2006

A Constraint-Based Approach to Solving Minesweeper

Ken Bayer
University of Nebraska-Lincoln, kbayer@cse.unl.edu

Josh Snyder
University of Nebraska-Lincoln, jsnyde@cse.unl.edu

Berthe Y. Choueiry
University of Nebraska-Lincoln, choueiry@cse.unl.edu

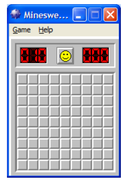
A Constraint-Based Approach to Solving Minesweeper

Ken Bayer, Josh Snyder, and Berthe Y. Choueiry
Constraint Systems Laboratory • University of Nebraska-Lincoln

1. Minesweeper

Relentless Logic

Minesweeper, circa 2000



Minesweeper in Windows Vista

Minesweeper is a game of logic. It originated from 'Relentless Logic,' which was written by Conway, Hong, and Smith around 1985. In Relentless Logic, the player is a soldier trying to crawl back to the Command Center, avoiding mines. The player knows only the number of mines adjacent to his/her current position.

The modern form of Minesweeper was developed by Donner and was released with Windows in 1989. The player can click on any square to reveal it. If the square has a mine on it, the player loses. If it doesn't have a mine, the square is replaced with a number indicating how many adjacent squares are mined. Using this information, the player tries to mark all of the mines on the board.

Recently, Kaye showed that determining whether a Minesweeper configuration is consistent is **NP**-Complete [1].

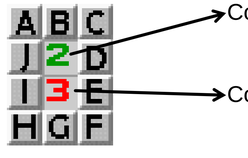
2. Our Goals

- Motivate the students for the study of Constraint Processing (CP). Minesweeper is perfect to this end because it allows us to illustrate the use of CP algorithms in a familiar context and show how they operate.
- Understand and demystify humans' fascination with puzzles.
- Discourage graduate students from losing too much time playing the game by making a program that plays the game for them.

3. Our Approach

We model Minesweeper as a Constraint Satisfaction Problem (CSP) and explore the application of constraint propagation techniques to interactively determine safe and mined squares.

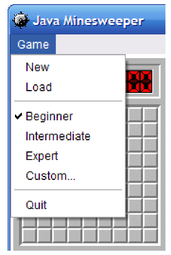
- Every square is a **variable** with two possible values: **safe** or **mined**.
- Every safe square yields a '**sum constraint**' over its 8 neighbors. For example, a square labeled **3** yields a constraint stating the square be surrounded by 3 mines.

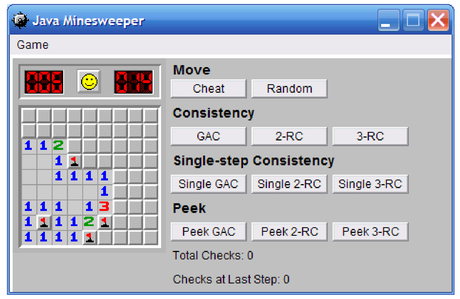


Constraint C_1: Scope = {A,B,C,D,E,I,J}
Exactly 2 neighboring squares must have mines

Constraint C_2: Scope = {D,E,F,G,H,I,J}
Exactly 3 neighboring squares must have mines

4. The Application



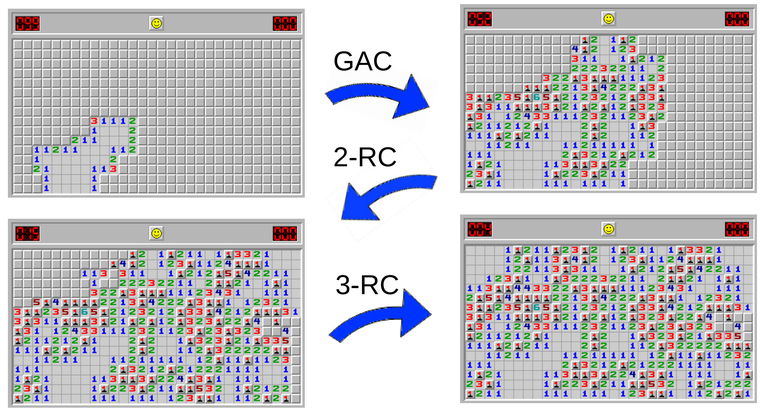


We use the same rules as the Windows version of Minesweeper. The player can:

- Choose a pre-defined level of difficulty or specify the board size and number of mines.
- Load a predefined game stored in an xml file.
- Trigger constraint propagation at 3 consistency levels:
  - GAC
  - 2-relational consistency, and
  - 3-relational consistency.

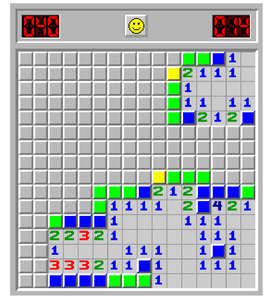
  We project the generated higher-arity constraints on the domains, but do not save those generated constraints in order to save on memory space.
- Execute each consistency algorithm to proceed either step-by-step or to run in a loop until quiescence.

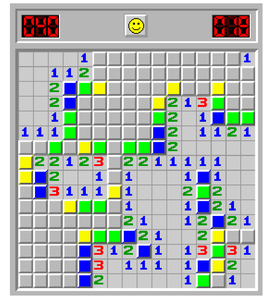
5. Constraint Propagation



GAC, 2-RC, and 3-RC are of increasing complexity: GAC ensures the consistency of each single constraint, 2-RC (respectively 3-RC) ensures the consistency of every combination of 2 (respectively 3) constraints with overlapping scopes. Higher levels of consistency are more costly, but can infer more information. Given the computational cost, one always applies GAC first, then 2-RC, followed by 3-RC.

6. Previewing Propagation

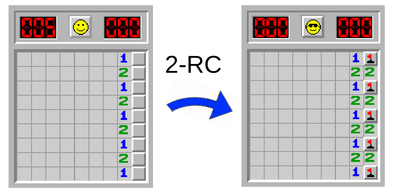




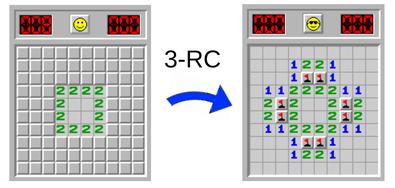
To illustrate the effects of the different levels of consistency, 'Peek' buttons show the user, using a color code, the squares whose 'state' can be determined by each level of consistency propagation without actually flagging them to reveal them.

We use **blue for GAC**, **green for 2-RC**, and **yellow for 3-RC**.

7. Interesting Configurations



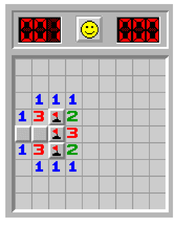
This configuration illustrates a situation where GAC is unable to filter any values; we must look at pairs of constraints (2-RC) to solve this puzzle.



Another interesting configuration is the circle of 2's: neither GAC nor 2-RC yields any filtering.

3-RC is necessary to solve this puzzle!

While increasing the level of consistency allows one to eventually find *all possible solutions* to a given configuration of a Minesweeper instance, constraint propagation cannot guarantee that the player will win every game…



… because of situations such as the one pictured here where two possible solutions exist.

Available online at consystlab.unl.edu/our_work/minesweeper.html

University of Nebraska Lincoln

This research was supported by CAREER Award #0133568 from the National Science Foundation.

References

1. Kaye, R.: Minesweeper is NP-complete. Mathematical Intelligencer 22.