A SAT attack on Killer Sudokus

Shuai Wang Aashish Venkatesh

University of Amsterdam

Abstract

Killer sudoku is a special class of sudoku where the sum of some adjacent cells is given. The goal is to fill in every cell with 1 to 9 and make sum equals to the sum as specified. This case study presents the first attempt using SAT solver for killer sudoku problems. This approach follows from that of classical sudokus. In addition, the first opensource killer sudoku database was generated.

1 Introduction

Sudoku is a Japanese game since 1986. In Japanese, "Sudoku" means "single number" [3]. A Sudoku Problem is a problem to fill in a n*n boards with numbers from 1-9. In this case study, we will only consider boards of 9*9. The filled board should have the first three of following constrains satisfied [2]. A killer sudoku differs from classical sudokus since it doesn't have pre-filled cells. Instead, its constraints are about the sum of numbers in cells (constraint 1-2 and 4). A set of such cells are named *cage*.

1. There is exactly one number in each cell.
2. Each number must appear exactly once in each row / column / 3*3 block (a.k.a. nonet).
3. Some numbers are pre-filled and we need to consider about them
4. **The sum of pairwise distinct numbers in cells of the same cage equals to that as labelled on the cage (i.e. the *label* of the cage)**

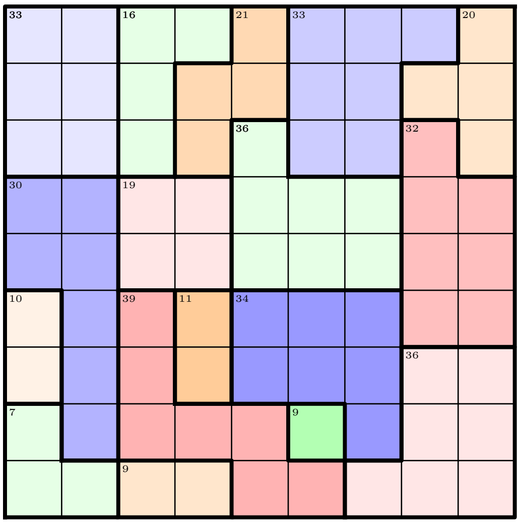

Figure 1: A Killer Sudoku Puzzle

2 A SAT attack on Killer Sudokus

A SAT approach of a sudoku problem is to translate constraints into equi-satisfiable propositional formulas in Conjunctive Normal Form (CNF) and obtain the result by interacting with a *SAT solver*, a program to solve a satisfiability problem in propositional logic [2]. By introducing for each cell on column i and row j and a possible number k a proposition $p^k_{i,j}$, we can encode the constraints as follows respectively:

1. $\bigwedge_{i=1}^{9} \bigwedge_{j=1}^{9} \bigwedge_{k_1,k_2=1}^{9} (\neg p^{k_1}_{i,j} \vee \neg p^{k_2}_{i,j})$, $k_1 \neq k_2$.
2. $\bigwedge (\neg p^{k_1}_{i_1,j_1} \vee \neg p^{k_2}_{i_2,j_2})$, for two different cells at (i_1, j_1) and (i_2, j_2) in the same rows, columns and 3*3 blocks (i.e. nonets), $1 \leq k_1, k_2 \leq 9$.
3. set $p^k_{i,j}$ to *True* if a cell at (i, j) is labelled with k.

4. For each possible combination $pos = \{b_0, b_1, \ldots, b_n\} \in POS$ regarding a cage C labelled s (with its cells named $c_0, c_1, \ldots, c_n \in C$), where $sum(b_i) = s$, we introduce a new proposition x_i for each pos and y_i for each number b_i.

   (a) y_i iff i was labelled on one of the cells in the cage.

   (b) x iff every $y_i \in b_i$ if true.

   (c) $\bigvee x$ for each x corresponding to a possible combination $pos \in POS$

2.1 Implementation and Evaluation

This project employed PycoSAT as the SAT solver [1]. By interacting with its Python API, it took less than one second to obtain a solution for any killer sudoku problem. In addition, the total time and the solving time (of PicoSAT) were recorded and the average time for each sudoku corresponding to a specified cage size is presented as in Table 1. It is clear that the more possibilities regarding a cage and its label, the harder it is to solve. In addition, we generated a database for the purpose of evaluation. The database consists of 1000 example sudokus and its answers for a maximum cage size of 2 to 9 each. The database and sourcecode are available on the project page[1].

Table 1: Evaluation Results using PicoSAT

Max. Cage Size	2	3	4	5	6	7	8	9
Total Time	0.3369	0.3422	0.3542	0.3609	0.3845	0.3619	0.3726	0.3764
Solving Time	0.0595	0.0629	0.0736	0.0823	0.1027	0.0874	0.0910	0.0895
Possible Labels	15	19	21	21	19	15	9	1
Possible Combinations	36	84	127	127	87	54	9	1

3 Conclusion and Acknowledgement

This paper presents a case study using a SAT solver solving killer sudoku problems by encoding into propositional constraints. Both the database and the source code are opensource. The authors would like to thank Prof. Frank van Harmelen for his guidance and inspiration.

References

[1] Armin Biere. PicoSAT essentials. *Journal on Satisfiability, Boolean Modeling and Computation*, 4:75–97, 2008.

[2] Inês Lynce and Joël Ouaknine. Sudoku as a SAT problem. In *International Symposium on Artificial Intelligence and Mathematics, ISAIM 2006, Fort Lauderdale, Florida, USA, January 4-6, 2006*, 2006.

[3] Tjark Weber. A SAT-based sudoku solver. In *12 th International Conference on Logic for Programming, Artificial Intelligence and Reasoning, LPAR 2005*, pages 11–15, 2005.

[1] `https://uva-kr16.github.io/KilerSudoku/`