

Minesweeper SAT Solver: A Constraint Satisfaction Approach to Minesweeper

Aymane Lotfi, Arthur Vogels

February 23, 2025

1 Introduction

Minesweeper is a classic puzzle game where players uncover cells in a grid without triggering hidden mines. Each revealed cell displays a number indicating the count of mines in its adjacent cells. Although seemingly simple, optimal solving can be challenging, especially when guesswork is involved.

In this project, we model Minesweeper as a Boolean Satisfiability (SAT) problem by encoding the game rules and constraints into a Conjunctive Normal Form (CNF) formula. A modern SAT solver (Glucose3) is then employed to compute a valid mine configuration. Additionally, an interactive Tkinter GUI allows the user to play the game by clicking on cells—if a mine is revealed, the game ends; otherwise, safe cells and their numbers are shown. The interface also provides options to **Solve** the puzzle automatically or **Reset** for a new board.

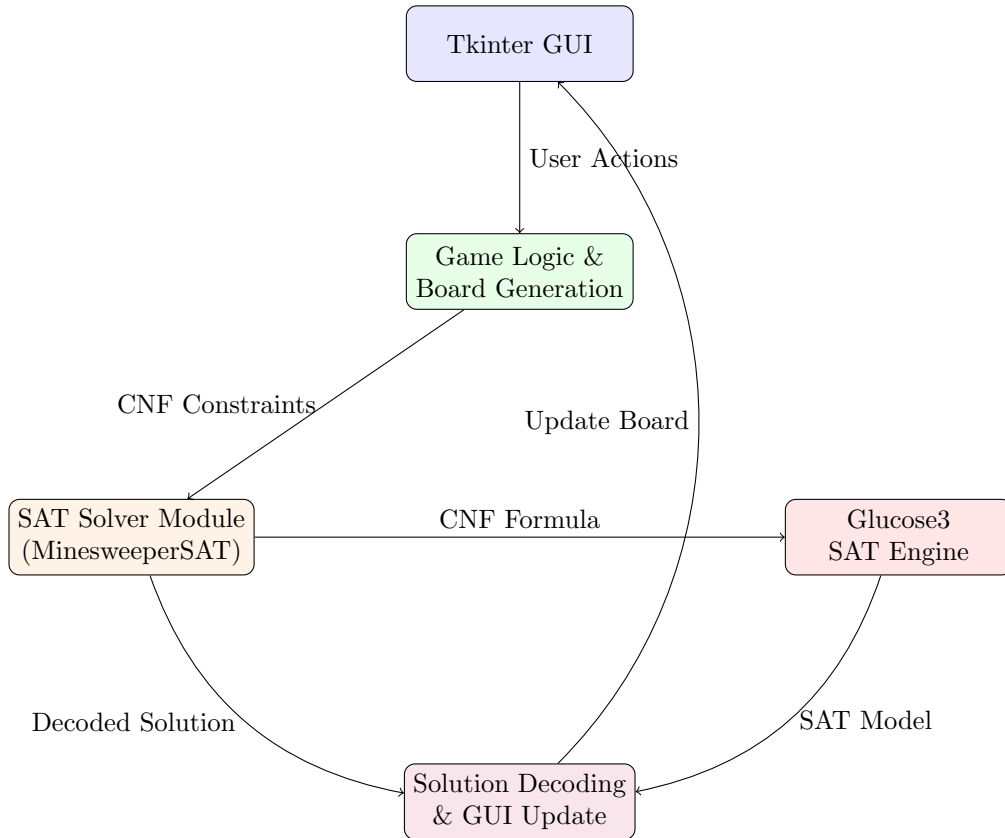


Figure 1: System Architecture of the Minesweeper SAT Solver..

2 SAT Solver Formalism

In our approach, the Minesweeper board is modeled as a two-dimensional grid

$$G = \{(i, j) \mid 1 \leq i \leq n, 1 \leq j \leq m\}.$$

For each cell $(i, j) \in G$, we associate a Boolean variable x_{ij} where:

$$x_{ij} = \begin{cases} 1, & \text{if a mine is present at } (i, j); \\ 0, & \text{otherwise.} \end{cases}$$

2.1 Local Number Constraints

If a cell (i, j) is revealed and displays a number k (where $0 \leq k \leq 8$), it indicates that exactly k of its neighboring cells contain mines. Let the set of neighbors be defined as:

$$N(i, j) = \{(p, q) \in G \mid |p - i| \leq 1, |q - j| \leq 1, (p, q) \neq (i, j)\}.$$

The corresponding constraint is:

$$\sum_{(p, q) \in N(i, j)} x_{pq} = k.$$

Since SAT solvers require formulas in Conjunctive Normal Form (CNF), we convert this cardinality constraint into CNF by splitting it into two parts: *at least* k and *at most* k .

At Least k : For each subset $S \subseteq N(i, j)$ with

$$|S| = |N(i, j)| - k + 1,$$

we require that at least one cell in S is a mine:

$$\bigvee_{(p, q) \in S} x_{pq}.$$

At Most k : For each subset $T \subseteq N(i, j)$ with

$$|T| = k + 1,$$

we ensure that not all cells in T are mines:

$$\bigvee_{(p, q) \in T} \neg x_{pq}.$$

Thus, the full constraint for cell (i, j) is:

$$\Phi_{ij} = \left(\bigwedge_{\substack{S \subseteq N(i, j) \\ |S| = |N(i, j)| - k + 1}} \bigvee_{(p, q) \in S} x_{pq} \right) \wedge \left(\bigwedge_{\substack{T \subseteq N(i, j) \\ |T| = k + 1}} \bigvee_{(p, q) \in T} \neg x_{pq} \right).$$

2.2 Global Mine Count Constraint

Let M denote the total number of mines on the board. The global constraint is:

$$\sum_{(i, j) \in G} x_{ij} = M.$$

This is similarly decomposed into:

$$\Phi_{global} = \left(\bigwedge_{\substack{S \subseteq G \\ |S| = |G| - M + 1}} \bigvee_{(i, j) \in S} x_{ij} \right) \wedge \left(\bigwedge_{\substack{T \subseteq G \\ |T| = M + 1}} \bigvee_{(i, j) \in T} \neg x_{ij} \right).$$

2.3 Complete SAT Instance

The final CNF formula that represents the Minesweeper board is given by:

$$\Phi = \left(\bigwedge_{(i,j) \in G'} \Phi_{ij} \right) \wedge \Phi_{global},$$

where $G' \subseteq G$ is the set of cells with a known clue.

This CNF formula is then passed to the SAT solver (Glucose3), which searches for a satisfying assignment. A solution to Φ corresponds to a valid mine configuration that satisfies all local and global constraints.

3 System Implementation

3.1 Project Organization

The project is divided into three main modules:

- `solver.py`: Contains the `MinesweeperSAT` class and the SAT encoding logic.
- `gui.py`: Contains the `MinesweeperGUI` class that implements the interactive gameplay using Tkinter.
- `main.py`: The entry point that initializes the GUI.

3.2 Interactive Gameplay

Users can play by clicking on cells. If a clicked cell is safe, its neighbor count is revealed; if a mine is clicked, a **Game Over** message is displayed. The **Solve** button allows the SAT solver to reveal the complete solution, while the **Reset** button generates a new board.

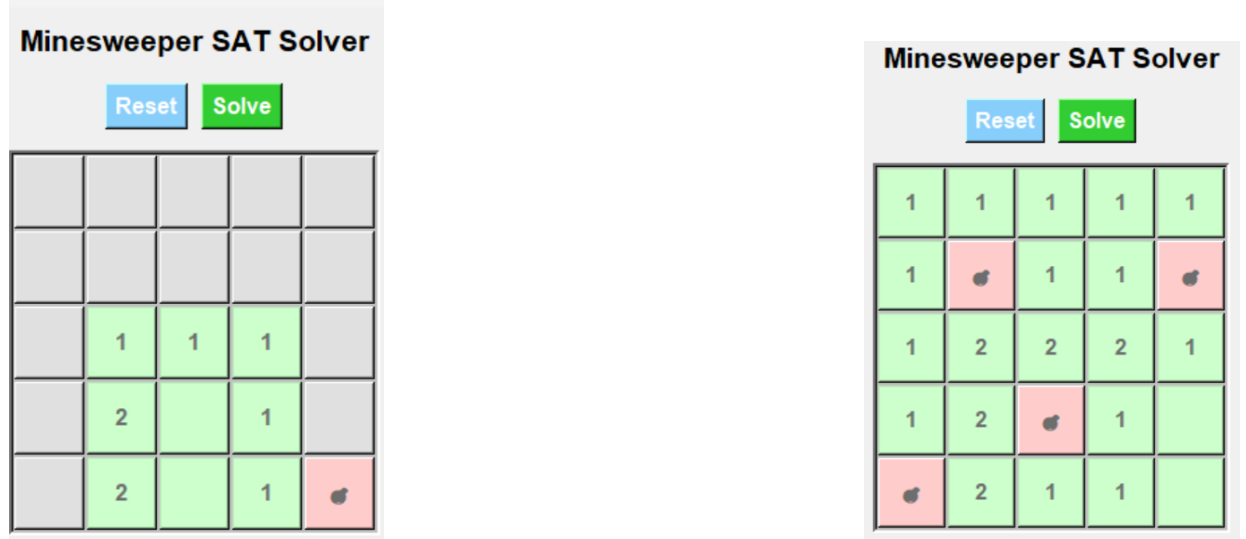


Figure 2: GUI Overview

4 Results and Discussion

4.1 Gameplay Observations

- **Safe Cell Reveal:** Clicking on a cell with no adjacent mines triggers a flood-fill that automatically reveals all contiguous safe cells.

- **Game Over:** Clicking on a cell containing a mine immediately ends the game.
- **Solver Functionality:** The SAT solver correctly computes a valid configuration based on the number constraints, and the solution is displayed when the **Solve** button is pressed.

5 Conclusion

This project demonstrates a SAT-based approach to solving Minesweeper by translating game constraints into a CNF formula. The detailed SAT formalism, combined with an interactive Tkinter GUI, shows both the theoretical underpinnings and practical application of constraint satisfaction techniques. The modular design aids maintainability and provides a solid foundation for future enhancements, such as scalability improvements or additional game features.

6 Future Work

Potential future improvements include:

- Enhancing scalability for larger grids.
- Incorporating advanced deduction methods to reduce random guessing.
- Adding dynamic difficulty adjustment and high-score tracking.
- Further refining the visual design and user experience.