Recursive Fitness Algorithm for Generating and Solving Sudoku

Sparsh Arora
Newgen Software Technologies, D 162, Okhla -1,
New Delhi-110020

ABSTRACT

This paper presents the modification in the Brute Force algorithm for generating and solving common Sudoku problem. Although a number of procedures exist for solving Sudoku and some papers also follow heuristic approach. In this paper the true capability of computer system known as recursion and comparison is explored to provide better results than Brute force Algorithm. Recursive fitness algorithm works with a condition or test applied, known in our context as the Fitness test. This algorithm shows a good value in terms of logic used in comparison to the brute force algorithm.

KEYWORDS

Sudoku, Brute Force, Recursion, Fitness Test, Algorithm.

1. INTRODUCTION

Since the first formation of Sudoku in 1979 world has seen the rapid growth in its popularity. Sudoku logic is very simple to understand but not simple to solve. It requires practice and dedication to solve Sudoku [1]. Sudoku is a logical number placement puzzle.

It consists of a grid of 9X9 squares. 9X9 squares are further divided into sub squares of 3X3. These sub squares are also known as boxes, blocks, regions or subsections. Logic behind solving Sudoku is that one number should appear only once in the row, column and 3X3 sub square [2]. Puzzle setter provides some of the grids filled and some empty depending on the complexity of problem. Depending on the complexity of the problem the Sudoku may have one or many solutions. [3]

There are multiple algorithms present that can solve and generate Sudoku problems. Section 2: Literature review section of this paper provides a brief description on brute force algorithm. Section 3 puts some light on the previous work in this field. Section 4 describes the functions and pseudo code of the Recursive Fitness algorithm. Section 5 of this paper shows the execution result of the algorithm.

1.1 History

Throughout the history Sudoku relates many number puzzles. In 1983, 9X9 Latin squares that was invented by Euler is the super-set of the Sudoku solutions and is similar to 9X9 magic squares solution set that was invented in beginning of 10th century AD. [4]

Dell Magazine published the first modern puzzle named as Number Place. Howard Grans, an American Architect invented it, and after being published by Nikoli that was the first Japanese Puzzle Magazine, it gained popularity in Japan. Sudoku refers to “Su-ji wa dokushin ni kagiru" which means “the number must be single” in Japanese. [1]

In 2005, Sudoku raised and become the “fasting growing puzzle in the world” by world media. Puzzle-addict was provided the puzzles on hundreds of websites; also many modern newspapers publish the puzzle on daily and weekly basis [5].

After that a efficient and plentiful generated puzzles become a profitable venture. According to the report of CNN money one firm that conducts the puzzle generation program received revenue of over 1 million dollar [6].

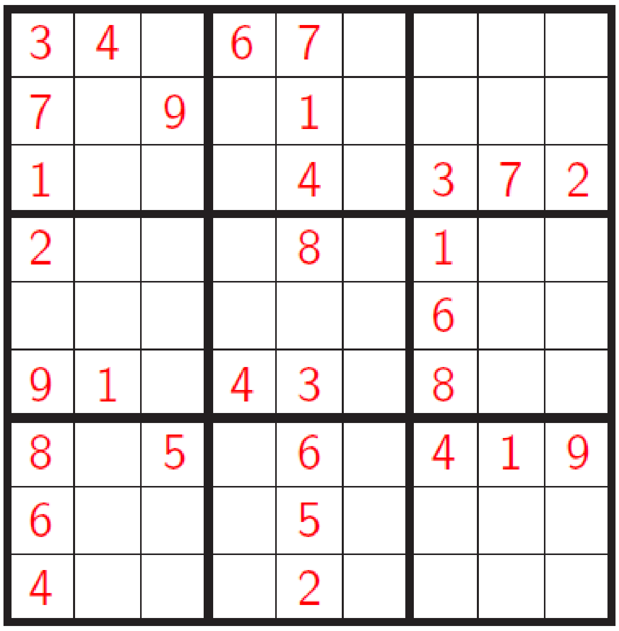
1.2 Terminology

In Fig 1 shows an example of Sudoku that is given with a 9X9 grid of cells and further divided into blocks that is the 3X3 subsections.

As an example, each cell contains single number as shown in figure to fulfill the requirement follows:

Cell must contain a number i.e. from 1 to 9.

- Every row must contain exactly one number from 1 to 9.
- Every column must contain exactly one number from 1-9.
- Each box must contain exactly one number from 1-9.

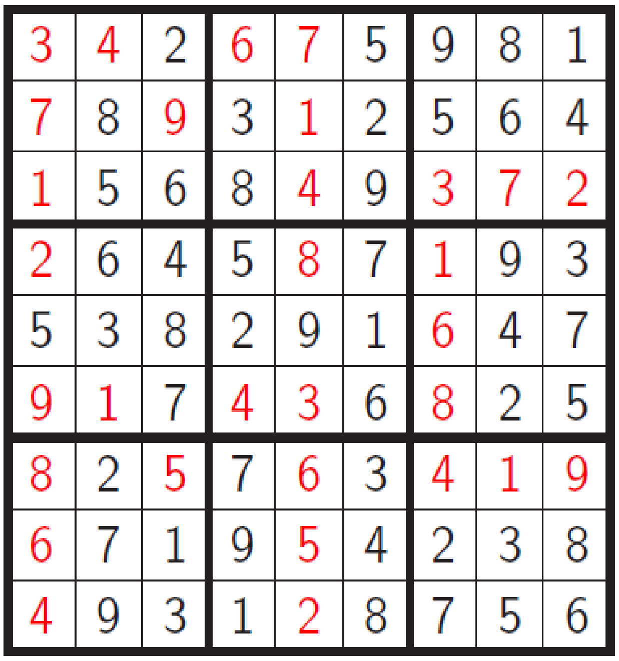


3	4		6	7				
7		9		1				
1				4		3	7	2
2				8		1		
						6		
9	1		4	3		8		
8		5		6		4	1	9
6				5				
4				2				

Fig. 1. Sample sudoku puzzle

Some numbers are previously filled in Sudoku that are called as givens, that must satisfy the conditions follows;

- No givens should void the above rules.
- In result of Sudoku could have a single or multiple solution i.e. given's set should be solvable.



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

Fig. 2. Solution to given puzzle

2. LITERATURE REVIEW

Brute force algorithm provides an effective approach in a least efficient manner to solve any problem. Brute force technique will enumerate all integers from 1 to 9 and tries to fit to find a solution. Brute force algorithm will provide a definite solution as long as problem is valid. It is to implement. This algorithm uses the computational capabilities of computers to maximum extent. This algorithm is efficient for solving problems with small complexity. [7]

It has been discovered that 6.67 x 1021 final grids exists for Sudoku. Any Sudoku problem is one of the grids. The efficiency of brute force algorithm depends how quickly brute force search finds the result so; the efficiency of the algorithm is unrelated to the complexity of the problem. [8]

3. PREVIOUS WORK

Sudoku solving techniques, algorithms and combinations are studied well. Approx. 6:671X1021 valid Sudoku can be formed by approx. 5.25X1027 9x9 Latin Squares that was shown by the Felgenhauer and Jarvis in 2005 [9]. Producing a puzzle requires minimal numbers of clues to get a unique solution is 17 is known as Royle hypothesizes. A Sudoku puzzle can be solved by NP- complete that is showed by Yato and Seta [10].

Human uses many logical strategies to solve the Sudoku, in 7 categories. [11] By omitting lengthy discussion we can say that they may vary from simple examination of different elements in row/columns. Mathematical solving techniques vary from brute force algorithms and recursive backtracking that is a classic computer science assignment to stochastic methods, integer programming, genetic algorithms, constraint solving algorithms and computer learning. [12]

Difficulty rating algorithm is used to make the puzzle more effective by determining computational difficulty without increasing the difficulties for human solver. Fowler describes a function of techniques to determine the human difficulty. Depending on the difficulty of puzzle more difficult techniques are provided more times although the choice has been made by the programmer.

Puzzle generating algorithms may vary in publicly available algorithms. As a randomize approach is used in grand majority i.e check for the unique solution by randomly replacing the numbers in random positions. [13]

Solving a Sudoku is easier then generating it and generating it is easier then evaluating its difficulty as per Flower that is also been confirmed by Sudoku aficionados an online forum

4. RECURSIVE FITNESS ALGORITHM

Recursive fitness algorithm works on two additional principles to the brute force algorithm. The detailed description of algorithm is given below.

4.1 Non-Repetitive Random number generation

This function generated random numbers between 1 and 9 without repetition. If the problem is solved by using any random number there are chances of repetition of these numbers. Due to this we may end up trying and trying with the numbers that are not part of solution. In other terms we are testing with repeated numbers and wasting time hence decreasing efficiency. This provides advantage of not entering the infinity loop while solving through brute force algorithm.

Whenever this function is used in the algorithm it will return a random number without repetition and if all number has been returned it will reset the counter. For example the sequence of numbers returned by this function will be: 1245793682543681

4.2 Row Fitness Test

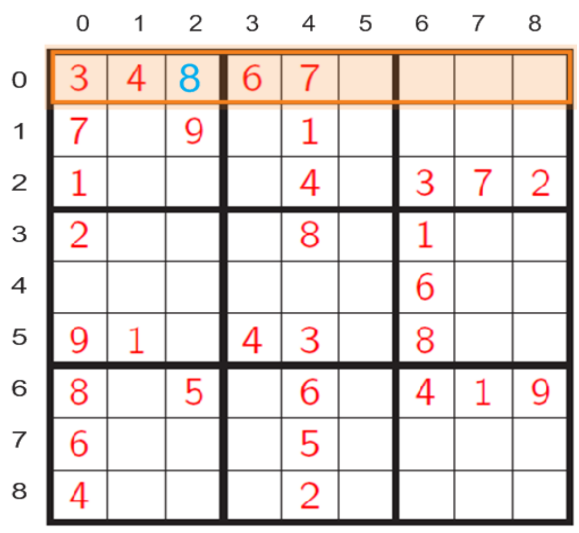
This function tests the fitness of a number in a row. The rule of Sudoku states that one number can only exist in a row once. So, if the number is already present in any cell of that row the test fails. This function repeatedly requests for a non-repetitive number form function Non-Repetitive Random Number Generator. If the fitness test fails for a particular number the function requests for a new number from Non-Repetitive Random Number Generator.In the example given in Figure 3 the Row fitness is tester for number 8 in 0 row

4.3 Column Fitness Test

This function is similar to row fitness test except it tests for fitness in column of the cell. If the number is already present in any cell of that column the test fails. This function also uses Non-Repetitive Random Number Generator function for generating random numbers.

4.4 Box Fitness Test

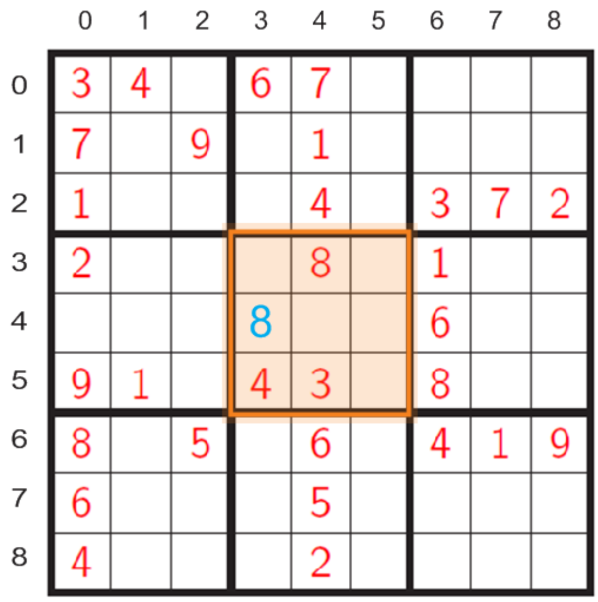
As per the rules of Sudoku each box should permit a number to appear only once. This function is trickier than previous two fitness functions. Sudoku has 81 cells and 9 boxes. The working of this function is divided into two parts. The first part of the function identify the box in which cell belongs. The second part of the function check for the fitness of the number generated by Non-Repetitive Random Number Generator in that box. If the test fails the second part of the function is repeated till a number is returned which fits in the box.



	0	1	2	3	4	5	6	7	8
0	3	4	8	6	7				
1	7		9		1				
2	1				4		3	7	2
3	2				8		1		
4							6		
5	9	1		4	3		8		
6	8		5		6		4	1	9
7	6				5				
8	4				2				

Fig. 3. Row fitness test

In figure 4, the example depicts the two phases of the Box Fitness Function Algorithm. For example while finding fitness of number 8 in the cell 4X3 the box is identified first. The box identified here ranges from 3X3 to 5X5. For the fitness the existence of number 8 is tested in all cells in the box. In our example the test fails as the number already exists in cell 3X4.



	0	1	2	3	4	5	6	7	8
0	3	4		6	7				
1	7		9		1				
2	1				4		3	7	2
3	2				8		1		
4				8			6		
5	9	1		4	3		8		
6	8		5		6		4	1	9
7	6				5				
8	4				2				

Fig. 4. Box fitness test

4.5 The Recursion

There no adaptive methods used here to deduce the result in this algorithm. Sudoku has possible 6:671X1021 solutions out of 5.25X1027 possibilities. This means that there are 5.25X1027 minus 6.671X1021 formations that are not valid Sudoku. There is good possibility that we may end up forming one of the invalid Sudoku.

In our algorithm if any of the functions: Row Fitness Test, Column Fitness test or Box Fitness Test calls the Non-Repetitive Random number generation for more than 9 times that means we are at the point of no-solution i.e. we have tried fitness for all numbers between 1 to 9. At this point we start processing again from cell 0X0.

The Algorithm

The algorithm runs as follows:

Step 1: Generate a Non-Repetitive Random number using Non-Repetitive Random number generation.
Step 2: Check for fitness in row using Row Fitness Test.
Step 3: If Step: 2 is repeated for more than 9 times go to Step: 1.
Step 4: Check for fitness in column using Column Fitness Test.
Step 5: If Step: 4 is repeated for more than 9 times go to Step: 1.
Step 6: Check for fitness in box using Box Fitness Test.
Step 7: If Step: 6 is repeated for more than 9 times go to Step: 1.
Step 8: If fitness test is successful for cell 8X8 then stop execution and print solution.

5. TESTING RESULT

The problem has been divided in to 5 complexity levels, 1 being the least complex problem and 5 being the most complex Sudoku problem. The execution was made on the Intel Pentium Dual CPU E2180 processor. Twenty five cycles of each complexity was executed and data captured as processing time in Nano seconds.

The data from execution is depicted in the graph in figure 6. The y axis of the graphs shows the execution time in Nano seconds and x axis of the graph is the pass or test number. The series 1 to 5 shows the complexity of problem. Series 1 being the lowest complexity problem and series 5 being the highest complexity problem.

It is very much clear from the graph that higher the complexity lesser the redundancy in the data. Thus this algorithm is stable at the lower complexity problems. In case of brute force algorithm we don't see this stability even for low complexity problems.

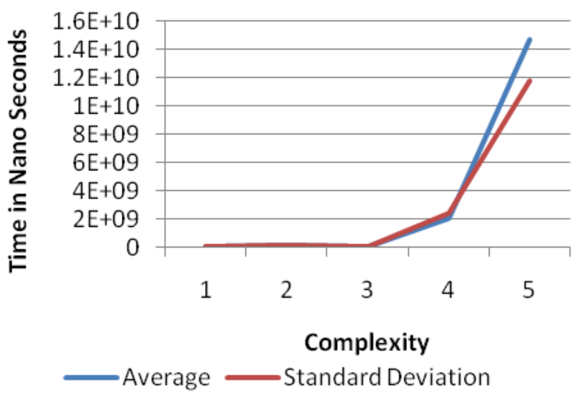


Fig. 5. Average and Standard Deviation

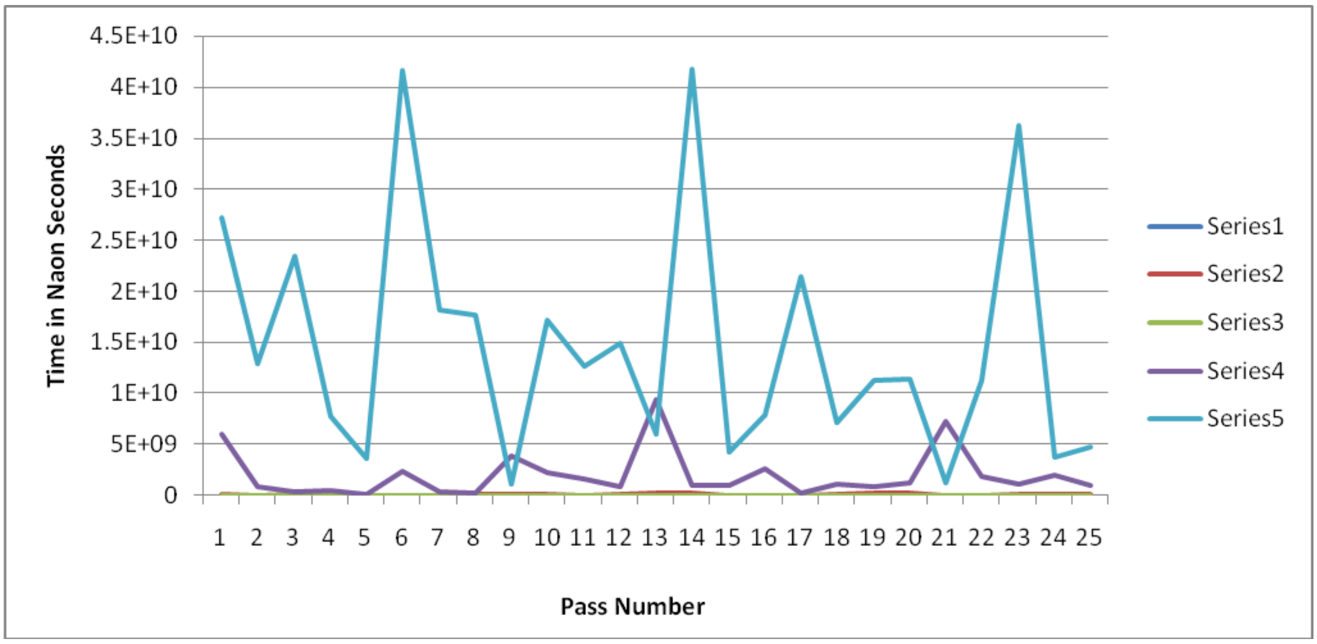


Fig. 6. Execution data

Figure 5, shows the relationship between the average and standard deviation with level of complexity of problem. This graph put some more light on the judgment derived from figure 6. Higher the complexity of the problem higher is the average time required for execution and standard deviation in the execution time.

6. CONCLUSION

From the testing results of our algorithm we can conclude that Recursive Fitness Algorithm for Generating and Solving Sudoku shows some stability for less complex problems i.e. deviation from average result is low. However it is not stable at higher complex problems.

As explained earlier, this algorithm is based on the brute force algorithm with to additional functions: Fitness Test and Non-Repetitive Random number generator. In comparison to brute force algorithm in which solving time is unrelated to the difficulty of problem this algorithm portrays the relation with the complexity of the problem i.e. the efficiency of this algorithm is greater for lower complexity problems and higher for more complex problems.

This may not be the best algorithm to solve Sudoku puzzles, but adding two functions to brute force algorithm overcomes the problem statement of brute force algorithm.

7. REFRENCES

[1] T. 2280, "Sudoku: Bagging a Di±culty Metric & Building Up Puzzles," 18 February 2008.

[2] J. K. L. A. R. K. D. S. E. L. Lawler, The Traveling Salesman problem- A Guided Tour of Combinatorial Optimization, John Wiley & Sons, 1985.

[3] "What is Sudoku?," [Online]. Available: http://www.sudoku-space.com/sudoku.php. [Accessed 1 February 2013].

[4] "Nikoli," 1 February 2013. [Online]. Available: http://www.nikoli.co.jp/en/publication/sudoku_books.html . [Accessed 23 April 2013].

[5] "Sudoku - popularity in the media," February 2008. [Online]. [Accessed 10 March 2013].

[6] P. L. Monica, "Much ado about sudoku," September 2005. [Online]. [Accessed 26 April 2013].

[7] J.P B.P. Christof Paar, Understanding Cryptography: A Textbook for Students and Practitioners, Germany: Springer, 2010.

[8] S. B. S. P. K. D. Kedar Nath Das, "A Retrievable GA for Solving Sudoku Puzzles," 2008. [Online]. Available: http://www.cse.psu.edu/~sub194/papers/sudokuTechReport.pdf. [Accessed 10 February 2013].

[9] F. J. B. Felgenhauer, "Enumerating possible Sudoku grids," 2005.

[10] T. S. T. Yato, "Complexity and completeness of ¯nding another solution and its application to puzzles.

[11] G. Royale, "Minimum Sudoku," [Online]. Available: http://school.maths.uwa.edu.au/~gordon/sudokumin.php. [Accessed 13 April 2013].

[12] J. O. Ines Lynce, "Sudoku as a SAT Problem," Proc. of the Ninth International Symposium on Artificial Intelligence and Mathematics. Springer, 2006.

[13] G. Fowler, "9x9 sudoku solver and generator," 2008.