Minesweeper AI

Quanhao Li, Saif Rahman, Adit Kumar

December 14, 2016

Abstract

The objective of our project was to determine whether Hill Climber or Pattern Recognition would be more suitable to solve Minesweeper. The way we determined which algorithm was better was by comparing their respective accuracies.

1 Introduction

Minesweeper is a single-player game played on a digital rectangular grid. The objective of the game is to "flag" or discover the location of all the bombs without detonating any of them. Every square can only hold two different kinds of information: the number of bombs adjacent to said square (can be zero) and if that square is a bomb (clicking on a bomb ends the game, resulting in a loss). The player plays the game by clicking on squares throughout the grid using the limited information gained from every square revealed.The game also includes a high score system for every level of difficulty and records the fastest time in completion of the game.

The Minesweeper game is a specialized SAT(SATISFIABILITY) problem because it can be rewritten as a set of combination logic and literals. However, every formula that can be made is for very specific scenarios in the game. This was shown in a study by Kaye [Kay00] where he attempted to find a perfect solution to the game. Kaye showed that the Minesweeper game could be used to "simulate digital computers," but he showed it using NOT, AND, OR and XOR gates. As described in the "MINESWEEPER" paper written by Preslav Nakov and Zile Wei [PN03], "the presence of AND and NOT gates (or alternatively OR and NOT gates) proves that MINESWEEPER is NP-complete."

It is also important to consider the difficulty of Minesweeper. A perfect example is shown in the paper "Learning Minesweeper with Multirelational Learning," written by Lourdes Pena Castillo and Stefan Wrobel [LPC03]. In this paper, the authors describe a game of Minesweeper with the following constraints: 8x8 grid (64 tiles) and 10 mines.There were two calculations performed: the first for the complexity and the second for the probability of winning. For the first calculation, a scenario was imagined where the player has selected every single tile except 11 tiles on the grid without having to flag a single bomb. With the assumption that the player makes the correct move to win the game, Castillo and Wrobel calculated that there are 10^{71} possible move sequences to win. For the second calculation, the same assumption was made about the game states. Since there are 54 tiles that can be chosen safely for the first move (assuming all move made thereafter are correct), the probability of making a good first move is 54/64. Upon reaching the last move, the probability of making the correct move is 1/11. Thus the total probability of a "random" player winning the game is $1/({}_{64}C_{54})$ or approximately 10^{-12}. The 8x8 game with 10 bombs is considered the "easy playing level."

Another thing to note is that there are sometimes game states that require the player to make a guessing move because there was not enough information provided by the squares already revealed. An example of one of these states was provided in the paper "Some Minesweeper Configurations" by Richard Kaye [Kay07] (shown below). All the squares surrounding the 2's are squares that have yet been clicked, but no matter how the information is processed, there's no way to guarantee which unpicked square is a bomb and which aren't. At most, the best that could be done is to find the square(s) with the least probability of being a bomb and hope that clicking on said square would provide more information instead of costing the game.

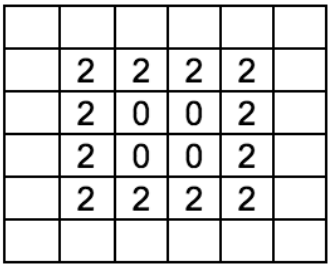

Figure 1:

2 Hill Climbing Algorithm

The basic idea is to find the square with the highest probability of advancing the game state without ending the game with a loss. There are some moves in Minesweeper that are 100% guaranteed to not end the game, so evidently those moves would be the first to be made. When no obvious moves can be made, a ticket system was used to calculate the square that would be least likely to be a bomb.

The ticket system works by assigning "tickets" to every square on the game tile based on their proximity to squares holding information about neighboring bombs. The square that held the smallest ticket number would then be the square with the lowest probability of being a bomb.

We used a matrix to hold the ticket information for each square. To implement this, a hash table was used. This choice was made because a hash table provided the necessary functionality of a matrix as well as reduced searching time. One of the edge cases we had to consider was, what if the ticket system produced a series of moves that held the same number of tickets? If this was the case, a random move from that series of moves would be selected. We also implemented a threshold value because we did not want the algorithm to make a move if the smallest number of tickets was too large and the probability of guessing a bomb was too high. With a threshold of 2, only moves with a probability of picking a bomb was less than 50% was made.

Because Minesweeper is not deterministic, we will always have to make a guess throughout the game (when there are no obvious moves and the ticket system does not produce a move that could be made). This algorithm can even be classified as greedy due to its nature of picking the best move at each step.

3 Pattern Recognition Algorithm

This algorithm uses first order logic to determine the state of each tile on a grid. Using pre-existing knowledge of bombs from surroundings tiles, we can either mark or dig tiles that we determine are bad or safe respectively. We try to mark the most amount of bombs in order to readily gain an advantage in the game. Our methodology stems from the fact that if we have a greater knowledge base, better inferences can be made about other tiles.

This system works by implementing strategies that are often taken by pros when they need to make inferences about certain tiles. According to the an article by Barrett [Bar99], the two main strategies are called 1-1 and 1-2 techniques. When such a pattern is recognized, an automatic gesture of a flag or dig is made without thinking. 1-1 and 1-2 techniques can be generalized where certain cases can reduce to an application of these techniques. Our AI takes this into account.

As for design implementations, our AI reduces tiles by subtracting the number of surrounding flagged tiles from the number of the tile. This allows it to search for all cases of reduced patterns. Additionally, this algorithm strives to maximize its accuracy, so this algorithm was run on all tiles and their adjacent neighbors. Implementing such an algorithm required many small functions that return boolean values to verify if this fell under our situation. In the worst case, if our algorithm failed, a random guess was made. Otherwise, there would be no progress made towards to the solution state of the game.

4 Results

Statistics	Easy	Intermediate	Expert
Average Number of Iterations	18.60	26.98	29.22
Average Time Spent	0.18	0.56	0.65
Average Number of Bombs Found	5.28	19.47	20.35
Accuracy	40%	30%	2%
Loss On First Turn	32%	38%	56%

Table 1: Hill Climbing Algorithm Results.

Statistics	Easy	Intermediate	Expert
Average Number of Iterations	18.16	34.02	48.55
Average Time Spent	0.18	0.83	1.11
Average Number of Bombs Found	7.00	31.37	40.14
Accuracy	64%	50%	4%
Loss On First Turn	26%	16%	26%

Table 2: Pattern Recognition Algorithm Results.

5 Conclusion

It is evident from the results that as the difficulty level of game increases(which is density of mine on a given grid), the probability of making a correct move decreases. It can also be noted that both the algorithms mentioned above runs in O(n*m) time where "n" and "m" depends on size of grid used. This running time allows both the algorithm to play the typical (50x50) grid game in worst running time of less than 5 secs. It is also found that running time of hill climbing based algorithm was slightly better than pattern based algorithm.However the accuracy of a move is found better in case of Pattern based algorithm. It can be concluded that there is a tradeoff between running time and accuracy.While both the Hill climbing and pattern recognition excels at determining safe tile and guaranteed mines, both algorithms rely on guessing the right move when the game is in a state where the probability of selecting any tile is same. Hence, Further research can be conducted to come up with a strategy to determine correct move when the game is in this state.

References

[Bar99] Sean Barrett. Minesweeper: Advanced tactics, jan 1999.

[Kay00] Richard Kaye. Minesweeper is np-complete, mar 2000.

[Kay07] Richard Kaye. Some minesweeper configurations, may 2007.

[LPC03] Stefan Wrobel Lourdes Pena Castillo. Learning minesweeper with multirelational learning, aug 2003.

[PN03] Zile Wei Preslav Nakov. Minesweeper, may 2003.