CS 440

PROFESSOR WES COWAN

Probabilistic Search (and Destroy)

APRIL 10, 2021

NABHANYA NEB (NN291)

ARPITA RAY (AR1516)

I. Introduction

Our project models knowledge and belief about a system probabilistically. We use this belief to determine future actions. We have a landscape represented by a map of cells and they all have a terrain type. They can be flat, hilly, forested, or a maze of caves, and they vary in how difficult they are to search. So, they all have different false negative values: 0.1 for flat, 0.3 for hilly, 0.7 for forested, and 0.9 for a maze of caves. Among these cells, one of them contains a hidden target. The probability of the target being in a cell begins as $\frac{1}{\text{dim*dim}}$, with dim being the dimension of the board. We call this the belief of a cell. This probability changes as we start searching cells. Our objective is to find the goal in as few searches as possible. We want to minimize performance, which we define as total distance traveled + number of searches.

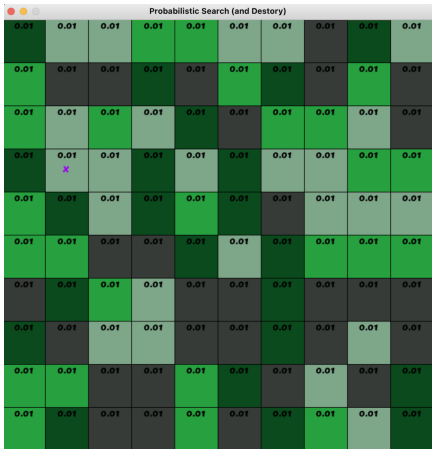

Fig. 1: Example of initial board before agent is placed

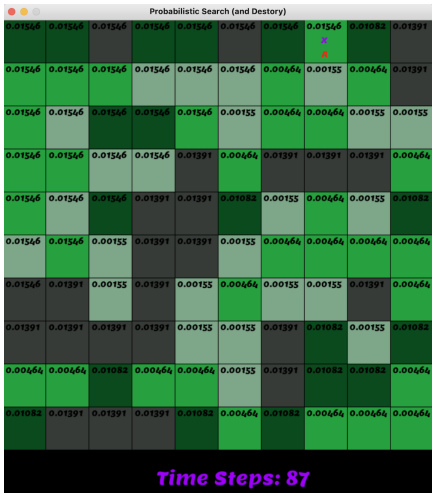

Fig. 2: Example of final board after target is found

II. Representation of the States

To begin, we define a state object. This object is used to make up the game board. Each cell is a state, which is initialized with a row, column, and value attribute. The row and column represent its location on the board, and the value is used to store whether it is flat, hilly, forested, or a maze of caves. It also stores the corresponding false negative value associated with the terrain type, the belief value, and whether or not it is the target. Lastly, we added a function to return a list of all the neighboring states. A neighboring state can either be above, below, to the left of, or to the right of the given state.

III. Representation of the Game

We next define a game object. This object is used to run the fundamentals of the game. The game holds the dimension, total number of searches anad steps taken, and a 2D array of states which is referred to as the grid. We also have a function that generates the environment for the board here. We go through the board and randomly assign one of the four terrain values to each State. Based on the terrain, we also assign the corresponding false negative value to the State.

IV. Problem 1

Given observations up to time t and a failure searching Cell_j, we used Bayes' theorem to efficiently update the belief state. So, this means we want to calculate: P(Target in $\text{Cell}_i|\text{Observations}_t \wedge \text{Failure in Cell}_j$). We call this probability the posterior probability.

First, we want to find this probability, given that $i = j$. In other words, we are finding the probability that the target is in a cell after there was a failure in the same cell. We will define X_i as the existence of target at Cell_i and Y_i as the search result of this time at Cell_j. Now, we can rewrite our formula as $\text{P}(X_i = 1|Y_i = 0)$ with 1 meaning a success and 0 meaning a failure. We will now use Bayes' Theorem. The likelihood in our problem is $\text{P}(Y_i = 0|X_i = 1)$ and the prior is $\text{P}(X_i = 1)$. We use these values and $\text{P}(Y_i = 0)$ to get the following formula: $\frac{\text{P}(Y_i = 0|X_i = 1)*\text{P}(X_i = 1)}{\text{P}(Y_i = 1)}$. Next, we calculate these individual probabilities.

The likelihood, P($Y_i = 0|X_i = 1$), ends up being the false negative value of the terrain type so it could be 0.1, 0.3, 0.7, or 0.9. The prior, P($X_i = 1$), is just the current belief we have so far for Cell$_i$. The last value, P($Y_i = 0$), we can compute using marginalization and it ends up being equal to: P($Y_i = 0|X_i = 0$)*P($X_i = 0$)+P($Y_i = 0|X_i = 1$)*P($X_i = 1$). We can now look into each of these probabilities. First, P($Y_i = 0|X_i = 0$) will just always be equal to 1. P($X_i = 0$) is 1 minus the current belief we have for Cell$_i$. P($Y_i = 0|X_i = 1$), as stated above, is the false negative value of the terrain for Cell$_i$. Lastly, P($X_i = 1$) is the current belief we have for Cell$_i$. When we combine all of this, we end up with:

$$\frac{\text{Prior Belief Cell}_i\text{*False Negative Rate Cell}_i}{(1-\text{Prior Belief Cell}_i)+(\text{False Negative Rate Cell}_i\text{*Prior Belief Cell}_i)}$$

Next, we want to find this probability, given that $i \neq j$. Similar to how we used Bayes' Theorem above, we find that this probability is equal to $\frac{\text{P}(Y_i = 0|X_j = 1)\text{*P}(X_j = 1)}{\text{P}(Y_i = 1)}$. The denominator of this probability is the same as the previous calculation so we can reuse that. However, the numerator is different so we will analyze it. For P($Y_i = 0|X_j = 1$), we know that means that if the target is in Cell$_j$, we want to know the probability of checking Cell$_i$ and finding nothing, but this must always be true so this value equates to 1. We know that P($X_j = 1$) is just the prior belief of Cell$_j$. When we combine all of this, we end up with:

$$\frac{\text{Prior Belief Cell}_j}{(1-\text{Prior Belief Cell}_i)+(\text{False Negative Rate Cell}_i\text{*Prior Belief Cell}_i)}$$

This covers all of the cases for calculating P(Target in Cell$_i$|Observations$_t$∧Failure in Cell$_j$) in our project.

V. Problem 2

Next, given the observations up to time t, the belief state captures the current probability the target is in a given cell. We want to compute the probability that the target will be found in Cell$_i$ if it is searched. So, this means we want to calculate: P(Target found in Cell$_i$|Observations$_t$).

This probability uses the current belief, which we define as the value we calculated in problem 1, and factors in the false negative value of the terrain of Cell$_i$. We will multiply this current belief by 1 minus the false negative to get our final formula: Prior Belief Cell$_i$ * (1-False Negative Rate Cell$_i$).

This is how we calculate P(Target found in Cell$_i$|Observations$_t$) in our project.

VI. Problem 3

Basic agent 1 and 2 use the probabilities from problem 1 and 2. They operate very similarly and only differ in which probability they use.

For basic agent 1, we start by generating the board and initializing the probabilities to $\frac{1}{\text{dim*dim}}$. Next, we randomly put the agent in a cell and search it. If we find the target, we end the game and return the performance value, which is the number of steps taken + the number of searches. If we don't find the target, we call the function update_info to update the probabilities, the way we discussed in problem 1, for the whole board. Now, we call function highest_state which picks the next spot for the agent to travel to. It does so by finding the spot with the highest probability. If multiple spots have the same probability, it picks the closest one to go to and if there are multiple with the same distance and probability, we pick randomly between them. After this, in function get_to_next, we move the agent to this new spot and keep track of the distance traveled. We search this cell and if we fail to find the target, we repeat this process until we do.

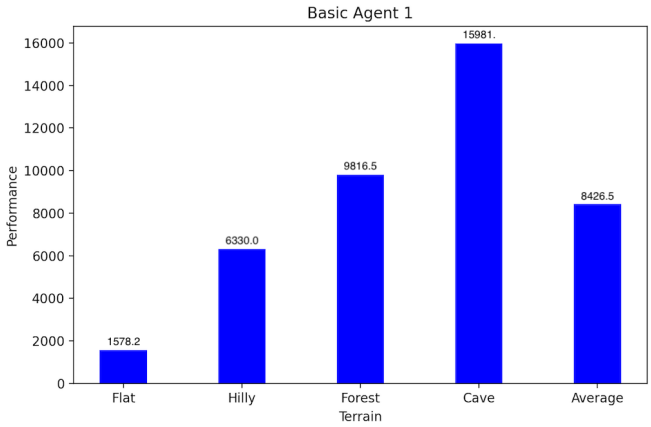

Fig. 3: Plot of performance grouped by terrain type for basic agent 1

For basic agent 2, we do exactly the same thing as we do in agent 1 with one difference, we use

a different belief calculation. Instead of using the probability from problem 1, we use the calculation from problem 2 to pick the next spot to go to.

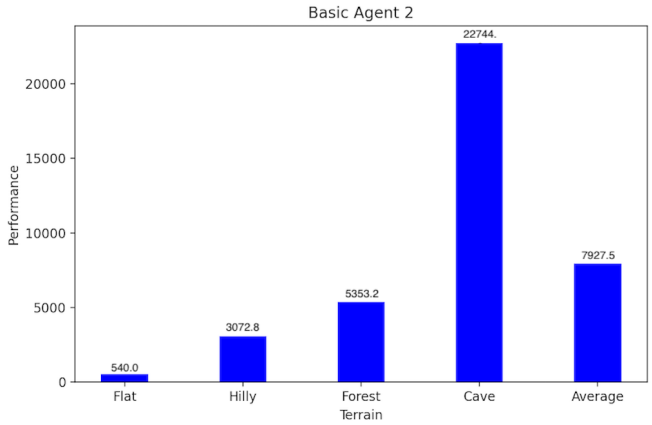

Fig. 4: Plot of performance grouped by terrain type for basic agent 2

To analyze the two agents, we generate maps with random target locations and random initial agent locations, each time. Instead of generating 10 maps, we generated 20, with the target in each terrain type 5 times, and found the average of them. We also calculated the average of all terrains and displayed this as a column in the plot. These plots are shown above.

On average, agent 2 performs better than agent 1 for all terrain types except for caves. Caves have the highest false negative rate, which leads to low probability values. Since basic agent 2 uses the probability that the target will be found, it tends to avoid caves. The overall average, including terrain types, was 8426.5 for basic agent 1 and 7927.5 for basic agent 2. It makes sense that agent 1 outperforms agent 2 on caves because these have the highest false negative value. If we disregarded caves, our agent 2 would perform even better than agent 1. Despite this, agent 2 still performs better overall.

VII. Problem 4: S.A.L.S.A

A new agent, the advanced agent, is designed to improve the basic agents and beats both of them. Our clever acronym for this agent agent is S.A.L.S.A, which stands for Standardize and Look Steps Ahead. This will make more sense with the explanation that follows.

This agent looks 1 step ahead of the basic agent. As long as we have not found the target, we predict what the board will look like one step ahead and calculate all of the probabilities for this temporary board, the same way we did in basic agent 2. With this new board of predictions that is one step ahead, we call the function highest_state and calculate which spot we would go to next if this was the real board.

However, we made another change for this agent. We pick which spot to go to next differently than basic agent 2. Instead of just picking the highest probability, we find the standard deviation of all of the probabilities on the board every time we search a cell. Then, we add any cell that is within 1 standard deviation of the highest probability on the board and add it to a list. We now pick the spot with the lowest Manhattan distance away from the current spot to go to next.

Now, we have two potential next spots to pick from. The first one comes from predicting the future and also using the standard deviation to decide. The next option comes from not using the future prediction and just using the standard deviation calculation. From these two, we pick the closer spot and go there. This is how our advanced agent works.

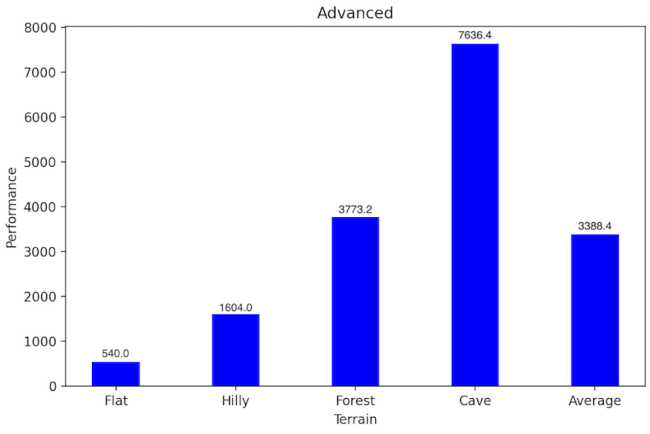

Fig. 5: Plot of performance grouped by terrain type for advanced agent

Efficiency wise, the advanced agent beats both basic agents, as shown by the plots. Its average is 3388.4. This is significantly better than both basic agents.

Given unlimited resources, space, and time, we would improve our advanced agent by looking more than one step ahead. We could go through and predict the entire board ahead of time, every time we search a cell, and this would give us the most accurate prediction results. But, it would take a lot of time and space to do this with the resources we currently have.

VIII. Bonus: A Moving Target

For the bonus, the target is no longer stationary and it can move between its neighboring cells: up, down, left, and right, with an equal chance of going in all directions. Every time you search a cell and fail to find the target, the target moves to one of its neighbors and the agent is told whether the target is within Manhattan distance of 5 from its current cell. We implemented this for both the basic agents, and the advanced agent.

For basic agent 1, we modified how the agent chose the next cell to visit. It iterates through the board and creates two sets of cells. One set contains everything that is less than or equal to a Manhattan distance of 5, and the other set contains everything that is greater than a Manhattan distance of 5. Instead of always going to the cell with the highest probability on the entire map of containing the target, it goes to the cell with the highest probability in one of these sets, depending on the information it receives for the target's current location.

Basic agent 2 does the same thing as basic agent 1, but instead of using the probability of the cell containing the target, it uses the probability of it actually finding the target in that cell, based on the false negative rates described above.

The advanced agent uses the same method as above, but this time it splits the set of all cells with a probability greater than or equal to the max probability minus one standard deviation into two subsets: one with the cells within Manhattan distance 5 and one with the cells greater than Manhattan distance 5. It chooses the next state according to the information it receives about the target's current location.

The bonus version of the advanced agent performs very well. Its average performance for 15 runs is 7639.5. The two basic agents perform similarly to each other. Basic agent 1 has an average of 11328.0. Basic agent 2 has an average of 10435.5. These averages are near each other because when the target is constantly moving around, the probability of finding it versus containing it does not impact the overall performance by much.

Acknowledgements

Nabhanya Neb (nn291):
We did both the code and the report together using Zoom. Because of this, we did not have to split anything up and we both contributed equally while working on call. We alternated who was coding/typing to make it fair.

Arpita Ray (ar1516):
The work was evenly divided because we did all of the code on screen-share. We took turns on who would be the person typing each time. For the report, we used Overleaf and did the write up the same way. We worked together on it and alternated who wrote.

Honor Statement:
We have read and abided by the rules of the project, we have not used anyone else's work for the project, and we did all of the work by ourselves.