Zekun Zhang, Ziqi Wang, Richa Rai

Artificial Intelligence
October 15, 2019

# Assignment 2

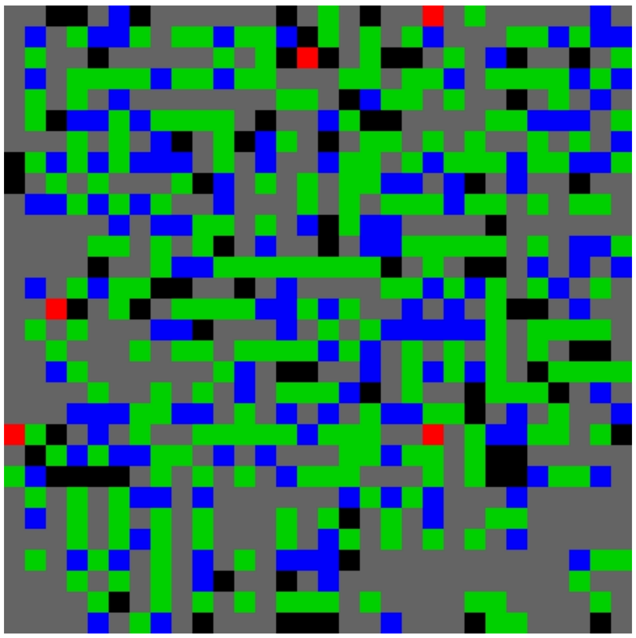

Figure 1: A sample traversed mine field! Dimenions: 20x20, mines = 240

## 1 Part 1

### 1.1 Representation

The mine field is represented as an dimxdim square board similar to the maze of the previous assignment. Actual mines are displayed in black color, flagged mine cells are displayed in green, flagged but unmined cells are displayed in blue and the traversal of the mine in gray if it its in empty space or red if we happen to step into a mine cell. A sample can be seen in figure 1.

### 1.2 Inference

Inference is done in the following way.

1. Initialize by assigning each cell to mine prob $P_m = |cells| * \frac{1}{|mines|}$
2. Pick the point with the lowest mine potential $P_M$.
3. Query the point. If Free get the mine clue. If mined, lose a life; continue.
4. Based on the clue distribute probability of mine in a a cell in the following manner:
    (a) if clue = 0, then all surrounding cells have $P_m = 0$

(b) if clue = 1, then all surrounding cells have $p_m = 1$

(c) when clue = 0 or 1,then redistribute or absorb the probability of that cell to all the other cells that where discovered on the same step and have not been visited.

(d) otherwise each cell's $P_m = P_M + \frac{|clue|}{d} - P_m\frac{|clue|}{d}$, where d is the amount of non discovered neighbors.

5. If a cell's $P_m > minedThreshold$ then it is considered mined, it is flagged and not queried until the the flag is removed.

6. Repeat steps 2-5 until all the cells have been visited or flagged.

The $minedThresholdparameter$ can be arbitrarily chosen, it was fixed at 0.5 for the tries presented in this report.

## 1.3 Decision Making

AS presented above, decision is made by picking the cell where our belief that it is mined is the lowest. Since this is a probabilistic approach and the threshold for considering a cell has dangerous is picked as 0.5 (point of uncertainty), there is the risk exploding even in relatively safe cells. This can happen as if we don't query enough points around a candidate cell, we might have a skewed belief due to our initialization that the cell is safer that it actually is. A way to counter this would be to pick a much more conservative initialization scheme that assigns higher $P_{m_{init}}$ to cells, as in large field if can lead to overoptimism and abundant explosions!

## 1.4 Play Through and Performance

A playthrough is included in the gif folder for a field of dimensions 20x20 and 240 mines. The algorithm I implemented has some difficulty identifying large blops of bombs. It tends to start from edge point, which is a decision I support, as corner points tend to be the most informative. The score results can be seen in figure 2.

## 1.5 Efficiency

The complexity of the algorithm has a worst case of $O(n^3)$ as each cell could trigger a re-computation of almost the entire matrix. It has an amortized cost of $O(n^2)$ as , we limit the redistribution or absorption of potential mass to one hop neighbors. The recomputation of probability is the main culprit of the workload, and in larger fields it can add up. Although for mazes up to 30x30 it runs reasonable fast in under a second. The main improvement of the implementation would be vectorization of all the computations involved. Although care was to taken to minimize unnecessary computations, checks and for loops, there are some points that for loops could not be avoided, as the vectorized alternative is extremely complex.

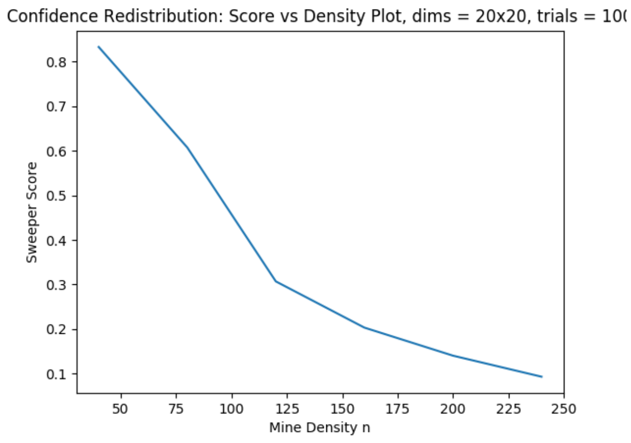

Figure 2: Performance of Implemented algorithm vs mine density.

The team members of this group are Zekun Zhang, Ziqi Wang, and Richa Rai. The three members discussed how to best implement the concepts and what methods may be more efficient. They then each worked on code and compared their different ideas. They then worked together to compile the project and discuss the analysis components.