Minesweeper

Knowledge and Data Mining Project

Leonardo Amato (ID:2028621),
Lorenzo Corrado (ID:2020623)

October 05, 2022

Università degli Studi di Padova

1. Introduction
2. Propositional Logic
3. Implementation
4. Conclusion

1 Introduction

2 Propositional Logic

3 Implementation

4 Conclusion

The Game

Minesweeper is a popular single-player puzzle video game released in 1990. This game became very popular as it was always present in the various releases of Windows. The aim of this game is to clear a minefield without detonating mines.

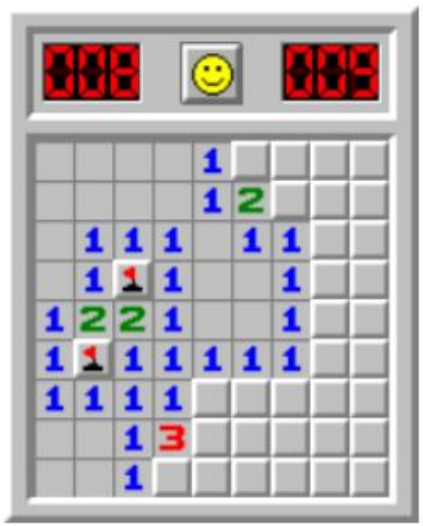


Figure 1: Example of game interface.

The Rules

1. The playing field consists of a square/rectangular field, which is divided into many cells. Each cell is uncovered by clicking on it;
2. Many cells contain mines: when a cell with a mine is clicked, it explodes and ends the game;
3. If the clicked cell does not contain mines, then we will be able to read the number of adjacent mines. If there are no adjacent mines the cell will have no number in it and the game will automatically click all the adjacent cell.
4. The game is won when all the cell that do not contain mines are identified with a flag.

The Technique

- It is necessary to read well the number inside the uncovered boxes as it will tell us how many adjacent boxes contain mines;
- If possible, it is always best to uncover boxes that are near areas that do not have a high number of adjacent mines.

Minesweeper is not always solvable without trying. Indeed, there are some game scenarios in which there is no certain method to determine in which of the squares the mine is present, in this case the player is forced to decide randomly.

1 Introduction

2 Propositional Logic

3 Implementation

4 Conclusion

Formalization

Given the structure of the game, it is possible to formalize any possible scenario of the game from a **propositional logic**.

- The techniques is to encode any game state in a set of formulas S and use a **SAT** solver to decide if for every cell $c_{i,j}$:
  - $c_{i,j}$ is safe, i.e. $S \models \neg Mine_{i,j}$;
  - $c_{i,j}$ is not safe, i.e. $S \models Mine_{i,j}$;
- As we have already mentioned, there are cases when we have to discover a cell randomly. So if we cannot decide if $c_{i,j}$ is safe or not we can apply **propositional model counting** to count the number of models in which $c_{i,j}$ is safe and choosing $c_{i,j}$ with the highest number of model in which it is safe.

Example of Formalization

In order to encode a game state into a set of formulas S let us consider the following example:

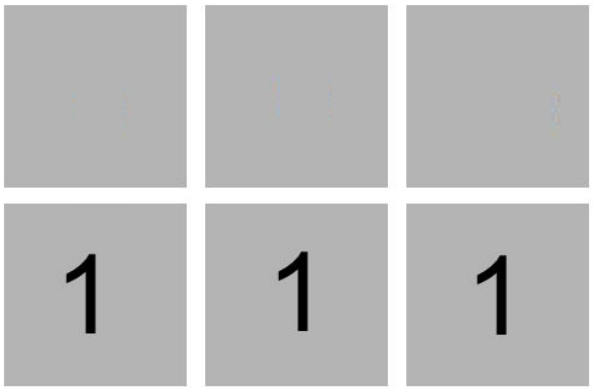


Figure 2: Example of a game state.

Let us consider the variable $c_{i,j}$ be true iff (i, j) contains a mine. In this case the variable of interest are the three unknown box $c_{1,1}, c_{1,2}, c_{1,3}$. We know the following facts:

Example of Formalization (cont'd)

- $c_{2,1}$ has exactly one adjacent mine, therefore we have:

$$(c_{1,1} \vee c_{1,2}) \wedge (\neg c_{1,1} \vee \neg c_{1,2})$$

- $c_{2,2}$ has exactly one adjacent mine, therefore we have:

$$(c_{1,1} \vee c_{1,2} \vee c_{1,3}) \wedge (\neg c_{1,1} \vee \neg c_{1,2}) \wedge (\neg c_{1,1} \vee \neg c_{1,3}) \wedge (\neg c_{1,2} \vee \neg c_{1,3})$$

- $c_{2,3}$ has exactly one adjacent mine, therefore we have:

$$(c_{1,2} \vee c_{1,3}) \wedge (\neg c_{1,2} \vee \neg c_{1,3})$$

- There is exactly one mine left (which is a global constraint):

$$(c_{1,1} \vee c_{1,2} \vee c_{1,3}) \wedge (\neg c_{1,1} \vee \neg c_{1,2}) \wedge (\neg c_{1,1} \vee \neg c_{1,3}) \wedge (\neg c_{1,2} \vee \neg c_{1,3})$$

The set of formulas S represented by the conjunction of the 4 clauses above, can be used to check , for examples, if $(1, 1)$ is a mine by deciding whether $S \models mine_{1,1}$.

1 Introduction

2 Propositional Logic

3 Implementation

4 Conclusion

Setup

The project was implemented using the Python programming language in Google Colab.

Figure 3: The project setup.

Furthermore, we used the standard python libraries in the programming part and the python library MiniSat[1] for the propositional logic part of the implementation.

[1] https://github.com/pgdr/python-minisat

Implementation

A game state can be represented through a set of triplets S, therefore we can say that if $(k, i, j) \in S$ then we are sure that:

- (i, j) is safe that can be expressed with the prop. formula:

$$Mine_{ij}$$

- There are exactly k adjacent mines to (i, j), this can be expressed with the prop. formula:

$$\bigwedge_{\substack{I \subseteq Adj(i,j) \\ |I|=8-k+1}} \bigvee_{(x,y)\in I} Mine_{xy} \wedge \bigwedge_{\substack{I \subseteq Adj(i,j) \\ |I|=k+1}} \bigvee_{(x,y)\in I} \neg Mine_{xy}$$

At last we know that there are exactly n in the field, this can be expressed with the prop. formula:

$$\bigwedge_{\substack{I \subseteq AllPos \\ |I|=w*h-n+1}} \bigvee_{(x,y)\in I} Mine_{xy} \wedge \bigwedge_{\substack{I \subseteq AllPos \\ |I|=n+1}} \bigvee_{(x,y)\in I} \neg Mine_{xy}$$

Example of Representation

By mapping each term $Mine_{i,j}$ to a univocal number we are able to express any game state in the following way:

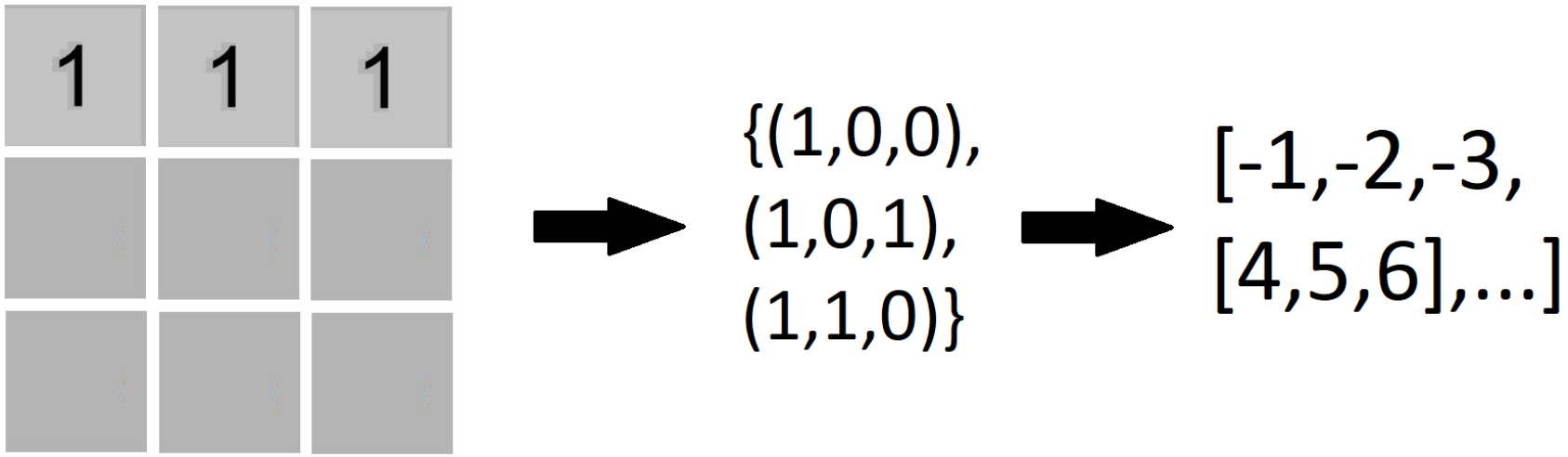


Figure 4: Example of game state representation.

Model Counting

Using the function **enum_models()** found in the MiniSat library we can perform model counting in uncertain situations so as to choose the move that maximizes the probability of success. To do so we:

- For each term $mine_{i,j}$ with uncertainty ($S \not\models mine_{i,j}$ and $S \not\models \neg mine_{i,j}$), we count the number of models in which $\neg mine_{i,j}$ appears;
- We return the position that is safe in the greatest number of models.

Example of First Scenario

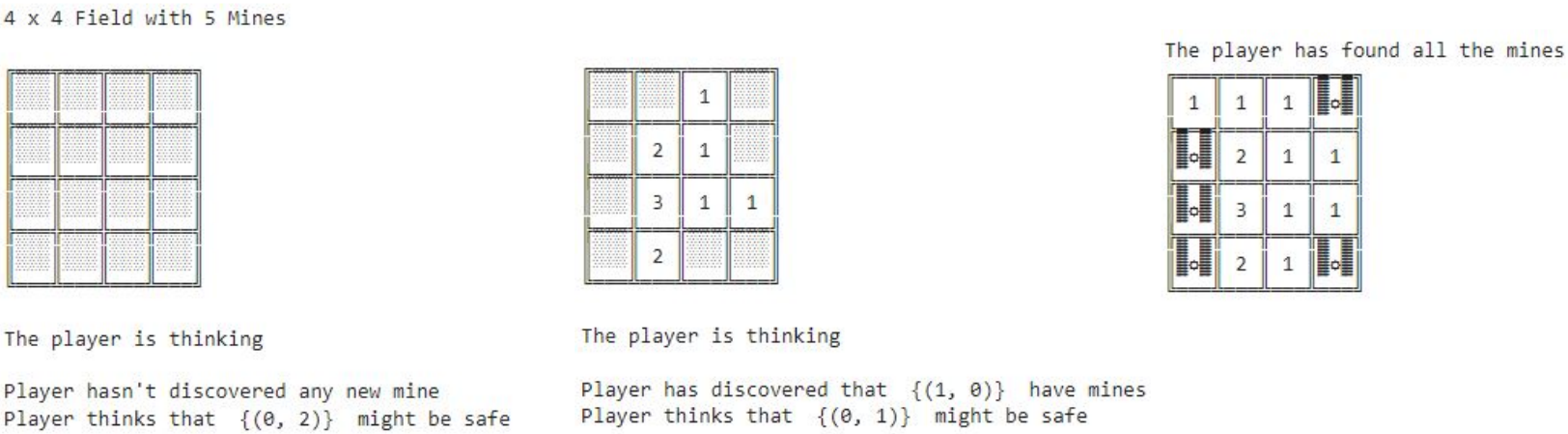
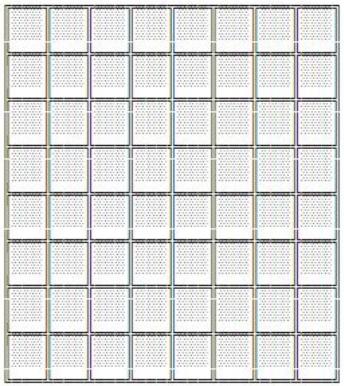


Figure 5: Example with a 4x4 grid and 5 mines.

Example of Second Scenario

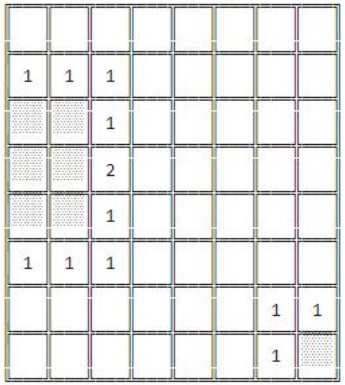
8 x 8 Field with 3 Mines



The player is thinking

Player hasn't discovered any new mine
Player thinks that {(5, 2)} might be safe

checking all the safe positions and flagging mines



The player is thinking

Player has discovered that {(4, 1), (7, 7), (2, 1)} have mines
Player knows that {(3, 0), (2, 0), (3, 1), (4, 0)} are safe

The player has found all the mines

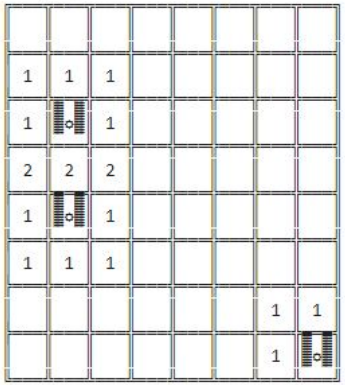


Figure 6: Example with a 8x8 grid and 3 mines.

Demo

1 Introduction

2 Propositional Logic

3 Implementation

4 Conclusion

Conclusion

In this project we wanted to use the tools of propositional logic to solve a popular video game, Minesweeper. This project was very useful since it allowed us to see what we studied from theory applied in practice. It allowed us to learn the use of some commonly used libraries in programming with logic and to understand how these kinds of techniques can be used as an alternative to standard algorithm.

Thanks!