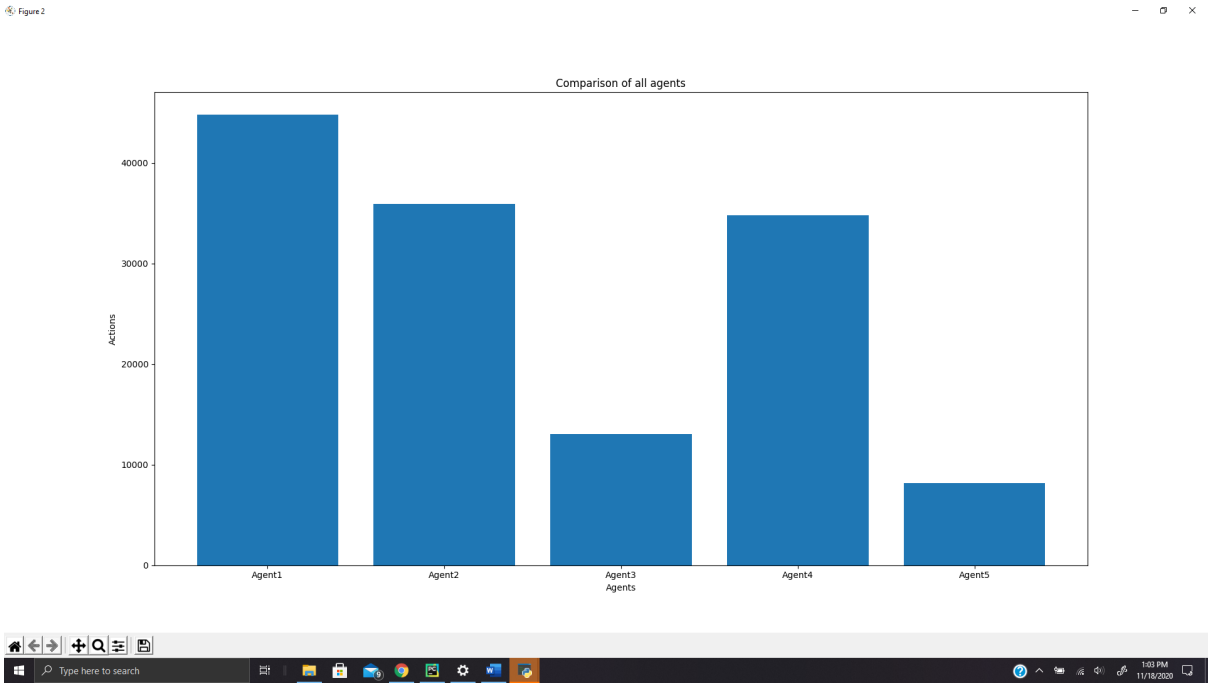

Figure 6: Bar Graph Comparison of All Agents

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.