Intro to AI Project 2 Write Up

March 20, 2021

On my honor, I have neither received nor given any unauthorized assistance on this assignment. All of our work and ideas are our own. We did not use any other external sources.
- Michael Zhang, mbz27, Section 1

On my honor, I have neither received nor given any unauthorized assistance on this assignment. All of our work and ideas are our own. We did not use any other external sources.
- Prasanth Balaji, pmb162, Section 3

Group members: Michael Zhang, Prasanth Balaji

Representation

The Minesweeper board is represented and stored as a 2D array. We have a Cell class that stores information regarding an individual cell such as its status (mine, safe, and covered), position, whether it is a mine or not, and information regarding its surrounding cells. This information includes the number of surrounding mines, identified surrounding safe cells, identified surrounding mine cells, and the number of hidden cells. We use the Knowledge class to represent the inferred relationship between cells. The Knowledge class has two attributes which is an ArrayList of cells and the number of mines. Both of these attributes are used to represent an equation. The equation basically stores pointers of the cell, and we can use these equations to gather inferences by looking for contradictions.

Inference

When we collect a clue, we can have 3 inferences:

One inference is the case in which a cell's total number of mines (the clue) minus the number of revealed mines equals the number of hidden neighbors. If this is the case, then we know that all the hidden neighbors are mines.

Another inference is the case in which a cell's total number of surrounding tiles minus the total number of mines (the clue) minus the number of revealed safe neighbors is the number of hidden neighbors. If this is the case, then we know that all the hidden neighbors are safe.

The final inference is that given a cell, the total number of mines contained in the surrounding hidden neighboring cells is the total number of mines (the clue) minus the number of identified mines.

In order to infer if a cell is a mine or not, we use proof by contradiction. We assume both cases in which the cell is a mine or not and check them using our knowledge base. If the knowledge space contains any illogical equations such as the number of mines being a negative number or that the total number of mines exceeds the amount of surrounding cells, then we can deduce that the specific assumption was false. Otherwise, the assumption would be true. We also compare both assumptions in the following way to make sure everything is logical:

If both assumptions are false, then the knowledge base is inconsistent and there is something wrong with it.

If both assumptions are true, then we can't be certain whether the given cell is a mine or not.

If the assumption of the cell being a mine is false and the assumption of the cell being a safe cell is true, then we can say that the cell is not a mine.

If the assumption of the cell being a mine is true and the assumption of the cell being a safe cell is false, then we can say that the cell is a mine.

We use the Knowledge class to store these inferences and add it to the global Knowledge base ArrayList. We will iterate through a list of hidden cells until we can infer something about one of the hidden cells, then we will reiterate through the entire list again and we repeat this process until there is nothing left to infer.

Decisions

When we first start the Minesweeper game we will pick random cells until we find one a cell that is safe (assuming there is no global information). We then look at all of the hidden cells and infer any information about the hidden cells given the random safe cell we picked. If we can infer that a cell is either safe or a bomb, then we repeat the process of iterating through the hidden cells with the goal of trying to infer more information until we can't infer anything else about the hidden cells. After this, we continue the process of picking random cells.

Performance

Play by Play Progress:

```
? ? ? ?
? ? ? ?
? ? ? ?
? ? ? ?
```

Picking random cell

At this moment the algorithm is picking random cells until it can infer something about any of the hidden cells.

It might be beneficial to select the cells in a specific pattern (for example picking diagonal cells of the board) as it will possibly give us better clues to infer from.

```
? ? x ?
? 3 4 ?
? ? ? ?
? ? ? ?

? ? x ?
? 3 4 ?
? ? x ?
? ? ? ?

? ? x ?
? 3 4 x
? ? x ?
? ? ? ?
```

```
? ? x ?
1 3 4 x
? ? x ?
? ? ? x
```

```
? ? x ?
1 3 4 x
x ? x ?
? ? ? x
```

```
? ? x ?
1 3 4 x
x ? x x
? ? ? x
```

We looked through the boards one by one, and it seems as though there is nothing to infer about from the randomly selected cells.

Picked safe

The algorithm has picked enough random cells to infer something about the hidden cells.

```
0 1 x ?
1 3 4 x
x ? x x
? ? ? x
```

It has assumed that the cell (0, 0) and cell (0, 1) are safe cells based on the 3 and 4. It concluded that cell (0, 1) is safe based on the 4 and it concluded that cell (0, 0) is safe based on the 3.

```
0 1 x 2
1 3 4 x
x ? x x
```

```
? ? ? x

0 1 x 2
1 3 4 x
x 4 x x
? ? ? x
```

It concluded that all of the non hidden cells around the 4 are safe. We agreed on all of these inferences the algorithm made, and it doesn't seem as there is anything else to infer.

Picking random cell
It randomly picked cell (3, 0) and (3, 1) and concluded that cell (3, 2) is a mine.

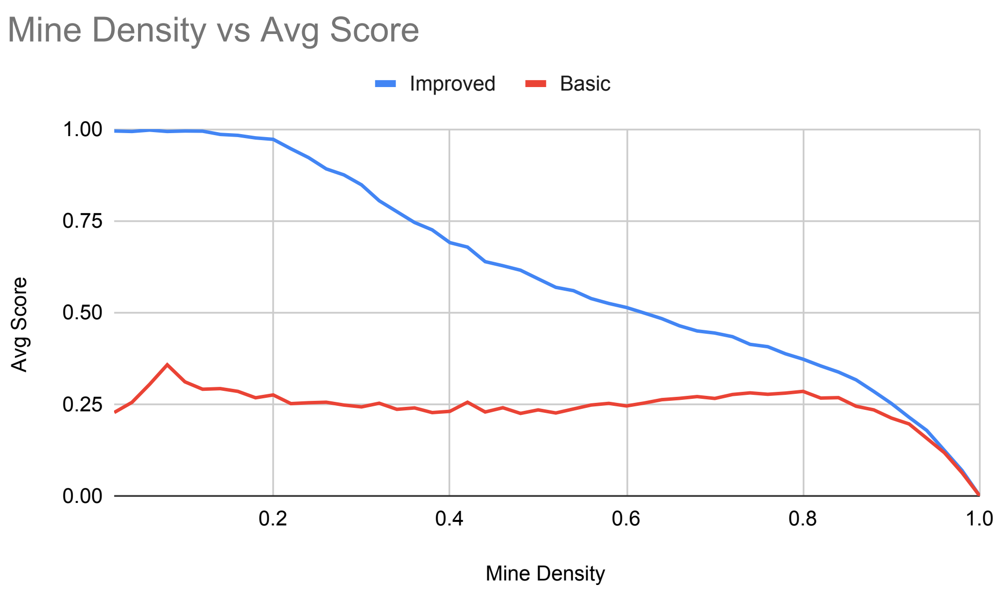
```
0 1 x 2
1 3 4 x
x 4 x x
x 4 ? x
```

No more cells to pick from
The minesweeper board is complete and solved.

```
0 1 x 2
1 3 4 x
x 4 x x
x 4 x x
```

The algorithm made logical decisions that we would agree with.

Performance: Improved vs Basic



It makes sense that the basic agent performs poorly in the beginning as there was less information inferred regarding how the board looked like, as the basic agent only accounted for hidden cells that were only composed of safe cells or mine cells. On the other hand, the improved agent accounted for hidden cells that were both safe and mine cells due to the extra inference abilities it had as opposed to the basic agent. However, as the mine density approached 100 percent, it was more likely that both would pick a cell that is a mine since there was much more mines to identify. This resulted in both to perform poorly. Our implementation of the improved agent was always superior to the basic agent. However, they both started to converge around a mine density of 0.8, so it is fair to say this is when the minesweeper becomes ”hard”.

Efficiency

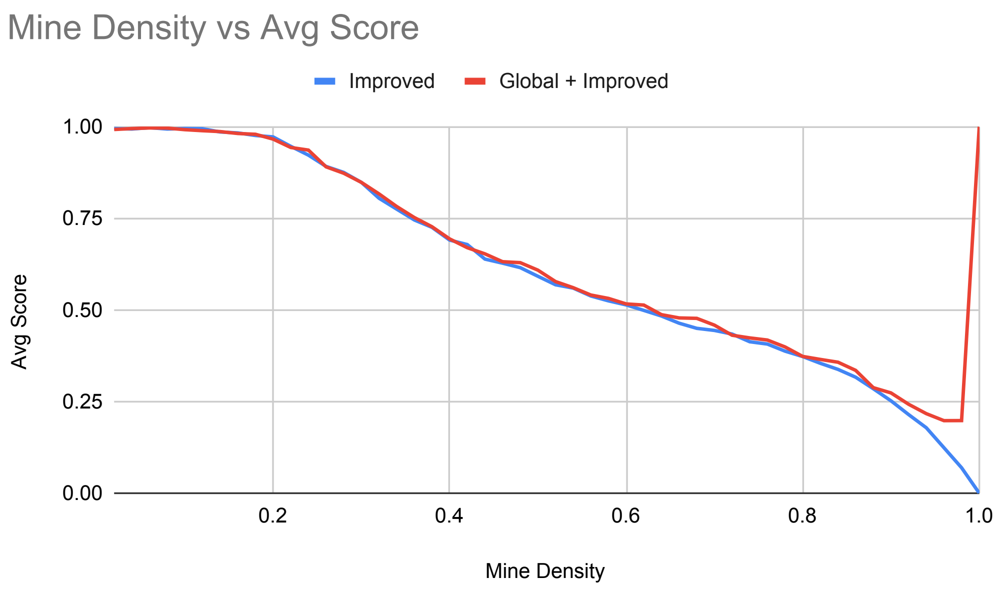
The time constraint would be the fact that we would have to iterate through the entire knowledge base every time we want to infer something and we would have to iterate through the entire list of the hidden cells to check if there is anything to infer about the hidden cells. Towards the end of the program, the knowledge base is greater than or equal to the size of the board which would be a space constraint. Another obvious space constraint is the storage for the minesweeper board.

A problem specific constraint is the fact that we have iterate through the entire knowledge base, as there is no way to bypass this. However, we can try improve the number of times we query the knowledge base so that the efficiency can slightly be improved. To do this, we come across an implementation specific constraint which is how we pick a specific cell to infer from. If we had an intelligent way to pick cells to infer from, we would be more efficient.

Another implementation specific constraint is regarding our Knowledge class. Currently we are using ArrayLists to represent the equations which results in an access time of O(n). We could use HashSets which will potentially give us an access time of O(1).

Global Information

At the beginning of the game, we can add an equation describing the fact that all tiles, which are hidden, is equal to the total number of mines on the board. This will be implemented by using the Knowledge class which stores the tiles and the total number of mines in those tiles. Theoretically, given the new clues we find throughout the game, we can use subset reduction to infer more clues based on this large equation.



The Global + Improved agent is definitely better towards the end as it knows how many mines there is to begin with, which results in an edge case when the mine density is 100 percent the Global + Improved agent will have a perfect score of a 100 while the Improved agent has a score of 0.

Documentation

How to run the project:

Please enter the arguments in this order, [dimension of the board] [num of mines] [type of agent] [global agent]

- Type of agent: 0 representing the basic agent, 1 representing the improved agent

- Global agent: 0 disabling the feature, 1 enabling the feature