An Interactive Constraint-Based Approach to Sudoku

Christopher G. Reeson[1] and **Kai-Chen Huang**[2] and **Kenneth M. Bayer**[1] and **Berthe Y. Choueiry**[1,2]
[1] Constraint Systems Laboratory, Department of Computer Science & Engineering, University of Nebraska-Lincoln
[2] Information Sciences Institute, University of Southern California
creeson@cse.unl.edu, kaichenh@usc.edu, kbayer@cse.unl.edu, choueiry@cse.unl.edu

Introduction and Goals

We present SOLVER, a Java applet that uses Constraint Processing (CP) techniques to assist human users in solving Sudoku puzzles. We also showcase CONSTRUCTOR, another applet that allows users to enter and store puzzles, which they can then load in SOLVER. These applets are available from sudoku.unl.edu/Solver and sudoku.unl.edu/Constructor. Our goals are as follows:

- *To the public:* Illustrate the power of CP techniques in the context of a popular easily approachable puzzle.
- *For the research:* Investigate how to use CP to interactively support and guide human players.
- *For education:* Use the Sudoku puzzle as a means to teach basic and advanced constraint propagation techniques.
- *For insight in human reasoning:* and understand how it differs from algorithmic approaches.

SOLVER is built to maximize the interactions between the human users and CP techniques. It allows users to apply various consistency algorithms to specific constraints or the entire puzzle, and interactively make assignments and domain reductions. We also designed a 'hint' functionality that uses increasingly complex propagation algorithms in a controlled manner to guide and train the users to play the game.

Background

The Sudoku Puzzle, originally named Number Place, was created by Howard Garns in 1979, and originally appeared in the Dell Pencil Puzzles and Word Games magazine. Nikoli began publishing Sudoku Puzzles in 1986 and introduced the Sudoku name, trademarked in Japan. More recently, newspapers across the United States have begun publishing puzzles daily. The game has become immensely popular and many web sites are devoted to this puzzle[1].

Several programs have been developed and released on the Internet that allow users to solve Sudoku puzzles. These implementations vary widely. Some programs provide no support to the player[2]. Others offer suggestions using a set of rules loosely based on logic[3]. Finally, others resort to backtrack search when their rules are not able to solve the puzzle[4]. Those rules are often 'tricks' resulting from generalizing observations made while solving puzzles. They are often based on convoluted patterns and have strange names, such as naked singles, cross-hatching, locked candidates, etc. (van der Werf 2007). They are not grounded in CP techniques and, often times, a number of seemingly distinct rules are subsumed by a single constraint propagation technique.

On the theoretical side, Yato has shown that Sudoku is NP-Complete by reduction from the Latin Square (Yato 2003). Simonis (2005) studied various models of the constraints in the basic 9×9 Sudoku puzzle and their associated propagation algorithms (often based on finding matchings in bipartite graphs) . He compared their ability to solve puzzles of various difficulties. Our approach differs from Simonis's in that we restrict ourselves to the most basic model of the constraints, and use only the most common constraint propagation algorithms (a subset of those used by Simonis). Unlike Simonis, we focus on the interactive aspects of the game, and have designed a 'hint' functionality that uses constraint propagation mechanisms of increasing complexity to guide the human players and train them in playing the game.

Constraint Modeling and Solving

Many variations of the Sudoku puzzle exist. While we plan to extend our interfaces to handle about 10 puzzle variations, our current implementation handles only the basic puzzle. This puzzle is a 9×9 grid, where some cells in the grid are filled with numbers in [1..9]. The task is to complete the grid by placing a number from 1 to 9 in each of the empty cells, while ensuring that a number appears exactly once in every column, row, or the nine 3×3 'blocks' that form the grid. A well-posed Sudoku has exactly one solution (Simonis 2005). Naturally, since the problem has a fixed size, it can be solved in fixed time (there are at most 9^{81} combinations).

The model of the Sudoku as a Constraint Satisfaction Problem (CSP) is straightforward. We use two representations: the binary and the non-binary models. In the former, we have 810 binary 'all-different' constraints, one constraint between any two cells that cannot take the same value. In the

[1] www.sudopedia.org, www.sudocue.net/guide.php
[2] www.websudoku.com, www.sudoweb.com
[3] www.sudokusolver.co.uk,www.sudocue.net
[4] www.techfinesse.com/game/sudoku_solver.php

latter, we have 27 'all-different' constraints, each of arity 9, defined over the rows, columns, and blocks.

We used the following common propagation algorithms. (References not provided for lack of space.) On the binary constraints: Arc Consistency (AC) and Singleton Arc-Consistency (SAC). On the non-binary constraints: Generalized Arc-Consistency (GAC) and Singleton GAC (SGAC). We filter the domains of the CSP variables interactively as the user places values in the cells with Forward Checking (FC) and Maintaining Arc-Consistency (MAC). Finally, a depth first backtrack search displays all solutions or gives their number on demand.

Interface

The interface allows the user to switch between SOLVER and CONSTRUCTOR. SOLVER is shown in Figure 1. The 'Load Instance' button (top left) allows the user to load any puzzle stored in the library. CONSTRUCTOR (not pictured) allows users to expand the library by entering and storing new instances, and modifying existing ones. The main functionalities provided by the interface are as follows. The buttons along the top and left side of the grid allow the users to enforce AC (binary) and GAC (non-binary) over the rows and columns. The buttons on the upper right corner allow the users to apply AC and GAC on the blocks. Below these buttons, the name of the loaded instance is displayed (here, 'Base-chris1'), and 'Reset Problem' allows the users to reload the problem from the library. The 'Undo' and 'Redo' buttons support every functionality of the interactive solver, thus providing a fully interactive experience to the users. 'Assign Singletons' allows the users to instantiate, at any point, all variables whose domains have a single value, thus reducing exertion. '#Solutions' gives the number of solutions for the grid being displayed. The 'Interactive Grid' panel below the grid gives the users the following control options. (1) 'Appearance' is a set of functionalities to control the appearance of the grid (showing and hiding domains, assigning values to cells, and removing possible values from domains). (2) 'Propagation' is a set of buttons to enforce select propagation algorithms on the entire grid at any point in time. And (3) 'Interactive Search' supports the users solving Sudoku one cell at a time. 'Interactive Search' allows the users to select a form of propagation to be applied after every assignment, thus monitoring the consistency of every selection and highlighting errors. The increasingly 'aggressive' choices are none, back checking (BC), forward checking (FC), and maintain arc consistency (MAC). Finally, the 'Hint' panel allows users to request hints.

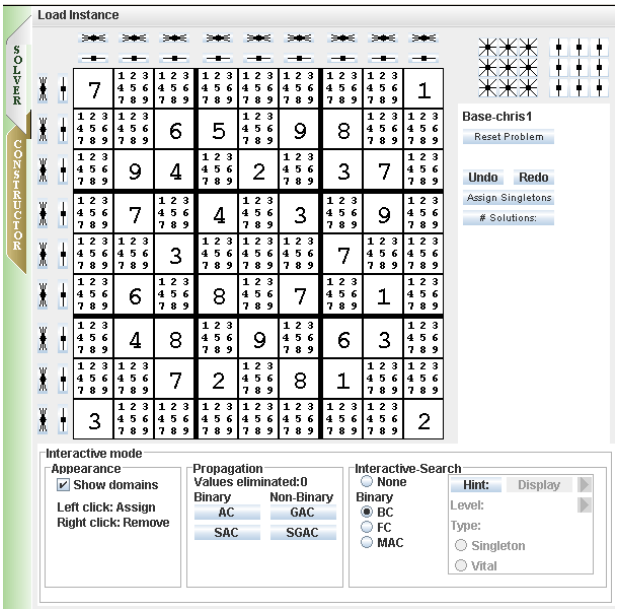

Figure 1: Interface of SOLVER.

This functionality is the most interesting one in our system and is *significantly more powerful than the corresponding one in the online player of the NY Times*. It organizes the hints depending on (1) the level of consistency required for guessing the hint, and (2) the condition of the domain of a cell. The first condition checks if a cell has a single value (Singleton); and the second condition checks if a value does not appear in the domain of any other variable in the same row, column, or block (Vital). Given the current state of the grid, the 'Hint' functionality checks for these two conditions as enforced by the following levels of consistency in increasing complexity order: FC, AC, GAC on each individual constraint, GAC on all constraints, SAC on each individual variable, SAC on all constraints, SGAC on each individual constraint, SGAC on all constraints. Starting from the simplest consistency level (FC), 'Hint' records the total number of cells that exhibit a given condition at the current consistency level, and displays this number to the users. Users can switch between the two conditions (Singleton and Vital), and visit each of the hints as they are highlighted on the grid. Users can then move to the next level of consistency, thus gradually sharpening their abilities to solve the puzzle.

Conclusion

SOLVER allows users to observe the effects of constraint propagation on a familiar puzzle, while allowing them to manually make assignments and filter domains. The 'Hint' functionality provides human players with help that is solidly grounded in CP techniques. The combination of a consistency level and hint condition guides users' attention and improves their understanding of the game and the underlying CP concepts. We plan to extend the approach to other variations of the Sudoku puzzle.

Acknowledgments: Undergraduate Creative Activities and Research Experiences (UCARE) grant of the University of Nebraska-Lincoln and NSF CAREER Award #0133568.

References

Simonis, H. 2005. Sudoku as a Constraint Problem. In *Working notes of the CP 2005 Workshop on Modeling and Reformulating Constraint Satisfaction Problems*, 13–27.

van der Werf, R. 2007. SudoCue–SudokuSolving Guide. www.sudocue.net/guide.php.

Yato, T. 2003. Complexity and Completeness of Finding Another Solution and its Application to Puzzles. Master's thesis, University of Tokyo.