Minesweeper Solver: Enhancing Performance through Probabilistic Methods

by: Ollie Bland & Shawn Frye

Introduction:

Minesweeper, a classic puzzle game, demands strategic thinking and careful analysis. Our objective in this project is to develop an Artificial Intelligence that can effectively navigate and excel in the game of Minesweeper, overcoming the challenges and randomness of the game. This report outlines our journey, the challenges we faced, and the strategies implemented to enhance the AI’s performance in Minesweeper.

Development Process:

We started our journey by discovering a basic console-based Minesweeper game through an extensive Google search and an exploration of GitHub repositories. This game served as an excellent foundation, yet it required substantial modifications to be optimally compatible with our AI's gameplay mechanics. Unlike other Minesweeper variations that reshuffle the board to prevent first-move losses, our version maintains the initial configuration, leading to a higher probability of losses based on luck, especially in high-density boards.

Once the AI completes its initial move, the board should exhibit a slight opening, letting the AI to loop through for definitive moves or mines based on the neithbors. In the absence of guaranteed moves, the AI resorts to a probabilistic approach, choosing the cell with the most favorable odds. For instance, if a decision boils down to two options - one adjacent to a “2” and the other to a '4' - the AI will go for the cell next to the '2', given the lower probability of being a mine.

We initially integrated an A\* heuristic into our algorithm, intending to enhance performance after the definitive and neighbor checking but prior to the probabilistic evaluation. However, this integration led to a noticeable decline in the win rate, prompting us to omit the heuristic in favor of our probabilistic strategy.

Code:

### Minesweeper **Class:**

* \_\_init\_\_: Initializes a Minesweeper game with a specified number of rows, columns, and mines, and optionally a callback function.
* external\_move: Processes a move made externally (such as by the solver), applying a flag or revealing a cell, and updating game state.
* generate\_board: Generates the Minesweeper board with mines randomly placed, and numbers indicating nearby mines.
* display\_board: Displays the current state of the Minesweeper board to the console.
* play: Starts a manual game of Minesweeper, allowing a user to make moves until the game is won or lost.
* reveal: Reveals a cell and recursively reveals adjacent cells if they are empty.
* current\_board: Returns a string representation of the current board state.

### Solver Class:

* \_\_init\_\_: Initializes a Minesweeper solver with a reference to the Minesweeper game it will solve.
* playGame: Plays the Minesweeper game, deciding on and making moves until the game is over.
* heuristic: Calculates a heuristic value for a given cell based on its distance to revealed numbers.
* decide\_move: First determines if it is the opening move of the game, in which case it calls the pick\_random\_corner method. Then, it loops through the adjacent cells to identify definite moves or mines based on the current board state.If no definite move, the method transitions to a probability-based strategy to make an informed decision.
* makeMove: Makes a move in the Minesweeper game by calling external\_move on the Minesweeper instance.
* gameCallback: A callback function that can be used to display the Minesweeper board after each move.
* make\_random\_move: Chooses a random cell to reveal.
* pick\_random\_corner: Chooses a random corner cell to reveal.

### \_\_main\_\_ Block:

* Initializes and plays a specified number of Minesweeper games, size of board, and mine density using the Solver class.
* Calculates and displays the win percentage of the solver.

Challenges and Solutions:

In developing the AI, we encountered several challenges, particularly in relation to the inherent randomness of Minesweeper and the need for the AI to make educated guesses under uncertainty. We addressed these challenges through a combination of strategies, including the implementation of a probabilistic approach to guess-making and the careful analysis of board patterns.

Future Improvements:

In the future, to significantly enhance our AI's performance in Minesweeper on larger and densely populated boards, we are planning to integrate advanced pattern matching techniques. This will allow for a comprehensive analysis of complex board patterns, aiding the AI in solving configurations that are currently challenging, and enabling a more strategic approach.

The implementation of these pattern matching techniques is hoped to transform the AI’s game play, improving its ability to navigate through challenging scenarios, and increasing its overall win rate and efficiency. This is not just about increasing the AI’s performance; it’s about deepening its understanding of the game’s mechanics and strategies.

Also, we are exploring the incorporation of machine learning techniques to enable the AI to learn from past games and adapt its strategies over time. This approach will result in a Minesweeper AI that is both reactive and proactive, capable of anticipating challenges and adjusting its strategies in real-time.

Conclusion:

Through research and algorithmic optimizations, we have developed an AI capable of playing Minesweeper at a high level of proficiency. While the game's randomness does introduce an element of luck, our AI has demonstrated the ability to make intelligent decisions and maximize its chances of success. We believe that the insights gained from this project will prove invaluable in future endeavors involving strategic game play and AI development.

References

Anand, A. (2020, June 24). *Create minesweeper using python from the basic to advanced*. AskPython. https://www.askpython.com/python/examples/create-minesweeper-using-python

Candra, A., Budiman, M. A., & Pohan, R. I. (2021). Application of A-star algorithm on Pathfinding Game. *Journal of Physics: Conference Series*, *1898*(1), 012047. https://doi.org/10.1088/1742-6596/1898/1/012047