ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 7, July 2016

Algorithm to Compute Sudoku Solution by Using Constraint Satisfaction and Permutation in Polynomial Time

Prashil Bhimani

Final Year Student, Department of Computer Science & Information Technology, VJTI, Mumbai, India

ABSTRACT: A standard Sudoku puzzle has a 9 x 9 grid which is to be filled with numbers 1-9 in such a way that the numbers are not repeated in individual row, column or the 9 mini-grids of 3 x 3. A partially filled grid having a unique solution acts a puzzle to be completed. Sudoku solvers are generally implemented using 2 techniques namely solving by rules or using brute force brute force. The algorithm proposed in this paper combines the two approaches to reduce the time complexity and solve it in polynomial time.

KEYWORDS: Sudoku, Solver, Constraint satisfaction, Permutation, Puzzle.

I. INTRODUCTION

'Sudoku' is the Japanese abbreviation of a longer phrase, 'Suuji wa dokushin ni kagiru', meaning 'the digits must remain single'. The basic constraint to solve it involves checking whether a particular number in the domain is present anywhere in containing row ,column or mini-grid. There are various sizes in which a Sudoku example is available like *4 x 4*, 9 x 9, *16 x 16*, *25 x 25* and so on. The mini-grids are of the square root of the number of rows or columns. So a *9 x 9* Sudoku will have mini-grids of *3 x 3* and will have 9 such exclusive mini-grids. Some of the blocks are filled with numbers which are called hints and these hints have to be used to fill all the remaining blocks.
Most of the techniques used by researchers are logical or rule based techniques. Some are simple logic, some are more advanced. Depending on the difficulty of the puzzle, a blend of techniques may be needed in order to solve a puzzle. Most Sudoku are ranked in difficulty on basis of number of blocks filled. Although the position of the blocks filled also matters. Table 1 shows a comparison chart of the number of clues for different difficulty levels [4].

Level	Empty Cells
Easy	40-45
Medium	46-49
Difficult	50-53
Very Difficult	54-58

In our proposed method, I have combined the two approaches of rules, as constraints, and permutations

II. LITERATURE REVIEW

Current algorithms are based on 2 different approached.

I. Brute force- In this approach all the empty blocks are substituted with numbers 1-9 and all possible permutations are checked. Considering there are r number of empty blocks. Each block will have 9 possibilities or n number of possibilities. So the number of permutations that can be formed is n^r. So the time complexity of this solution is very high. For a *3 x 3* Sudoku solution with 35 blocks filled i.e. 46 blocks empty the number of permutations generated are 7.86×10^{43} . Hence it is not feasible to apply permutation.

II. Apply rules of Sudoku to solve - Mathematics of Sudoku describes various rules[2] of solving a Sudoku using pure mathematics :

- Unique missing candidates

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 7, July 2016

- Naked singles
- Hidden singles
- Locked Candidates
- Naked/Hidden pairs/triplets/quads
- X-Wings and swordfish
- XY Wing
- Coloring and Multi coloring
- Unique Solution
- Forcing chains

As we try applying more and more rules, the time complexity to apply these rules increases. Applying only certain rules will also not give the complete solution in some cases where the number of blocks that have to filled are more. So it is not prudent to apply permutation of all solutions neither is it prudent to apply all rules.

III. PROPOSED ALGORITHM

A. *Description of the Proposed Algorithm:*

The proposed algorithm has 2 function:

I. Constraint Satisfaction

The domain of a particular block of a Sudoku puzzle of 3 *x 3* are the numbers 1-9. The domain is initially set for all empty blocks as the numbers 1-9 and the domains for the non-empty blocks is the value of that block.

The Mathematics of Sudoku, by Tom Davis[2], postulates various constraints to solve the Sudoku problem. The algorithm applies the following constraints sequentially.

I. **Unique missing candidates** - If 8 elements of a particular row, column, box are filled then the last remaining element of that particular row, column or mini-grid must be filled with the remaining number.

II. **Naked singles** - If domain of a particular block has only one remaining value then that value must be filled with the number remaining in its domain.

III. **Hidden singles** - If a particular number appears only once in the union of domains of all elements of a particular row, column or mini-grid then the block with that number as an element must eliminate all the remaining numbers from its domain.

These constraints reduce the domain size of many blocks. On using it iteratively domains of most of the blocks will then have only one value.

II. Permutation

Once we have iteratively applied constraint satisfaction till we domain for any of the blocks is not reduced further. All the values currently present for every block domains will be valid up till now. So all these values can be alternatively applied to find the solution by using permutation. But permutation for all blocks will exponentially increases time for each permutation. So we can applying permutation on one block reducing its to domain we can apply constraint satisfaction again till the domain for all the blocks cannot be reduced further. To optimize the algorithm further the permutations is applied to a block which has least number of possible values in its domain.

If at any point domain for any block becomes null the algorithm backtracks to the previous state of domain. The value that was currently used is discarded from the domain and the next available value is chosen.

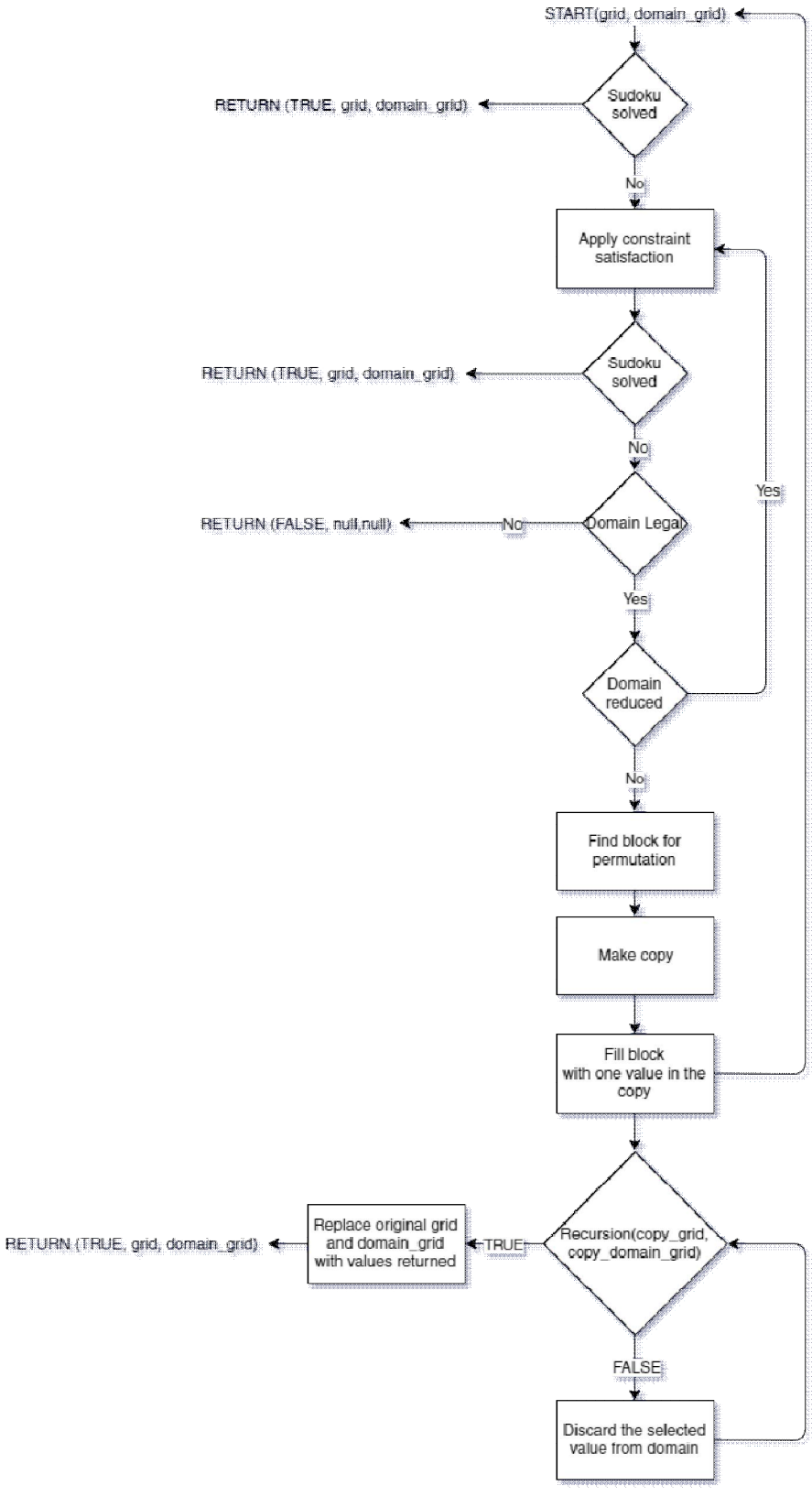
A case where more than 1 permutation is applied, it may happen that the value of the first permutation is invalid, but it does not cause the domain of any block to become null, but the permutation applied after it may cause this condition. In such a case the algorithm backtracks twice to the domain before the 1st permutation was applied.

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 7, July 2016



ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 7, July 2016

IV. TIME COMPLEXITY

The time complexity of an algorithm quantifies the amount of time taken by an algorithm to run as a function of the length of the string representing the input. For a Sudoku problem the input can be considered as the value of every block at the start of the play. So for a *3 x 3* Sudoku problem the length of the input will be 81. For the algorithm we can divide the time complexity in 2 parts. The time used by constraint satisfaction and time required by the permutations. For a *3 x 3* Sudoku to have a unique solution a minimum of 17 blocks of at least 8 different numbers[3]. So suppose there are p number of empty blocks.

For Constraint Satisfaction:

I. **Unique missing candidates** - To find the unique missing candidates we can traverse across the row once and find single missing candidate. When this is done for n row the complexity become n^2. Repeated for column and box the complexity to find unique missing candidate becomes $3n^2$.

II. **Naked singles** - To find naked singles we need to traverse the complete grid once to find blocks with one 1 element in its domain. So the time complexity is n^2.

III. **Hidden singles** - To find the hidden singles we can traverse across the row once and find the element which occurs only once in the entire domain of that row. When this is done for n row the complexity become n^2. Repeated for column and box the complexity to find unique missing candidate becomes $3n^2$.

So for over all time complexity for the 3 rules applied is $3n^2 + n^2 + 3n^2 = 7n^2$ i.e. $O(n^2)$**.** But constraint satisfaction should be applied multiple times as to give solution. Considering reduction in at least 1 domain value every loop. The rules have to be applied for *n*p* times. So time complexity for Solution by only constrain satisfaction is $O(pn^3)$.

For Permutation

The time complexity of the permutation will be $O(n^r)$ where r is the number of times permutation is applied. Hence r>=0 or r<=p.

The total time complexity of the algorithm becomes $O(n_1^r)$. This sounds like a very high complexity to solve the Sudoku. The brute force algorithm requires $O(n^p)$ time complexity. The benefit of this algorithm is that by applying constraint after every permutation we can reduce the number of possibilities in permutation can be applied. Apply permutation to values with least values increases the probability of a right choice increases exponentially.

The following table show the data of number of permutation applied for solving 60,000 Sudoku for a given number of blocks filled[1] using the given algorithm. The second table shows the average length of the domain to permutation was applied.

Number of blocks filled	Total number of permutation applied for 10,000 Sudoku	Sum of domain length where permutation applied for 10,000 Sudoku	Average of permutations	Average domain length
17	13755	27707	1.3755	2.0143
25	8904	17808	0.8904	2
30	1157	2314	0.1157	2
35	171	342	0.0171	2
40	10	20	0.001	2
45	3	6	0.0003	2

So average time complexity is very less where n~2 and r<2. So the total time complexity of the algorithm can be given by $O(n_1^r pn^3)$. Where worst case value of $n_1^r = 4$.

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 7, July 2016

V. SIMILAR APPLICATIONS

The combination of such constraint satisfaction and permutation can also be used to solve variants of Sudoku which are different is size like *4 x 4*. It can be also used to solve other variants of Sudoku like Mini Sudoku[5], Killer Sudoku[6], Hyper Sudoku[7], Alphabetical Sudoku[8]. the combination of the 2 approaches can be extended to KenKen[9], Futoshiki[10].

VI. CONCLUSION

We proposed general algorithm that cope with the problems of both primitive approach that is applied to solving Sudoku puzzles. We also attempted to improve the time complexity by optimizing the permutation to the minimum value. The result is that solutions could be found with low time complexity, even for the puzzles that are difficult to solve by the conventional method.

REFERENCES

1. Links to Sudoku sample -
http://www.printable-sudoku-puzzles.com/wfiles/files/1.zip - 45 blocks 10,000 examples.
http://www.printable-sudoku-puzzles.com/wfiles/files/2.zip - 40 blocks 10,000 examples.
http://www.printable-sudoku-puzzles.com/wfiles/files/3.zip - 35 blocks 10,000 examples.
http://www.printable-sudoku-puzzles.com/wfiles/files/4.zip - 30 blocks 10,000 examples.
http://www.printable-sudoku-puzzles.com/wfiles/files/6.zip - 25 blocks 10,000 examples.
http://staffhome.ecm.uwa.edu.au/~00013890/sudoku17 - 17 blocks first 10,000 examples.
2. Tom Davis, The Mathematics of Sudoku, Online - http://www.geometer.org/mathcircles/sudoku.pdf , September 13, 2012.
3. Gary McGuire , Bastian Tugemann , Gilles Civario, There is no 16-Clue Sudoku: Solving the Sudoku Minimum Number of Clues Problem, Online - http://arxiv.org/pdf/1201.0749.pdf , August 31, 2013.
4. Wei-Meng Lee, Programming Sudoku, Apress, USA , 2006.
5. Wikipedia, the free encyclopedia, Mini Sudoku - https://en.wikipedia.org/wiki/Sudoku#Mini_Sudoku
6. Wikipedia, the free encyclopedia, Killer Sudoku - https://en.wikipedia.org/wiki/Killer_sudoku
7. Wikipedia, the free encyclopedia, Hyper Sudoku - https://en.wikipedia.org/wiki/Sudoku#Hypersudoku
8. Wikipedia, the free encyclopedia, Alphabetical Sudoku -https://en.wikipedia.org/wiki/Sudoku#Alphabetical_Sudoku
9. Wikipedia, the free encyclopedia, KenKen - https://en.wikipedia.org/wiki/KenKen
10. Wikipedia, the free encyclopedia, Futoshiki - https://en.wikipedia.org/wiki/Futoshiki

BIOGRAPHY

Prashil Bhimani is a student in the Department of Computer Science & Information Technology, Veermata Jijabai Technological Institute(VJTI), Mumbai, India. He is currently a B. Tech student and is expected to graduate in the year 2016-2017. His research interests are Data Structures, Algorithms, Machine Learning, Data analysis etc.