Sudoku SAT solver

Jiangyue Wang, Zuyu Cai

June 12, 2020

Abstract

Sudoku is a popular logic game, and in the computer field, NP-hard problem is also a problem that has been studied by many people. Based on the NP-hard problem, this paper relates it to the Sudoku problem and designs a Sudoku sat solver based on the particularity of Sudoku requirements. In order to exercise the designer's logical thinking, but also for those who want to quickly get answers to Sudoku questions designed a tool.

1 Introduction

Sudoku is a logical game that many people like to play in today's society. It's a math game that originated in Switzerland in the 18Th century. In this game, the player has to solve a problem in a grid of $n^2 * n^2$. Each large grid is divided into n^4 tables. Where each suitable is divided into sub-blocks of $n^2 * n^2$. In this game, each number can appear only once in each row and each column, and each number can appear only once in each sub-grid (1-9). Fill in a number in each blank area until the entire grid is completed.

By solving Sudoku problems, people can exercise their logical thinking. The application of Sudoku in the computer field can help scholars better understand theoretical computer problems and improve their ability of logical thinking. And can improve the programmer's interest in learning, with a better spirit to learn and solve related problems. By solving the Sudoku problem, many similar problems can be solved and applied in real life. Many researchers have solved Sudoku sat solvers such as 9*9, n*n, etc., but they are not based on NP-hard problems. Based on Sudoku NP-hard problem, this paper intends to design a Sudoku sat solver about $n^2 * n^2$, which can find an effective truth-value assignment tool for variables.

2 Sudoku is a NP-hard problem

If assume that the set of all possible Sudoku problems is l and the size belongs to $n^2 * n^2$, the problem is expressed as l and the grid size is $n^2 * n^2$. There is only one solution for each Sudoku problem, which means that the result of each Sudoku problem can only be right or wrong. So the answer to l could be Yes or No. For problem L, there exists L (x) =x|x$\in$ grid, x satisfies all Sudoku constraints and its output is true. According to the passage, Sudoku problem is a NP-complete problem. In addition, NP-hard $\in$ NP-complete[TT03]. Therefore, for Sudoku problems of $n^2 * n^2$ size, it is NP-hard. Problem L belongs to Sudoku problem and meets the condition of $n^2 * n^2$, so problem L is NP-hard problem.

3 Sudoku problem convert to CNF formula

For an $n^2 * n^2$ NP-hard Sudoku problem, it can be converted to the formula expression of CNF.[RI18] There is only one solution to every Sudoku problem. In order to further solve the Sudoku problem, the constraint conditions can be written and converted into CNF formula [KJ06] according to the particularity of the problem. Where, i is the row of the whole Sudoku, j is the column of the whole Sudoku, and k is the index of the sub-lattice. Its constraint conditions and formulas are as follows:

1. Each square must have a value ranging from 1 to n^2.

$$\forall_{i,j}[cell_{i,j} \in 1, ..., n^2]$$

2. Each row contains all the numbers from 1 to n^2.

$$\forall_i[\cup\forall_i cell_{i,j}) = 1, ..., n^2]$$

3. Each column contains all the numbers from 1 to n^2.

$$\forall_j[\cup\forall_j cell_{i,j}) = 1, ..., n^2]$$

4. Each n^2 cell contains all the numbers from 1 to n^2.

$$\forall_k[(\cup\forall_{i,j\in k} cell_{i,j}) = 1, ..., n^2]$$

For every global Sudoku problem, there are Sudoku sub-lattices. In this case, there is a sub-table of n^2. [LO06]Set each Sudoku to have n^2 rows, denoting them as x, n^2 columns, and denoting them as y and z as the specified numbers. They can be denoted as (x,y,z), and their constraint conditions and formulas are as follows:

1. Each entry has at least one number:

$$\bigwedge_{x=1}^{n^2}\bigwedge_{y=1}^{n^2}\bigwedge_{x=1}^{n^2}(x,y,z)$$

2. Each number appears at most once in each row:

$$\bigwedge_{x=1}^{n^2-1}\bigwedge_{y=1}^{n^2}\bigwedge_{z=1}^{n^2}\bigwedge_{i=x+1}^{n^2}\neg(x,y,z)\vee\neg(i,y,z)$$

3. Each number appears at most once in each column:

$$\bigwedge_{x=1}^{n^2}\bigwedge_{y=1}^{n^2-1}\bigwedge_{z=1}^{n^2}\bigwedge_{j=y+1}^{n^2}\neg(x,y,z)\vee\neg(x,j,z)$$

4. Each number appears at most once in each $n^2 * n^2$ square:

$$\bigwedge_{x=1}^{n^2}\bigwedge_{y=1}^{n^2}\bigwedge_{z=1}^{n^2-1}\bigwedge_{m=z+1}^{n^2}\neg(x,y,z)\vee\neg(x,y,m)$$

4 The realization of Sudoku Sat Solver

4.1 Requirements and Analysis

When the Sudoku sat solver is completed, the following objectives are expected to be achieved.

1. The Sudoku problem is NP-hard, which can solve the $n^2 * n^2$ Sudoku problem.
2. Have reasonable input and output formats.
3. The game should be able to provide solutions to Sudoku puzzles if the user asks.
4. Find a suitable sat interface to help the program run. By filtering and selecting the appropriate interface, the project solver should generate a format to be accepted by the interface.
5. The solver designed in this project should be able to understand the model generated by the selected interface and be able to successfully transform the model into the answer to the given Sudoku problem.
6. It is easy to use and easy to learn.
7. Input and output methods are simple and easy to observe.

Analyze the requirements of the whole design by applying the knowledge of computer theory. By analyzing the particularity of Sudoku, it is connected with the knowledge learned. According to the requirements of Sudoku problem, it can be converted into code that can be applied in the computer, and its constraints can be written. And through the appropriate sat interface it is translated into the code required by the program. Finally, the Sudoku sat solver can be obtained through continuous testing and adjustment.See the Figure 1 is the process of Sudoku sat solver.

Figure 1: The process of Sudoku sat solver.

4.2 Design and Implementation

4.2.1 Choice of programming language:

This project chooses C++ language to write programs. Compared with other languages, C++ has its own advantages, mainly reflected in the following aspects: through the use of C++, object-oriented programming can be realized. In the high-level language, the processing speed is the fastest, most of the game software, the system is written by C++. Second, the C++ language is very flexible and powerful. If the advantage of C is Pointers, the advantage of C++ is performance and class hierarchy design. In addition, C++ is very rigorous, precise and numerically and physiographically defined[Sha95]. The syntax of C++ language is hierarchical and corresponding, and its syntax structure is explicit and explicit. C++ can be embedded in any modern processor, is supported by almost all operating systems, and is very cross-platform.

4.2.2 Interface selection:

In this project, when the sat interface was selected, Mini-Sat[SE09] was selected as the interface. The Mini-Sat has the following advantages: first, it is a minimalist, open source SAT solver that meets the requirements of this project. It was licensed by MIT, and it was released, and its performance was to be believed. Secondly, the Mini-Sat was easy to use and easy to understand. Its documentation is relatively complete and small, suitable for initial use. Mini-Sat is more efficient and has an integrated design that adds non-clause constraints and is easy to modify.[ES06] Finally, Mini-Sat has a clean c++ implementation and API that works well with the language used in this project. Accordingly, the interface for this project will be Mini-Sat.

4.2.3 Program design:

1. Input format:

The Sudoku problem of $n^2 * n^2$ is determined by entering the size of n. Define the puzzle through the header file and enter the Sudoku problem as a matrix. So people can visualize what the input problem is. As shown in the Figure 2:

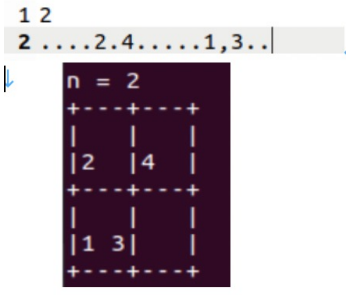

Figure 2: Input format.

2. Main file:

Further coding is done by making sure that the input problem is a Sudoku problem. If it is a Sudoku problem and meets the project requirements, you can write out its constraints. This is then converted through the mini-Sat interface. Get the answer to the Sudoku question and output it.

3. Reading problem:

For programs, reading issues are a concern. In order to facilitate the operator's execution, this project simplifies the way the problem is read. In a program, ". "represents an unfilled number, and", "means there is no gap between two adjacent Numbers. The main function combines all the functions so that they can be implemented together. When entering the problem number, if the last position is the number, an extra ". "should be added.

For example, in the following question, there is a blank space between the number 2 and the number 5, so the unwritten number is represented by the symbol "." Between the Numbers 3 and 4, there is no gap, so we use the symbol ", ".As shown in the Figure 3:

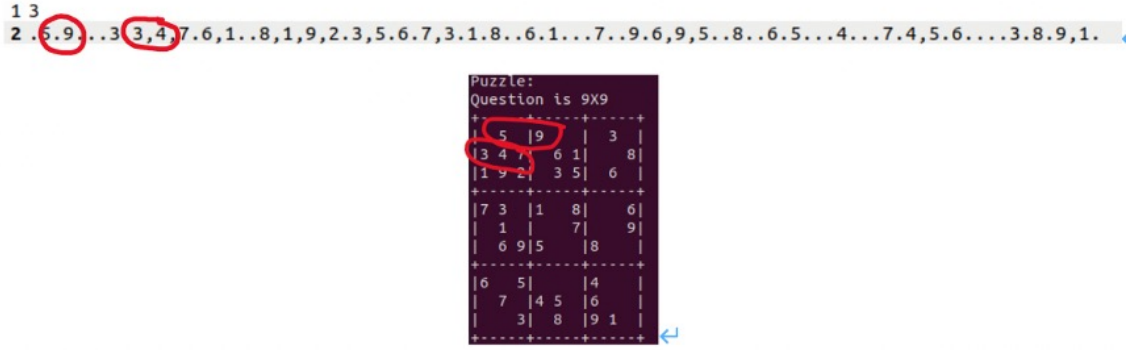

Figure 3: Reading Sample.

4. Compilation method:

By building a file make-file, the console compiles the program with make. This compilation method can be more simplified.

5.Output:

During the execution of the file, obtain the Sudoku solution by using the. Solve Sudoku problem file name. Output the answer to the Sudoku problem in the instruction table in matrix form. As shown in the Figure 4,If there is no output, there is a prompt: no solution and print the wrong question. As shown in the Figure5:

Moreover, To make the data for testing even simpler and more intuitive, the amount of Mini-SAT and the entire Sudoku size was programmed and printed out as the topic was output.As shown in the Figure 6

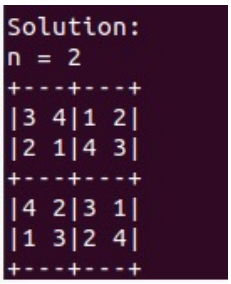

Figure 4: Output Sample.

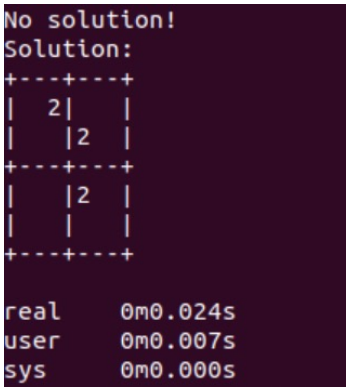

Figure 5: NO Output Sample.

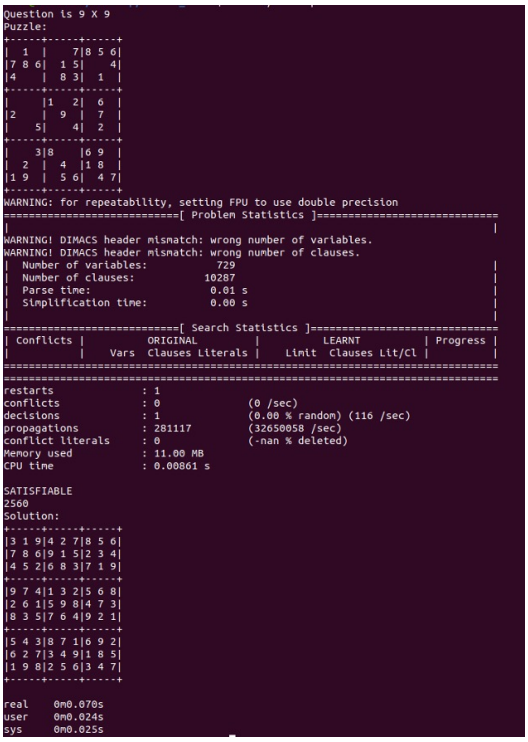

Figure 6: Mini-SAT and time cost output.

4.3 Test

During the detection, based on the existing $n^2 * n^2$ Sudoku problem, the new problem is obtained by increasing or decreasing the numeric value of the input problem. To increase the number of test questions and expand the test scope. According to the existing project schedule, we conducted classification tests on the projects according to different attributes:

1. When the size of the input value n is different, the number of input characters and the running time size of the table: Real time is the elapsed time between the execution of the command line and the termination of the run. User Time is the sum of the execution time of the command in user mode. System time is the sum of the time the command is executed in the system core mentality.As shown in the Figure 7,8,9,10,11:

N=2,

Input number	Real time(s)	User time(s)	System time(s)
4	0.033	0.002	0.003
3	0.019	0.002	0.003
2	0.016	0.002	0.002
1	0.018	0.004	0.001

Figure 7: n=2.

N=3,

Input number	Real time(s)	User time(s)	System time(s)
38	0.046	0.019	0.021
33	0.049	0.013	0.028
27	0.048	0.020	0.020
17	0.046	0.019	0.021
10	0.047	0.014	0.026
3	0.051	0.017	0.026

Figure 8: n=3.

N=4,

Input number	Real time(s)	User time(s)	System time(s)
104	0.526	0.197	0.228
95	0.476	0. 198	0.245
79	0.459	0.212	0.214
54	0.460	0.208	0.223
32	0.490	0.186	0.272
4	0.546	0.278	0.217

Figure 9: n=4.

N=5,

Input number	Real time(s)	User time(s)	System time(s)
265	2.271	0.633	1.575
249	3.571	1.812	1.503
229	3.162	1.434	1.518
218	3.476	1.723	1.551
198	2.991	1.400	1.442
171	3.074	1.441	1.455
144	3.169	1.456	1.517
119	3.163	1.517	1.436
87	3.315	1.657	1.427
69	3.290	1.708	1.388
32	3.222	1.558	1.445
17	3.064	1.381	1.474
8	3.050	1.345	1.486

Figure 10: n=5.

N=6:

Input number	Real time(s)	User time(s)	System time(s)
746	14.439	7.444	6.605
739	14.376	7.592	6.393
708	14.696	7.402	6.6848
616	21.060	13.973	6.611
564	32.473	25.506	6.530
404	15.144	8.407	6.341
308	17.723	10.559	6.572
137	16.542	9.610	6.517
53	36.408	26.084	9.053
25	83.356	71.283	10.152

Figure 11: n=6.

2. By combining the values obtained by the test, the mean values of different n values at different sizes can be obtained as follows, the actual time table obtained when the value of input n is not the same.As shown in the Figure 12:

N	Real time(s)
2	0.0215
3	0.0478
4	0.4928
5	3.1098
6	26.6217

Figure 12: n and real time.

3. The Sudoku solver cost table obtained when the value of input n is not the same.As shown in the Figure 13:

N	Cost(s)
2	0.001591
3	0.00861
4	0.143084
5	1.28014
6	31.2072

Figure 13: n and solver cost.

4. Tables with and without solution: As shown in the Figure 14:

	2(2)	2(3)	2(7)	3(4)	3(8)	3(13)
yes	0.016	0.018	0.019	0.051	0.047	0.046
no	0.025	0.024	0.027	0.067	0.056	0.055

Figure 14: n and solution.

5 Conclusion and Future Development

For Sudoku questions that meet the requirements of the project, this project can get accurate answers. The tool is easy to learn and intuitive to see the results. The results show that the Sudoku problem of $n^2 * n^2$is NP-hard and can be solved by sat solver. The Sudoku sat solver made in this project can run effectively and get results. In addition, the tool is easy to use and easy to understand with a simple interface. The operator can get the answer he wants intuitively.

In addition, some conclusions can be drawn intuitively and clearly by converting the tables obtained from previous tests into charts.

1. According to15,16,17,18,19: It can be known that in a grid of the same size (n is the same), there exists a certain number of input cells, which takes the least time to obtain Sudoku. For the remaining values, to the left of this value, the time decreases as the input problem value increases. On the right side of the value, the time increases with the increase of the value of the input problem.

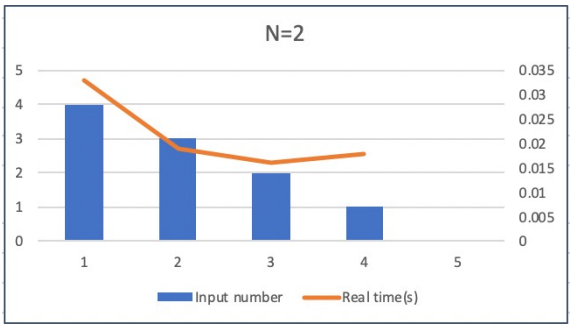

Figure 15: n=2.

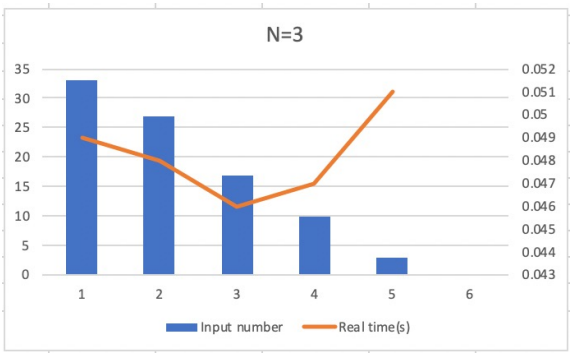

Figure 16: n=3.

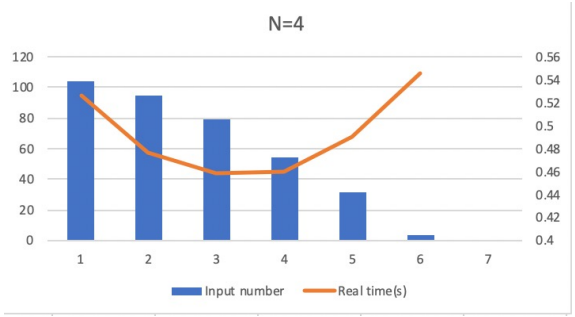

Figure 17: n=4.

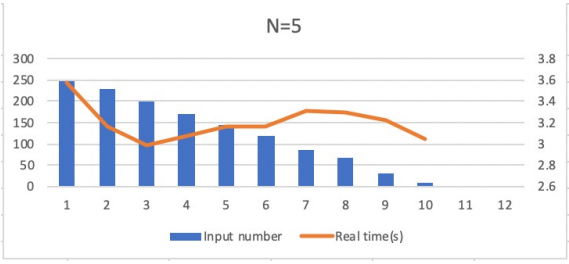

Figure 18: n=5.

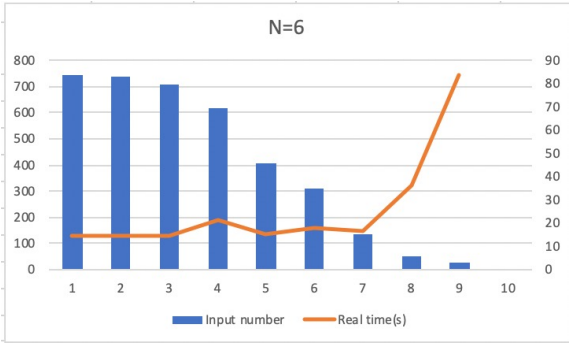

Figure 19: n=6.

2. According to20 ,it can be seen that as the input value of n increases($n^2 * n^2$increase), the more time is needed for the Sudoku to solve.

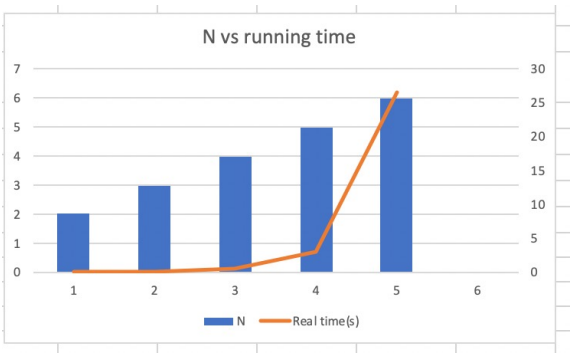

Figure 20: n vs running time.

3. According to21 ,it can be seen that as the input value of n increases($n^2 * n^2$increase), the cost of the Sudoku SAT solver system increases.

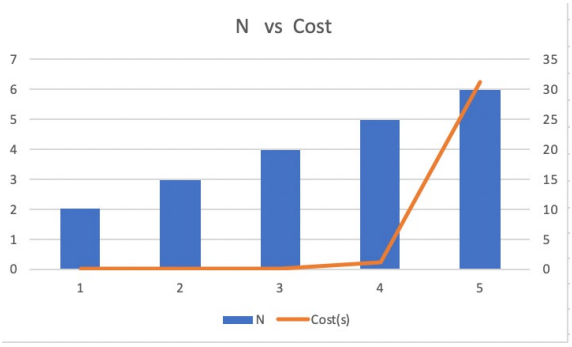

Figure 21: n vs cost.

4. According to22 ,it turns out that Sudoku problems with no solutions take longer to solve than Sudoku problems with solutions.

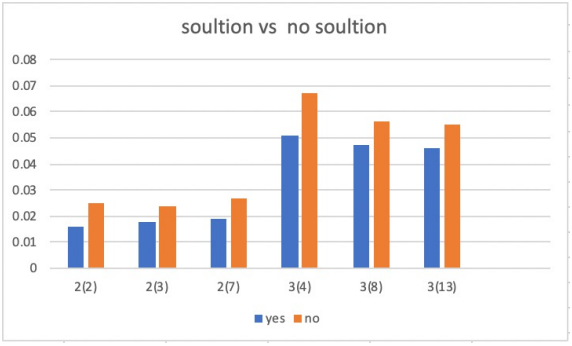

Figure 22: n vs cost.

5.According to23 ,it can be seen that when the input n=7, namely 49*49 Sudoku questions, the system crashes and cannot solve Sudoku questions of this size.

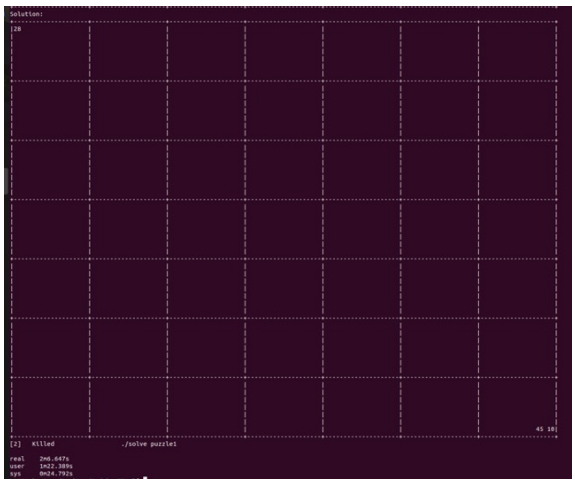

Figure 23: limited n =7 .

Although the $n^2 * n^2$Sudoku sat solver made in this project has been completed, there are still many shortcomings. For some large Sudoku problems, for example, when n=4, due to the need to input a lot of symbols, will lead to some problems in the case of input errors. In future work, you can optimize this. Second, you can try to find some bigger Sudoku problem and test it to see if it works. In addition, in the future work, by using different interfaces to compare the test, and to optimize the program and interface.

References

[ES06] Niklas Een and Niklas Sörensson. Minisat v2. 0 (beta). *Solver description, SAT race 2006*, 2006.

[KJ06] Gihwon Kwon and Himanshu Jain. Optimized cnf encoding for sudoku puzzles. *Proc. 13th International Conference on Logic for Programming Artificial Intelligence and Reasoning (LPAR2006)*, 2006.

[LO06] Inês Lynce and Joël Ouaknine. Sudoku as a sat problem. *ISAIM*, 2006.

[RI18] N. S. Chaudhari Rai, Deepika and Maya Ingle. Polynomial 3-sat reduction of sudoku puzzle. *International Journal of Advanced Research in Computer Science 9.3*, 194, 2018.

[SE09] Niklas Sörensson and Niklas Eén. MiniSat 2.1 and MiniSat++ 1.0—SAT race 2008 editions. *SAT*, 31, 2009.

[Sha95] Namir Clement Shammas. *C/C++ Mathematical algorithms for scientists and engineers.* McGraw-Hill, Inc., 1995.

[TT03] Yato Takayuki and Seta Takahiro. Complexity and Completeness of Finding Another Solution and Its Application to Puzzles). 2003.