SOLVING PUZZLES (EG. SUDOKU) AND OTHER COMBINATORIAL PROBLEMS WITH SAT

GAO SHAN[1] AND ROLAND YAP[2]

School of Computing, National University of Singapore
Computing 1, Law Link, Singapore 117590

ABSTRACT

There are many combinatorial puzzles around the world. Solving these puzzles is an intellectual challenge to human kinds. In computing, one way of solving these combinatorial puzzles is by modeling them as a satisfiablility problem (SAT). The SAT problem is to find given a propositional logic formula an assignment of the variables which makes the formula true. In this project, we use Sudoku, the Japan originated combinatorial puzzle, as the "play yard" for experiment on SAT modeling and current SAT solver technology evaluation. In this paper, we demonstrate the relationship between human logic rule and CNF resolution. As a result, SAT problem can be used to find practical rule in combinatorial problems. In the rest of paper, higher dimension Sudoku and Sudoku variations are modeled and examined using existing SAT solvers. At the end, basic SAT algorithms are used for examine the efficiency of different modeling of a given problem.

1 INTRODUCTION

A SAT problem is to find an assignment of Boolean variables making a given propositional logic formula true. If such an assignment can be found, this problem is said to be satisfiable, otherwise unsatisfiable.

For industry users, SAT problem is more or less a black box problem. Users supply CNF to well-designed SAT solvers, and SAT solvers produce the result. However, we suspect, some knowledge of the underlying SAT algorithm will be helpful to produce more efficient CNF formula. In the sense the minimal CNF may not be the most efficient formula, and some extra clauses maybe helpful.

In this project, we are interested in evaluating the existing SAT techniques. Resolution is the mathematical approach of solving SAT problems which states that two clauses having the form (x ∨ C) and (¬x ∨ D) can produce another clause (C ∨ D). Because the space inefficient of resolution most of modern SAT solvers are based on basic backtrack algorithm. A good approach to evaluate the existing SAT algorithms is to use some interesting puzzles as the tool. Sudoku is a good candidate for this purpose. In this project, Sudoku and its variations are used to examine the SAT modeling and SAT solving techniques.

---

[1] Student

[2] Supervisor

1.1 Sudoku Puzzles

A *Sudoku puzzle* is represented by a 9x9 grid, which comprises of nine 3x3 sub-grids (*boxes*). Some of the entries have a given number ranging from 1 to 9, while other left bank. These entries are called *givens*. The goal of solving this puzzle is to assign numbers from 1 to 9 to all the blank entries making every row, every column, and every box contains exactly the nine numbers 1 to 9. The solution of a Sudoku puzzle is usually referred as *Sudoku Square.* Every three boxes in a row are called a *band*, and every three boxes in a column are called a *stack*. Figure 1 and Figure 2 is one example of Sudoku Square and its puzzle:

7	2	6	4	9	3	8	1	5
3	1	5	7	2	8	9	4	6
4	8	9	6	5	1	2	3	7
8	5	2	1	4	7	6	9	3
6	7	3	9	8	5	1	2	4
9	4	1	3	6	2	7	5	8
1	9	4	8	3	6	5	7	2
5	6	7	2	1	4	3	8	9
2	3	8	5	7	9	4	6	1

Figure 1. Example Sudoku Sqaure

	2	6				8	1	
3			7		8			6
4				5				7
	5		1		7		9	
		3	9		5	1		
	4		3		2		5	
1				3				2
5			2		4			9
	3	8				4	6	

Firgure 2. Example Sudoku Puzzle

A good Sudoku puzzle is defined as the puzzles that 1) have only one solution 2) can be solved only using reasoning, i.e. no search. However, the gap between reasoning and search for human solvers is weakly defined, that some of the solvers are skillful in search, and some of the "hard" puzzles are more probably solved by search even though there some "logic" that can avoid search. The hardness of Sudoku puzzles are usually referred as the number of human solving techniques used.

A typical Sudoku puzzle has 3^2*3^2 = 3^4 grids. These puzzles are usually referred as base-3 or dimension-3. A base-4 or base-n Sudoku puzzle meaning these puzzles have 4^4 or n^4 grids. The base-3 Sudoku puzzles are trivial, but the time complexity for general puzzles with base n is proven to be NP-complete. There are also some variations of Sudoku puzzle, such as Diagonal Sudoku, Even-Odd Sudoku, and Greater Than Sudoku. They will be modeled in later section.

2 SUDOKU MODELING

2.1 Minimal coding and extended coding

In Lynce and Ouaknine's [3] proposed two CNF encoding for Sudoku modeling. These encodings are used as the starting point of modeling Sudoku and its variation in this paper.

In this Sudoku modeling, 729 variables are used to represent 9 possible values of the 81 entries. Sxyz represents whether the row x, column y is assign to the number z. Besides these encoding, givens are appended as unit clauses. These are the proposed encodings:

The minimum encoding:

There is at least one number in each entry:

$$\bigwedge_{x=1}^{9}\bigwedge_{y=1}^{9}\bigvee_{z=1}^{9} s_{xyz}$$

Each number appears at most once in each row:

$$\bigwedge_{y=1}^{9}\bigwedge_{z=1}^{9}\bigwedge_{x=1}^{8}\bigwedge_{i=x+1}^{9}(\neg s_{xyz} \vee \neg s_{iyz})$$

Each number appears at most once in each column:

$$\bigwedge_{x=1}^{9}\bigwedge_{z=1}^{9}\bigwedge_{y=1}^{8}\bigwedge_{i=y+1}^{9}(\neg s_{xyz} \vee \neg s_{xiz})$$

Each number appears at most once in each 3x3 sub-grid:

$$\bigwedge_{z=1}^{9}\bigwedge_{i=0}^{2}\bigwedge_{j=0}^{2}\bigwedge_{x=1}^{3}\bigwedge_{y=1}^{3}\bigwedge_{k=y+1}^{3}(\neg s_{(3i+x)(3j+y)z} \vee \neg s_{(3i+x)(3j+k)z})$$
$$\bigwedge_{z=1}^{9}\bigwedge_{i=0}^{2}\bigwedge_{j=0}^{2}\bigwedge_{x=1}^{3}\bigwedge_{y=1}^{3}\bigwedge_{k=x+1}^{3}\bigwedge_{l=1}^{3}(\neg s_{(3i+x)(3j+y)z} \vee \neg s_{(3i+k)(3j+l)z}).$$

This minimum encoding consists of 8829 clauses consist of 81 nine-ary clauses and 8748 binary clauses.

In addition to the minimum encoding, the **extended encoding** has:

There is at most one number in each entry:

$$\bigwedge_{x=1}^{9}\bigwedge_{y=1}^{9}\bigwedge_{z=1}^{8}\bigwedge_{i=z+1}^{9}(\neg s_{xyz} \vee \neg s_{xyi})$$

Each number appears at least once in each row:

$$\bigwedge_{y=1}^{9}\bigwedge_{z=1}^{9}\bigvee_{x=1}^{9} s_{xyz}$$

Each number appears at least once in each column:

$$\bigwedge_{x=1}^{9}\bigwedge_{z=1}^{9}\bigvee_{y=1}^{9} s_{xyz}$$

Each number appears at least once in each 3x3 sub-grid:

$$\bigvee_{z=1}^{9}\bigwedge_{i=0}^{2}\bigwedge_{j=0}^{2}\bigwedge_{x=1}^{3}\bigwedge_{y=1}^{3} s_{(3i+x)(3j+y)z}.$$

This extended encoding (include minimal encoding) has 11988 clauses consist 324 nine-ary clauses, and 11664 binary clauses.

I suspect they made a mistake in the minimal encoding. The CNF in extended encoding (except minimal encoding) has only 3159 clauses, and the CNF in the minimal encoding has 8829 clauses. Experiment shows that the 3159-clause encoding is suffice to represent Sudoku characteristics. This 3159-clauses encoding is indeed the minimal encoding for Sudoku Puzzles.

The 8829-clause encoding and 3159-clause encoding can express Sudoku characteristics. They are logically equivalent. As the data indicates, the 3159-clause encoding has a larger portion of nine-ary clauses than the 8829-encoding. It is suspected to be less efficient than the 8829-clause encoding. This will be proved in the experiment in later chapter.

Beside the 8829-clasue encoding and 3159-clause encoding (with 3-most/1-least and 1-most/3-least conditions), another question is whether a CNF that consist only 4 "at least" conditions is suffice to represent the Sudoku characteristics. If it is sufficient, it will only contain the 324 nine-ary clauses. The experiment turn out this encoding is not sufficient, and the result show under this CNF modeling, one grid can have more than one number, which is restricted in Sudoku rule. On the other hand, the 11664-encoding is not sufficient too, because 4 "at most" conditions allow some grids to be empty.

2.2 Human strategy and CNF modeling

There are many hand solving techniques around. Essentially, they are formed using logics that can be easily applied by human. Some of these techniques are straight forward (Naked Single, Hidden Single, Naked Subset, Hidden Subset). However, there exist some "hard" puzzles, that after applying those simple logic rules, most of their grids still remain blank. Thus, some sophisticated reasoning techniques are introduced. These techniques include X-wings, Swordfish, XY-wing, and Coloring. At the end, if all of these techniques failed, search come to the place, and some Sudoku masters refer it as Nishio. In this paper, we code the SAT algorithm to track the logic tree of unit propagation and resolution. The result can be used to validate existing human solving techniques and propose new techniques from observation.

The easiest techniques probably are the Naked Single and Hidden Single. The Naked Single rule states that grids that have only one candidate left should be assigned that value. Hidden Single rule states that if a grid is the only possible grid to assign a value in a row/column/box, this grid should be assigned this value. In SAT algorithm, unit propagation will reduce clauses to unit clause which only has one literal. This is the process of Naked Single and Hidden Single rule.

Beside these straight forward rules. Naked Pairs is a commonly used technique. As shown in Figure 3, the left most two girds have only the candidate 2 and 8. Naked Pairs states that 2 and 8 candidates in other girds should be omitted since 2 and 8 are used by these two grids. However, in SAT modeling, "there is at least one number in each entry" clause in the left bottom grids will be reduced to (S112 ∨ S118) and (S212 ∨ S218). The "each number appears at most once in each row" for 2 and 8 are reduced to (¬S112 ∨ ¬S212) ∧ (¬S212 ∨ ¬S511) ∧ (¬S112 ∨ ¬S511) and (¬S118 ∨ ¬S218). (S112 ∨ S118) and (¬S112 ∨ ¬S511) entails (S118 ∨ ¬S511). (S212 ∨ S218) and (¬S212 ∨ ¬S511) entails (S218 ∨ ¬S511). And finally, (S118 ∨ ¬S511), (S218 ∨ ¬S511), and (¬S118 ∨ ¬S218) entails ¬S511. This is the logic foundation of Naked/Hidden Pairs, Triples, and Quads.

Figure 3 Naked Pairs

Figure 4, X-Wing

Human solvers who can frequently make use of X-wing technique are usually expected as Sudoku experts. X-wing is probably the first technique applies to "hard" puzzles. As shown in Figure 4, the circled four grids are laying on two columns. Since 5 must be assigned to each of the columns, and no column can contain more than one 5, other grids within their rows should omitted 5 from their candidates, e.g. the grid which has candidate 2, 5, and 7 at the moment. Similar to the Naked pairs, the X-wing can be viewed as a set of resolution over a given clause group.

Actually, X-Wing uses one step less resolution than the naked pairs. The reason X-Wing is expected as a higher level technique than the naked pairs is probably because the clauses involved in naked pairs is lock inside same row/column/box. Whereas, the clauses evolved in X-Wing basically come from two different rows, two different columns, and four different boxes. During the tracking process, other rules are found, however, they are usually having too many clauses to participate and this make it hard for human to recognize and apply.

2.3 Greater Than Sudoku

Greater Than Sudoku probably is the most interesting variations of Sudoku. In Greater Than Sudoku, no digits is given, only the relationships between adjacent grids are given. Figure 5 shows two examples of Greater Than Sudoku.

The bonded inequality property of Greater Than Sudoku makes it hard to model. For two adjacent grids A, and B, we propose to model it as (A9 ∧ (B8 ∨ B7 ∨ B6 ∨….)) ∨ (A8 ∧ (B7 ∨ B6 ∨ B5 ∨…)) ∨ ….. Special treatment is made for A9 and B1 cases. After apply De Morgen's law, each pair of adjacent grids will be modeled using 2^6 clauses. Furthermore, the bond of each grid will be tightened by the number of its neighbor, e.g. the central of each box, which has four neighbors, will have four candidates eliminated because of the inequalities.

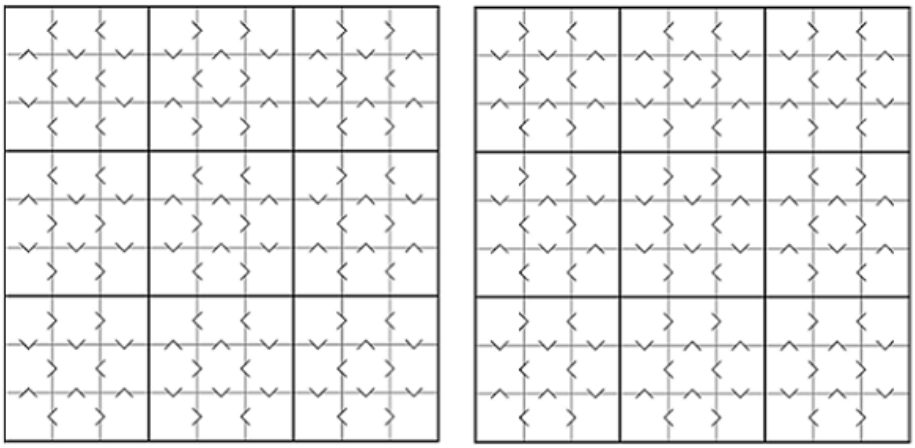

Figure 5. Greater Than Sudoku

3 EXPERIMENT ON SAT ALGORITHMS

We performed an experiment on SAT algorithms. 200 puzzles each under easy, median and hard levels are collected from different sources. The proposed three encodings are applied to construct CNF models. And different configurations of SAT algorithms are used to solve these models. The SAT algorithms used are: 1) Unit Propagation (UP), 2) Unit Propagation + Backtrack Search in 3 levels (UP + 3BS), 3) Unit Propagation + Backtrack Search in 6 levels (UP + 6BS), 4) Unit Propagation + DPLL in 3 levels (UP + 3DPLL), 5) Unit Propagation + DPLL in 6 levels (UP + 6DPLL)

The percentage of puzzles solved under each modeling and SAT algorithm configurations is concluded into Figure 6.

3159-clause	Easy	Median	Hard
UP	7.5%	0.5%	0%
UP+3BS	19.5%	3%	0%
UP+6BS	42.5%	8%	2%
UP+3DPLL	27%	5.5%	1%
UP+6DPLL	65%	16%	2.5%
8829-clause			
UP	93%	55%	0%
UP+3BS	96%	64%	0.5%
UP+6BS	100%	78.5%	7.5%
UP+3DPLL	98.5%	71%	2%
UP+6DPLL	100%	83.5%	10.5%
11664-clause			
UP	100%	87%	22%
UP+3BS	100%	94.5%	31.5%
UP+6BS	100%	97.5%	65%
UP+3DPLL	100%	96%	42.5%
UP+6DPLL	100%	100%	87%

Figure 6 Percentage solved under different configuration combinations

The result is consistent to the hypothesis we proposed. From the result, we can conclude the 3159-clauses encoding has the worst performance. This is because more steps of resolution or search are needed to extract useful information. The human solving techniques can usually be simulated under the 11664-clause encoding and 6-7 steps of resolution. The result shows that, the extended encoding with 6 levels of search can solve a big portion of so-called "hard" puzzles

4 CONCLUSION

In our study, the existing CNF modeling approaches and variations of SAT solver algorithms have been examined.

The first finding is SAT modeling can be used to find practical rule for human to follow for combinatorial problems. For existing combinatorial problems, e.g. Sudoku, existing human solving techniques could be examined under SAT modeling. Beside these, a CNF modeling for Greater Than Sudoku is given.

The experiment on SAT solvers gives us a better sense of the importance of good modeling formula. A well designed redundant formula can largely increase the probability of success under bonded level of search.

5 REFERENCE

[1]. H. Simonis. (2005) Sudoku as a constraint problem. In CP Workshop on Modeling and Reformulating Constraint Satisfaction Problems, pages 13-27. http://www.icparc.ic.ac.uk/-_hs/sudoku.pdf.

[2]. Huang, J. (2007). The effect of restarts on the efficiency of clause learning. In

Proceedings of the 20th International Joint Confer- ence on Artificial Intelligence

(IJCAI), 2318–2323.

[3]. I. Lynce and J. Ouaknine. (2006) Sudoku as a SAT problem. In Proc. of the Ninth International Symposium on Artificial Intelligence and Mathematics.

[4] J.P. Warners, (1996) A linear-time transformation of linear inequalities

into conjunctive normal form, Technical Report CSR9631. CWI, Amsterdam.

[5]. L.BORDEAUX, Y. HAMADI and L.ZHANG,(2006). Propositional Satisfiability and Constraint Programming: A Comparative Survey, ACM Computing Surveys, Vol. 38, No. 4, Article 12,

[6]. Mitchell, D. (2005). A SAT solver primer. EATCS Bulletin 85:112-133. Columns: Logic in Computer Science.

[7]. T. Weber. A SAT-based Sudoku solver. In LPAR-12, Short paper proc., 2005

[8]. The international SAT Competitions web page: http://www.satcompetition.org/

[9]. SAT4J: A satisfiability library for java: http://www.sat4j.org/

[10]. TiniSAT: http://users.rsise.anu.edu.au/~jinbo/tinisat/

[11]. Minimum Soduku: http://people.csse.uwa.edu.au/gordon/sudokumin.php

[12]. "Simple Sudoku" Webpage: http://www.angusj.com/sudoku/hints.php