Minesweeper Final AI Report

Team number : 80
Member #1 : Alonso Lopez / Alonsl1
Member #2 : Dararith Darrien Sao / ddsao

I. Minimal AI

I.A. Briefly describe your Minimal AI algorithm. What did you do that was fun, clever or creative?

Our minimal AI algorithm for the game is designed to navigate a board and avoid mines efficiently. The constructor initializes the board dimensions, the total number of mines, the starting position, and a dictionary to track moves, while setting up the board with all cells initially marked as unexplored and preparing to track neighboring cells. The AI employs a layered decision-making process: in the first layer, if an uncovered cell has zero neighboring mines, the AI adds its neighbors to the list of safe moves; in the second layer, if no safe neighbors are left, calculates the danger level of each cell based on its known neighbors and selects the least dangerous unexplored cell to uncover next. It employs a simple heuristic of taking the known values of each neighbor and adds 1 to the danger level with each neighbor higher than 0. We think this was clever because it was able to solve all 1000 Easy worlds in as little as 2 seconds, just by predicting and avoiding the cell with the least “danger”.

I.B Describe your Minimal AI algorithm's performance:

Board Size	Sample Size	Score	Worlds Complete
5x5	1000	1000	1000
8x8	1000	0	0
16x16	1000	0	0
16x30	1000	0	0
Total Summary	4000	1000	1000

II. Final AI

II.A. Briefly describe your Final AI algorithm, focusing mainly on the changes since Minimal AI:

The final AI algorithm introduces several enhancements over the minimal AI version. In addition to tracking moves, it now maintains sets for safe moves, flags, uncovered cells, and a frontier of potential moves. The constructor initializes these additional sets, and the getNeighbors function has been added to identify neighboring cells, while the getFlags function counts the flagged neighbors around a cell. By implementing updateFrontier, we analyze flagged and uncovered neighbors, dynamically adjusting the sets of safe moves and flags. Now there are three levels to the getAction function: the first level to get all the 0 cells out of the way, the second level to update the danger list with the frontier, and the third level to randomly guess. We were pleased to see that the 645 of the Beginner worlds were solved in about 12 seconds, and the 554 of our Intermediate worlds were solved in about 2 minutes and 30 seconds.

II.B Describe your Final AI algorithm's performance:

Board Size	Sample Size	Score	Worlds Complete
5x5	1000	1000	1000
8x8	1000	645	645
16x16	1000	1108	554
16x30	1000	0	0
Total Summary	4000	2753	2199

III. In about 1/4 page of text or less, provide suggestions for improving the performance of your system.

To enhance the performance of the AI system, several strategies could be implemented. Firstly, introducing a backtracking mechanism would allow the AI to revisit previous decisions and explore alternative paths if it encounters a high-risk situation or an unsolvable state. This approach can significantly reduce the chances of hitting a mine by enabling the AI to reconsider and adjust its moves based on newly uncovered information.

Additionally, maintaining a history of past performances could improve the AI's decision-making over time. By recording and analyzing successful and unsuccessful strategies, the AI can learn from its experiences, identify patterns, and optimize future moves. This historical data can be used to refine the danger assessment algorithm, making it more accurate and reliable.

Another improvement could involve enhancing the danger assessment by incorporating more advanced probabilistic models or machine learning techniques. These models can better predict the likelihood of mines based on the current state of the board and past game data. Implementing a more sophisticated risk evaluation system can help the AI make more informed and safer moves.