Assignment 2

Farhat Alam, Karthik Paka, Matan Shachnai, Andrea Wu
198:520 Intro to Artificial Intelligence
Rutgers University
March 13, 2019

#3

Representation: The board is represented as a grid (2D array) where each cell contains variables to indicate cell value (how many bombs are bordering the given cell) and whether the cell is a bomb. The information that clue cells reveal is represented by logical statements in the form of a list of cell coordinates and the total number of mines distributed across these cells. For example, if the cell with coordinates (0,0) reveals a 2, the resulting information would be represented as [(0,1), (1,0), (1,1)] = 2. For our implementation, this representation is used for a list of cells whose value is known either by being clicked or through deduction and a list of longer logical statements that keep track of the information gained by the agent.

Inference: When a cell is clicked, the agent processes the information differently depending on what is revealed. If a cell's value is revealed to be 0, then all of its neighbors are clear and each neighbor is removed from the priority queue that is used to determine which cell is clicked next, added to the list of known cells if not already in the list, and used to update the list of logical statements. If a neighboring cell has a nonzero value, then a logical statement is created from its neighbors' coordinates and its value, updated based on the list of known cells, and added to the list of logical statements if it is not already present in the list. The agent also uses the same logic for any neighboring zeroes to add as much information to the knowledge base as possible.

If a cell's value is not 0, then it is added to the list of known cells and used to update the list of logical statements. If a cell's value is revealed to be equal to the number of neighbors it has, then all of its neighbors are mines and each neighbor is removed from the priority queue that is used to determine which cell is clicked next, added to the list of known cells if not already in the list, and used to update the list of logical statements. If a cell has any other value, then a logical statement is created from its neighbors' coordinates and the value that is revealed, updated based on the list of known cells, and added to the list of logical statements if it is not already present in the list.

Before any addition to the list of known cells, the list of logical statements is updated using the other statements in the list to consolidate information. After any change to the list of logical statements, the list of logical statements is checked for any deductions that allow a cell to be added to the list of known cells. If a cell is added to the list of known cells and has a mine, then it is removed from the priority queue that determines which cell is clicked next. Otherwise, if the cell is clear, it is placed at the top of the priority queue.

Our program deduces everything it can from a given clicked cell, using the information from a clicked cell to update the list of known cells and the list of logical statements. In addition to processing this information, the list of logical statements is also consolidated and used to further update the list of known cells, using the information gathered to make additional deductions.

Decisions: The mechanism by which our program determines which cell to click next is a priority queue. Originally, the priority queue is filled with all of the cell coordinates, randomly ordered. When it is deduced that a cell is a mine, its coordinates are removed from the priority queue. When it is deduced that a cell is clear, its coordinates are moved to the front of the priority queue to be visited next. If it has not been determined whether a cell has a mine or not, we assume that using randomness to decide which cell to click next will allow the agent to gain more information about the board and update its knowledge base to improve future decision making. This also provides more equal opportunity for revealing new information about the board. However, it is also important to consider that at times, randomness in choosing which cell to consider next can decrease the performance of the program.

Performance: We provide a step-by-step progression of the program through completion for a 4x4 board with 6 mines. The first cell to be clicked is randomly selected to be the one with coordinates (0,3). Because this cell reveals a value of 0, the program reveals the bordering squares as well (Figure 1). Each of the cells is added to the list of known cells and those with values greater than 0 are used to add to the list of logical statements. Next, the cell with coordinates (2,3) is clicked and revealed to have a mine (Figure 2). This cell is added to the list of known cells and used to update the list of logical statements. As a result, the program deduces that the cell with coordinates (2,2) must be clear, so it is added to the list of known cells and placed at the front of the priority queue of cells to be clicked. The next cell that is clicked is the cell with coordinates (2,2) as deduced by the program (Figure 3). This cell is added to the list of known cells and used to update the list of logical statements. The program then deduces that because the cells with coordinates (0,1) and (1,1) have mines, the cell with

coordinates (2,1) is clear, so it is added to the list of known cells and placed at the front of the priority queue of cells to be clicked.

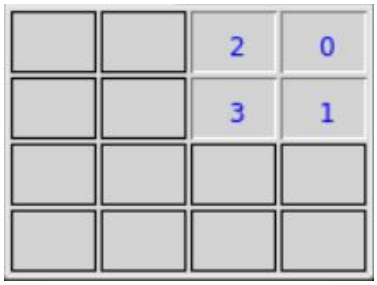


Figure 1

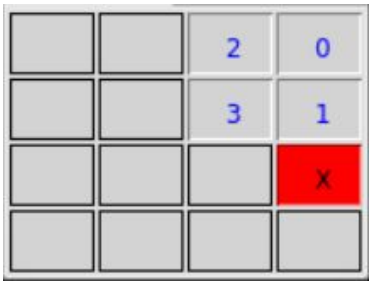


Figure 2

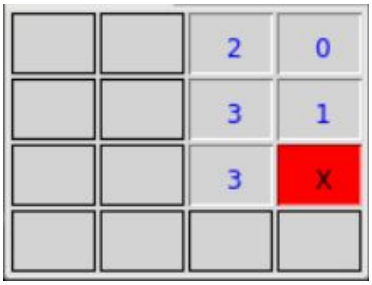


Figure 3

The program now clicks the cell with coordinates (2,1) which was deduced to be clear during the previous step (Figure 4). This cell is added to the list of known cells and used to update the list of logical statements. The next cell to be clicked is the cell with coordinates (2,0) and is found to be clear (Figure 5). This cell is added to the list of known cells and used to add to and update the list of logical statements. The next cell to be clicked is the cell with coordinates (3,3) and is revealed to have a mine (Figure 6). This cell is added to the list of known cells and used to add to and update the list of logical statements. As a result, the program deduces that the cells with coordinates (3,1) and (3,2) must be clear, so they are added to the list of known cells and placed at the front of the priority queue of cells to be clicked.

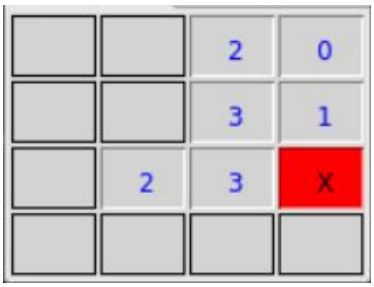


Figure 4

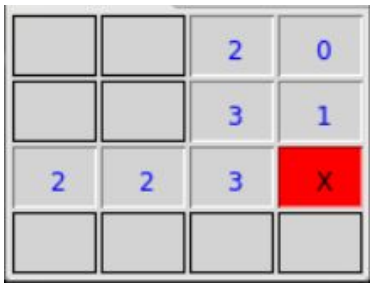


Figure 5

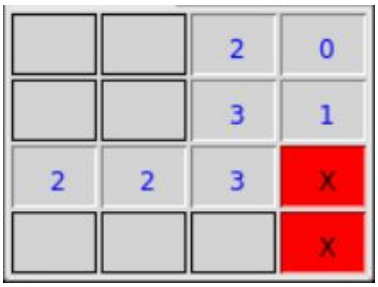


Figure 6

Next, the program clicks the cells with coordinates (3,1) and (3,2) which were deduced to be clear in the previous step (Figure 7). These cells are added to the list of known cells and used to update the list of logical statements. As a result, the program deduces that the cell with coordinates (3,0) has a mine and thus removes it from the priority queue. The next cell to be clicked is the cell with coordinates (0,0) and revealed to have a mine (Figure 8). This cell is added to the list of known cells and used to add to and update the list of logical statements. As a result, the program deduces that the cell with coordinates

(1,0) is clear and places it at the top of the priority queue. The program concludes after clicking the cell with coordinates (1,0) with 3 out of 6 mines identified (Figure 9).

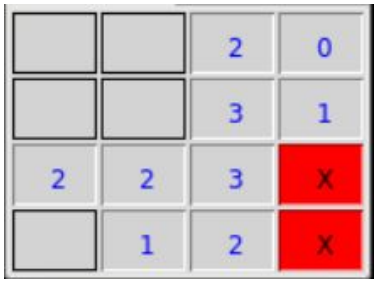


Figure 7

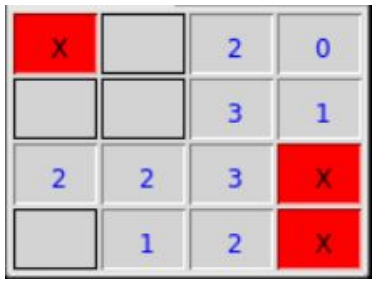


Figure 8

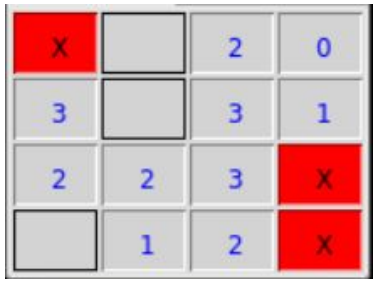
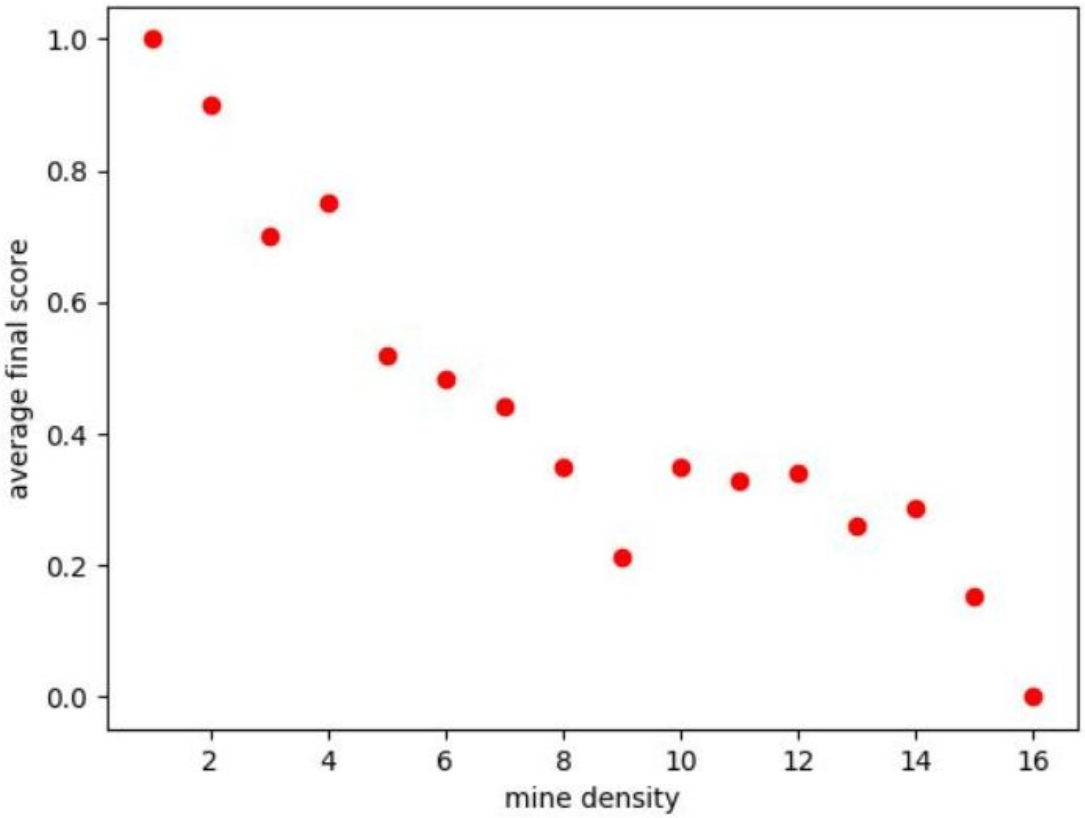


Figure 10

At every point in the program, given the information available to the agent and the state of the board, the program behaves as expected.

Performance: We analyze the average final score as a function of mine density for a 4x4 board. The average final score ranges between 0.5 and 1 for mine densities 1-5. For mine densities 6-16, the average final score falls between 0 and 0.5. The average final score generally decreases as mine density increases, which is consistent with what we expected to occur.

It is important to consider that as mine density increases, it also becomes more likely that the program will click on a mine during the first few steps of gameplay. For a mine density of 8 (half of the total number of cells) and greater, the program is able to safely identify up to 40% of the mines present on average. For a mine density of 13 and greater, the program is only able to do so for up to 30% of the mines present. Thus, we quantify such boards as “hard” boards.



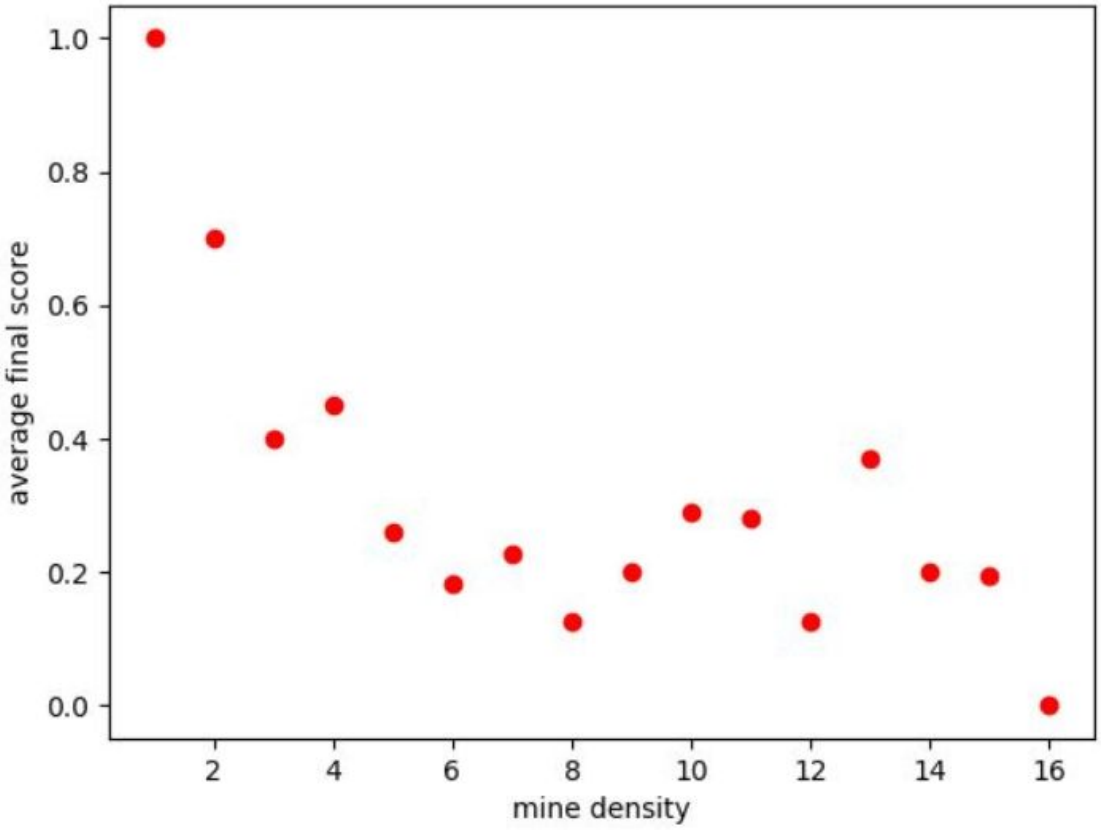
Efficiency: As the board size increases, the magnitude of information that must be stored in the knowledge to inform decision making increases as well. This space constraint is specific to the problem as whole, as it must be considered regardless of how an approach to the problem is implemented. An increase in board size also results in a program that takes longer to run, as there are more cells to analyze and click through. This is a problem specific constraint as well, as the time complexity of any approach will be dependent on the size of the board.

As for implementation constraints, it is important to consider that using the knowledge base to make deductions and update itself takes, in the worst case, $O(n^2)$ time. This is because the list of logical statements uses itself to consolidate information, which more time consuming as the size of the list increases. The size of the list is also dependent on the size of the board. This may be improved by finding a different way to store the information gathered to increase the efficiency of this operation.

Improvements: The additional information provided by the number of mines in a given board can be used for early termination of the program, removing the need for the program to carry out any additional deductions. The number of mines can also be used to add additional logical statements to the list, particularly when a clear cell is revealed

to have a value greater than zero. For example, if a clear cell with a value of 3 is revealed in a board with 6 mines, then the remaining cells, excluding its neighbors, must have 3 mines. This, expressed as a logical statement, provides additional information to the agent which can be used when making deductions and decisions.

The following graph shows the newly generated plot of mine density as a function of average final score.



#4

1. Based on our model, we could build this chain of influence by keeping track of intermediate facts such as if a group of 8 cells centered around a cell with the number 2 already being fulfilled. That way, we could infer that the other 6 cells definitely do not have a mine in them. This information, if kept around, can be combined with future information we gather by querying cells in the nearby area (i.e. there is a cell that indicates that there must be 7 mines around, but you have kept information around telling you that one of the cells around it must definitely not be a mine, then you know the other 7 are mines).

Another piece of information to keep around is how many of the revealed cells are "satisfied," or to what extent are they satisfied. So if you have a cell with the number 5 in it, and you only revealed 1 mine around it, then you know 4 other mines must be around it, you just do not know which cells they are in. Keeping this information and combining it with information gained from future queries proves to be very useful when attempting to clear the board.

2. The longest chain of influence is determined by the size of the board, or if there is an entire row or column of the board where the cell values are purely 0. This effectively cuts the board into two new boards, since a row of zeros would imply that there is absolutely nothing on one side of the board that could influence the other side. It acts as a divider.

3. The longer the chain of influence, the more information the solver must keep around and calculate with. This reduces the efficiency of the solver significantly, in terms of time and space complexity.

4. Particularly long chains of influence are created in boards where the mines are almost uniformly dispersed. This causes cells to always reveal some amount of useful information. Unlike when mines are all located in one region, then you might find a row of entirely zero value cells. This situation is more likely as the number of mines is neither too small nor too large, since having too few mines can lead to a row of all zeros, and having too many mines makes chains of influence almost too short to be worth computing.

5. As long as there is no dividing row or column of zeros, technically a chain of influence can span the length of the board. However, the influence it has will be very weak and the knowledge/advantage gained from it may not be worth the cost of keeping that information around.

6. Yes, since chains of influence that are very long often don't yield information valuable enough to be worth keeping around at the cost of efficiency, this tells us that keeping chains of short length is the best approach, as you are able to make educated guesses in local regions, without taking too large of a hit in terms of computational complexity.

7. Yes, solving minesweeper is incredibly challenging because you must keep track of a lot of knowledge and use that to make educated guesses. Additionally, the element of luck is required in almost all games of minesweeper, except for

extremely easy ones, (but again, getting an easy board requires luck too) so solving minesweeper is challenging for humans, as well as computers.

#5

As the graph depicts, when adding uncertainty, the performance of the solver deteriorates. More mines are detonated as the solver attempts to clear the board. The chosen method to add this uncertainty was the first one mentioned, that is, when the solver presses on a cell in the grid and it is not a mine, it may or may not receive a clue about the cells around it. This gave the solver less information to work with, and its predictions became less accurate as a result of that.

The metrics used were:

1. Grid dimensions: 5x5
2. Number of mines: 15
3. Probability that a cell would not reveal a clue: .3
4. Number of iterations run for each version of the game to find the average: 10

As you can see, when this limitation is applied, the number of mines detonated on average, rose from 11 to 14. The ratio of mines to number of cells already makes this specific version very hard to solve. It would be classified as a “hard” game or “expert mode” in traditional minesweeper terms.