Advances in Computer Science and Information Technology (ACSIT)
Print ISSN: 2393-9907; Online ISSN: 2393-9915; Volume 2, Number 8; April-June, 2015 pp. 71-76
© Krishi Sanskriti Publications
http://www.krishisanskriti.org/acsit.html

An Advanced Algorithm for Solution of Sudoku Puzzles

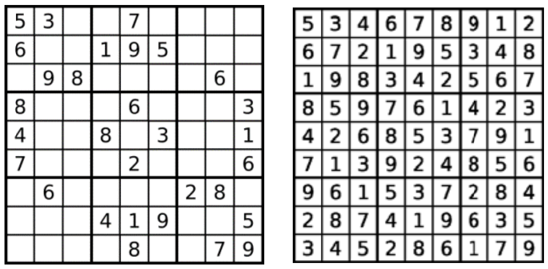
Puja Majumder

Tripura Institute of Technology
E-mail: puja.majumder1992@gmail.com

Abstract—*From the last decade, solving the Sudoku puzzle has become every one's passion. The simplicity of puzzle's structure and the low requirement of mathematical skills caused people to have enormous interest in accepting challenges to solve the puzzle. Therefore, developers have tried to find algorithms in order to solve the puzzles for human players so that they could be even solved by computer programming. This study aims to develop an advanced algorithm so that this puzzle can be solved easily & also implement the puzzle using Java programming language. The algorithm is implemented using Sudoku Operators- Row Operator, Column Operator & Block Operator.*

1. INTRODUCTION

'Sudoku' is the Japanese abbreviation of a longer phrase, 'Suuji wa dokushin ni kagiru ', meaning 'the digits must remain single'. It is a very popular puzzle that trains our logical mind. There are several approaches to solve this well-known & well-liked puzzle. The objective is to fill a (9*9) grid with digits so that each column, each row and each of nine (3*3) sub-grids that compose the grid (also called "boxes", "blocks", "regions" or "sub-squares") contains all of the digits from 1 to 9.The puzzle setter provides a partially completed grid, which for a well-posed puzzle has a unique solution. The same single integer may not appear twice in the same row, column or in any of the nine (3*3) sub-regions of the (9*9) playing board. Puzzle was popularized in 1986 by the Japanese puzzle company Nikoli under the name 'Sudoku', meaning single number. It became an international hit in 2005. In the last decade, solving the Sudoku puzzle has become every one's passion. The simplicity of puzzle's structure and the low requirement of mathematical skills caused people to have enormous interest in accepting challenges to solve the puzzle. Therefore, developers have tried to find algorithms in order to generate the variety of puzzles for human players so that they could be even solved by computer programming. Many researchers had presented an algorithm called pencil-and-paper using human strategies. The purpose was to implement a more efficient algorithm and then compare it with another Sudoku solver named as brute force algorithm. This algorithm is a general algorithm that can be employed in to any problems. The results have proved that the pencil-and-paper algorithm solves the puzzle faster and more effective than the brute force algorithm. There are currently different variants of Sudoku such as 4X4 grids, 9X9 grids and 16X16 grids. This work is focused on classic and regular Sudoku of 9X9 board. One such problem and its solution are shown in Fig. 1(a) and Fig. 1(b) respectively.



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

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

(a) (b)

Fig. 1: (a) An instance of the Sudoku problem. (b) A solution of this Sudoku instance (shown in Fig. 1(a)).

2. BACKGROUND

Sudoku was originally a Japanese game; however, some say it was Chinese. Born in the 18th century (about 200 years ago) it fairly recently stormed the world and can be seen everywhere (good things takes time).It's a fun puzzle dealing with numbers, but it is not math. No basic knowledge is needed, and anyone can play it. That is why it can be a game for the whole family. However, patience, time and concentration are necessary for addicts. It can be played anywhere, anytime and can be a great "time killer" (like waiting for a bus, at the beach, or for doctor's appointments).The Sudoku is a puzzle yet a sophisticated one that can keep you intrigued for long time. Perfection will come with practice. The long and interesting history of the Sudoku is quite a puzzle in itself. The name Sudoku comes from Japan and consists of the Japanese characters Su (meaning 'number') and Doku (meaning 'single') but the was not invented in Japan. Sudoku originated in Switzerland and then travelled to Japan by way of America. Sudoku has its deep roots in ancient number puzzles. For many centuries people have been interested in creating and solving puzzles. Puzzles of all kinds continue to be the basis of developing important new mathematics.

3. DIFFERENT SUDOKU ALGORITHMS

3.1 Brute-Force Methods

Kovacs outlines some of the brute-force methods used to solve Sudoku puzzles. The simplest method randomly assigns numbers to the empty cells and checks whether the completed puzzle is a solution. If not, this process is repeated until a solution is found. Clearly, this method can be very time consuming. A similar method is to generate all possible combinations for the empty cells. This can only be done for easy problems. Kovacs also suggests creating a search space for the puzzle and applying searches such as a depth-first search with backtracking[1]. Moreover, there is no enjoyment in solving Sudoku puzzles using Brute-Force technique[5].

3.2 Evolutionary and Genetic Algorithms

A fair amount of research has been conducted into the use of evolutionary and genetic algorithms to solve Sudoku puzzles. Evolutionary and genetic algorithms are based on Darwin's theory of evolution and as such evolve an initial population through the processes of evaluation, selection and regeneration to find a solution. Genetic operators such as reproduction, mutation and crossover are generally used for regeneration purposes. In addition, unauthorized users without knowing the secret key and the secret parameters used for the toral automorphism can not extract the embedded message[6].

3.3 Harmony Search

Geem evaluates harmony search as a means of solving Sudoku. This algorithm emulates different behaviors of musicians including random play, memory-based play and pitch-adjusted play. The harmony algorithm was able to solve an easy Sudoku puzzle in 9 seconds. The algorithm was unable to solve the hard problem which it was also applied to. The following table shows the number of clues given in a Sudoku puzzle in defining the level of difficulty of a Sudoku instance [2].

2.4 Data on Human Sudoku Solving

As a measure of problem difficulty for humans we use the mean solution time from all solutions. We have thoroughly checked that the main results are not dependent on this choice (we have done the analysis also for median time or for mean time computed only from a selected subset of active users) [3].

Table 1: Difficulty Level of Sudoku Puzzles based on number of clues given.

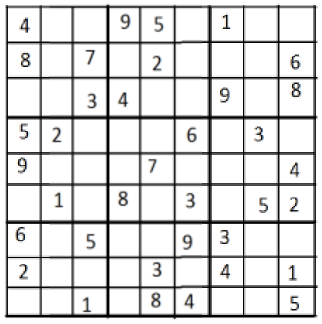
Difficulty Level	Number of Clues
Easy	38-46
Difficult	32-37
Evil	17-27

4. TIME-COMPLEXITY

Time complexity of an algorithm describes the time that is needed to run on a computer and it is commonly expressed using with O notation. This examines the time complexity of Sudoku solver. The input to a Sudoku solver is a Sudoku board. The standard board is a 9X9 grid and both smaller and larger boards are used. Theses types of puzzles with small sizes can be solved quickly and faster by computer, but only because this is small for a computer. Solving Sudoku is one of NP-complete problems and it says that Sudoku algorithms do not scale well to larger boards and puzzles, for example 10000X10000 grids is not feasible. If the size of input to Sudoku solver goes to infinity the time of complexity will increase exponentially. However, when solving the puzzles with limited input size such as 9X9 grids it is feasible because they can be solved in polynomial time[4].

5. SUDOKU SOLVING BY VARIOUS OPERATORS

The various Operators for solving Sudoku Puzzle are described below with appropriate figures.



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

Fig. 2: The Given Sudoku Puzzle

5.1 Row-Operator (r)

This Operator applies to each row of the Sudoku grid[1]. The pseudo code for the row operator is listed in.The row operator attempts to place the missing numbers in the row. For example, consider the first row in the grid in previous Figure. The row operator will attempt to place the missing numbers, namely, 2, 3, 6, 7 and 8. The numbers 2, 6 and 7 have more than one empty cell as an option and therefore are not placed. The numbers 3 and 8 can be placed. The number 3 cannot be placed in the first two available cells as the block these cells occur in contains a 3. Similarly, a 3 cannot be placed in the next two available cells as the columns that these cells occur in already have a 3. Thus, the 3 is placed in the last available cell. The number 8 cannot be placed in the first two available cells and the last two available cells in this row as the blocks containing these cells already have the number 8. The 8 is placed in the third empty cell in this row. The row operator is applied to the nine rows of the grid sequentially. The effect of applying this heuristic to the grid in Fig. is illustrated in Figures.

```
Procedure row_op(row)
Begin
```

```
for nums →1 to 9
begin
fin → false
for cells → 1 to available spaces in row AND NOT fin
begin
if (the column containing the next available cell does not contain
nums AND the block containing the next available cell does
not contain nums)
if (a cell has not been found yet)
cell → next available cell
else
fin → true
endfor
//If fin is not true only one cell has been found for nums
if (!fin)
Update the grid to store nums in cell
endfor
End
```

Fig. 3: Pseudocode for Row Operator

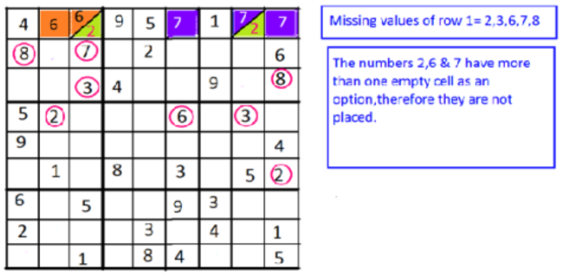

Missing values of row 1= 2,3,6,7,8
The numbers 2,6 & 7 have more than one empty cell as an option,therefore they are not placed.

Fig. 4: Application of Row_Operator

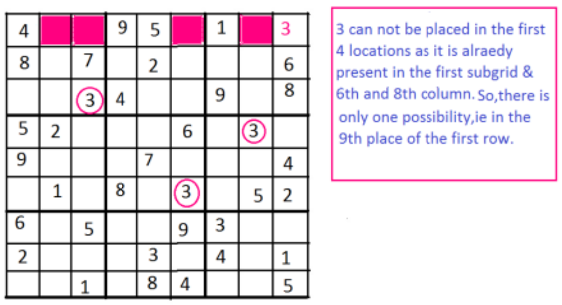

3 can not be placed in the first 4 locations as it is alraedy present in the first subgrid & 6th and 8th column. So,there is only one possibility,ie in the 9th place of the first row.

Fig. 5: Application of Row_Operator(Rest Part)

The Row_Operator can be applied as many times as possible in the rows one after another in the blank cells and the overall application of Row_Operator is shown in the Fig. as follows.

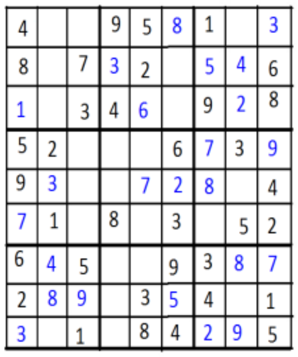


Fig. 6: Application of Overall Row_Operator

4.2 Column_Operator (c)

Like the row Operator the column Operator applies to each column sequentially. The column operator also attempts to place the missing numbers in the particular column.

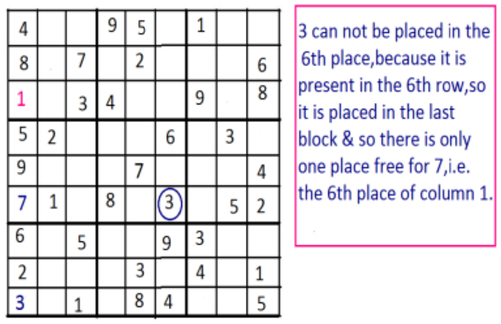

3 can not be placed in the 6th place,because it is present in the 6th row,so it is placed in the last block & so there is only one place free for 7,i.e. the 6th place of column 1.

Fig. 7: Application of Column_Operator

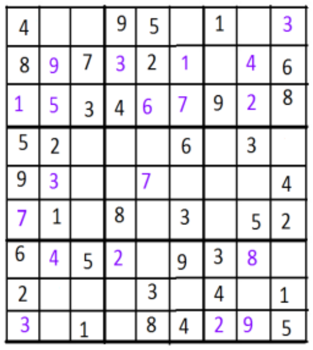
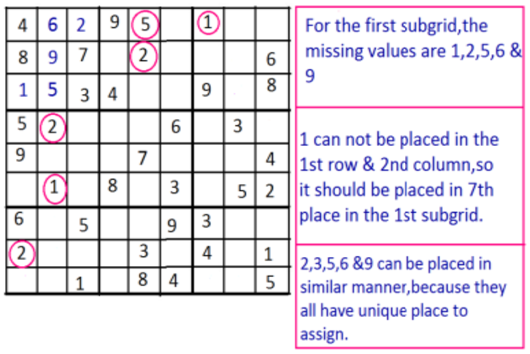


Fig. 8: Application of Overall Column_Operator

4.3 Block_Operator (b)

This Operator performs a similar function to that of the row and column heuristics and applies the block operator to each block sequentially.



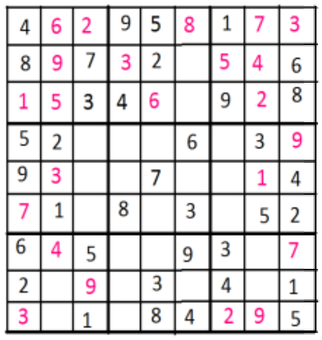
4	6	2	9	5		1		
8	9	7		2				6
1	5	3	4			9		8
5	2				6		3	
9				7				4
	1		8		3		5	2
6		5			9	3		
2				3		4		1
		1		8	4			5

For the first subgrid,the missing values are 1,2,5,6 & 9

1 can not be placed in the 1st row & 2nd column,so it should be placed in 7th place in the 1st subgrid.

2,3,5,6 &9 can be placed in similar manner,because they all have unique place to assign.

Fig. 9: Application of Block_Operator



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

Fig. 10: Application of overall Block_Operator

4.4 Other Operators

Other Operators include Row_Three_Blank_Cells Operator, Column_Three_Blank _Cells Operator, Block_ Three_Blank Cells Operator, which are applied in the Row, Column and Blocks respectively with three blank cells. Similarly there are Row_Two_Blank_Cells Operator, Column_Two_Blank_Cells Operator and Block_ Two_Blank_Cells Operator, which can be applied in the Row, Column and Block respectively with two blank cells.

```
Procedure row_3 _op(row) Begin
fin →false
for nums →NOT fin AND 1 to 9 begin
if (there are three free cells in which nums can be placed) begin
cell →randomly choose between the three cells options
fin →true
Update the grid to store nums in cell
endif
endfor
End
```

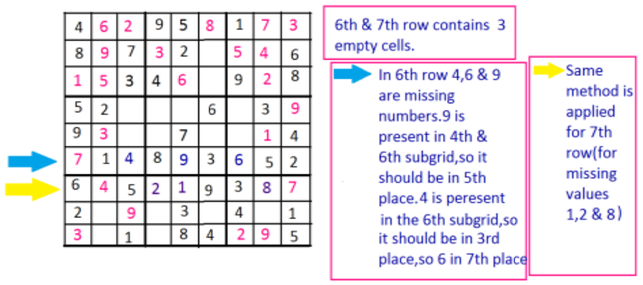
Fig. 10: Pseudocode for Row_Three_Blank_Cells

Operator

```
Procedure row_2_op(row) Begin
fin →false
for nums →NOT fin AND 1 to 9 begin
if (there are two free cells in which nums can be placed) begin
cell →randomly choose between the two cell options
fin →true
Update the grid to store nums in cell
endif
endfor
End
```

Fig. 11: Pseudocode for Row_Two_Blank_Cells

Operator



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

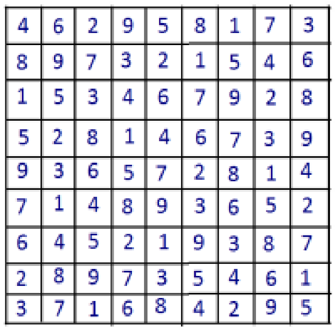
6th & 7th row contains 3 empty cells.

In 6th row 4,6 & 9 are missing numbers.9 is present in 4th & 6th subgrid,so it should be in 5th place.4 is peresent in the 6th subgrid,so it should be in 3rd place,so 6 in 7th place

Same method is applied for 7th row(for missing values 1,2 & 8)

Fig. 12: Application of Row_Two_Blank_Cells Operator

After applying all the Operators the Complete Solution is:



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

Fig. 13: The Complete Solution of the given

Sudoku Puzzle

```
public class Sudoku_Solver {
private static final int UNASSIGNED = 0;
private static final int N = 9;
static public void printSudoku(int[ ][ ] grid) {
for (int i = 0; i < N; i++) {
for (int j = 0; j < N; j++)
System.out.print(grid[i][j] + " ");
System.out.println(); } }
static boolean solve(int row, int col, int[ ][ ] cells) {
if (row == 9) {
row = 0;
if (++col == 9)
return true;
}
if (cells[row][col] != UNASSIGNED)
return solve(row + 1, col, cells);
for (int val = 1; val <= 9; ++val) {
if (is Safe(row, col, val, cells)) {
cells[row][col] = val;
if (solve(row + 1, col, cells))
return true;
}
}
cells[row][col] = UNASSIGNED;
return false;
}
static boolean isSafe(int row, int col, int val,
int[ ][ ]cells) {
for (int k = 0; k < 9; ++k)
// row
if (val == cells[k][col])
return false;
for (int k = 0; k < 9; ++k)
// col
if (val == cells[row][k])
return false;
int boxRowOffset = (row / 3) * 3;
int boxColOffset = (col / 3) * 3;
for (int k = 0; k < 3; ++k)
// box
for (int m = 0; m < 3; ++m)
if (val == cells[boxRowOffset + k][boxColOffset + m])
return false;
return true;
}
public static void main(String[] args) {
int grid[][] = { { 0, 0, 0, 0, 0, 0, 0, 0, 0 },
{ 0, 0, 0, 0, 0, 0, 0, 0, 0 }, { 0, 0, 0, 0, 0, 0, 0, 0, 0 },
{ 0, 0, 0, 0, 0, 0, 0, 0, 0 }, { 0, 0, 0, 0, 0, 0, 0, 0, 0 },
{ 0, 0, 0, 0, 0, 0, 0, 0, 0 }, { 0, 0, 0, 0, 0, 0, 0, 0, 0 },
{ 0, 0, 0, 0, 0, 0, 0, 0, 0 }, { 0, 0, 0, 0, 0, 0, 0, 0, 0 } };
solve(0, 0, grid);
printSudoku(grid);
doubletimeStart=System.currentTimeMillis();
double timeEnd=System.currentTimeMillis();
System.out.println("That took " + (timeEnd-timeStart)+ "
millis .to complete ");
}
}
```

Fig. 14: Java Code for Solving Sudoku Puzzle (of all difficulties)

6. CONCLUSION

Sudoku puzzle can be easily Solved by various Operators. Since, it has no backtracking problem, it can be considered to be an Advanced Algorithm.

7. ACKNOWLEDGEMENTS

I have great pleasure to express my deepest feelings of gratitude to Smt. Jhunu Debbarma, Associate Professor, Department of Information Technology, Tripura Institute of Technology, Agartala and Sri Bibhash Roy, Assistant Professor, Department of Computer Science and Engineering, Tripura Institute of Technology, Agartala, who as my supervisor and co-supervisor have extensively helped in completion of this work with their valuable suggestions, guidance, cooperation, patience and love. I do consider myself very lucky for working under such helpful gentle persons. I would like to thank all the teaching faculty, non-teaching staff members of this department for their valuable help and suggestions for the completion of my work. Last but not the least, most importantly I am thankful to my deceased father, my mother, family members for providing me with their unfailing support, understanding and love.

REFERENCES

[1] Finding Solutions to Sudoku Puzzles Using Human Intuitive Heuristics by Nelishia Pillay, School of Mathematics, Statistics and Computer Science, University of KwaZulu-Natal,South Africa, Research Article — SACJ No. 49, September 2012.

[2] An Exhaustive Study on different Sudoku Solving Techniques by Arnab Kumar Maji (Dept. of IT, North Eastern Hill University, Shillong, Mehgalaya), Sunanda Jana (Dept. of CSE, Haldia Institute of Technology, West Bengal), Sudipta Roy (Dept. of IT, Assam University, Silchar, Assam), Rajat Kumar Paul (Dept. of CSE, University of Calcutta, West Bengal).

[3] Difficulty Rating of Sudoku Puzzles by a Computational Model by Radek Pel´anek, Masaryk University Brno, Czech Republic, Proceedings of the Twenty-Fourth International Florida Artificial Intelligence Research Society Conference.

[4] A report on the sudoku solver by Fiorella Grados & Aref Mohammadi, Bachelor's Essay at dept. Computer Science Royal Institute of Technology.

[5] The Model and Algorithm to Estimate the Difficulty Levels of Sudoku Puzzles by Chungen Xu & Weng Xu Department of Applied Mathematics, Nanjing University of Science & Technology, Journal of Mathematics Research,Vol1,no 2,September 2009.

[6] A Sudoku Based Wet Paper Hiding Scheme by The Duc Kieu 1, Zhi-Hui Wang 3, Chin-Chen Chang 1, 2, and Ming-Chu Li 3, 1 *Department of Information Engineering and* Computer Science, Feng Chia University, Taichung, 2 Department of Computer Science and Information Engineering,National Chung Cheng University, Chiayi 3 School of Software, Dalian University of Technology, Dalian, Liaoning, China, International Journal of Smart Home Vol.3, No.2, April, 2009.