An Exhaustive Study on different Sudoku Solving Techniques

Arnab Kumar Maji[1], Sunanda Jana[2], Sudipta Roy[3], and Rajat Kumar Pal[4]

[1]Department of Information Technology, North Eastern Hill University, Shillong, Meghalaya 793 022, India

[2]Department of Computer Science and Engineering ,Haldia Institute of Technology, Haldia, West Bengal 721 657, India

[3]Department of Information Technology, Assam University, Silchar, Assam 788 011, India

[4]Department of Computer Science and Engineering,University of Calcutta, Kolkata, West Bengal 700 009, India

Abstract

'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-liked puzzle. In any case, the problem of solving a given Sudoku puzzle finds numerous applications in practice. In this paper, an exhaustive study has been made on different techniques for solving a Sudoku puzzle**.**

Keywords: Sudoku puzzle, Cell, Minigrid, Elimination, Backtracking*.*

1. Introduction

A Sudoku is usually a 9×9 grid based puzzle problem which is subdivided into nine 3×3 minigrids, wherein some clues are given and the objective is to fill it up for the remaining blank positions. Furthermore, the objective of this problem is to compute a solution where the numbers 1 through 9 will occur exactly once in each row, exactly once in each column, and exactly once in each minigrid independently obeying the given clues. One such problem instance is shown in Figure 1(a) and its solution is shown in Figure 1(b).

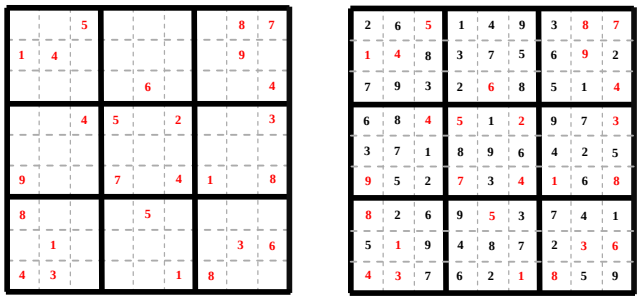


Figure 1. (a) An instance of the Sudoku problem. (b) A solution of this Sudoku instance (shown in Figure 1(a)).

Solving an instance of Sudoku problem is NP-complete [4]. So it is unlikely to develop a deterministic polynomial time algorithm for solving a Sudoku instance of size $n \times n$, where n is any large number such that the square root of n is an integer. But incidentally when the value of n is bounded by some constant, solutions may be obtained in reasonable amount of time [2, 3].

TABLE I. Number of clues given in a Sudoku puzzle in defining the level of difficulty of a Sudoku instance.

Difficulty Level	**Number of Clues**
1 (Extremely Easy)	More than 46
2 (Easy)	36-46
3 (Medium)	32-35
4 (Difficult)	28-31
5 (Evil)	17-27

TABLE II. The lower bound on the number of clues given in each row and column of a Sudoku instance for each corresponding level of difficulty.

Difficulty Level	**Lower Bound on the Number of Clues in Each Row and Column**
1 (Extremely Easy)	05
2 (Easy)	04
3 (Medium)	03
4 (Difficult)	02
5 (Evil)	00

There are quite a few logic techniques that researchers use to solve this problem. Some are basic 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. In fact, most computer generated Sudoku puzzles rank the difficulty based upon the number of empty cells in the puzzle and how much effort is needed to solve each of

them. Table I shows a comparison chart of the number of clues for different difficulty levels [3].
However, position of each of the empty cells also affects the level of difficulty. If two puzzles have the same number of clues at the beginning of a Sudoku game, the puzzle with the givens (or clues) in clusters is graded in higher level than that with the givens scattered over the space. Based on the row and column constraints, the lower bound on the number of clues are regulated in each row and column for each difficulty level [3] as shown in Table II.

In this paper, an attempt has been made to make an exhaustive study on the basic backtracking approach and other elimination based approaches for solving Sudoku puzzle.

2.Study on different Sudoku Solving Techniques

A 9×9 Sudoku puzzle can be divided into nine 3×3 minigrids. We have labelled each minigrid from 1 to 9, with minigrid 1 at the top-left corner and minigrid 9 at the bottom-right corner; minigrid numbers are shown in faded larger font size in Figure 2. Also we refer to each cell in the grid by its row number followed by its column number, as shown in the same figure.

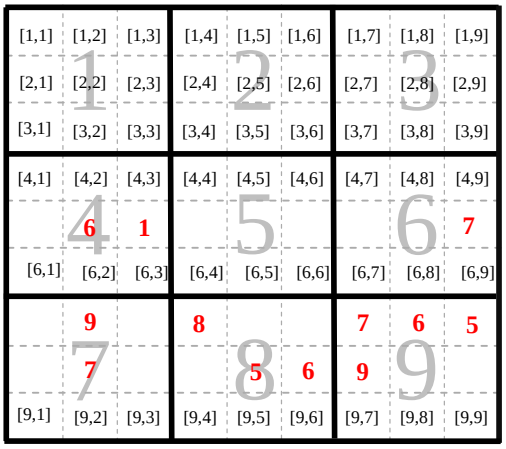


Figure 2. The structure of a 9×9 Sudoku puzzle (problem) with its nine minigrids of size 3×3 each as numbered (in grey outsized font) 1 through 9. Representation of each cell of a Sudoku puzzle and some example givens (or clues coloured by red) in the remaining cells. So, the cells are [1,1] through [9,9], and the distinct cells may have some clues as well. Minigrid numbered 1 consists of the cell locations [1,1], [1,2], [1,3], [2,1], [2,2], [2,3], [3,1], [3,2], and [3,3], minigrid numbered 2 consists of the cell locations [1,4], [1,5], [1,6], [2,4], [2,5], [2,6], [3,4], [3,5], and [3,6], and so on.

Now we review on the backtracking technique that has been adopted for solving Sudoku puzzles [3].

2.1 Backtracking

The basic backtracking algorithm works as follows. The program places number 1 in the first empty cell. If the choice is compatible with the existing clues, it continues to the second empty cell, where it places a 1 (in some other row, column, and minigrid). When it encounters a conflict (which can happen very quickly), it erases the 1 just placed and inserts 2 or, if that is invalid, 3 or the next legal number. After placing the first legal number possible, it moves to the next cell and starts again with a 1 (or a minimum possible acceptable value). If the number that has to be altered is a 9, which cannot be raised by one in a standard 9×9 Sudoku grid, the process backtracks and increases the number in the previous cell (or the next to the last number placed) by one. Then it moves forward until it hits a new conflict.

In this way, the process may sometimes backtrack several times before advancing. It is guaranteed to find a solution if there is one, simply because it eventually tries every possible number in every possible location. This algorithm is very effective for size two puzzles. Unfortunately, for size three puzzles, there are nine possibilities for each cell. This means that there are roughly 9^{81-n} possible states that might need to be searched, where $n \times n$ is the size of a given puzzle. Obviously this version of backtracking search does not work for size 3 puzzles. Fortunately, there are several means by which this algorithm can be improved: *constraint propagation*, *forward checking*, and *choosing most constrained value first* [2] are some of them.

Some other techniques include *elimination based approach* [3] and *soft computing based approach* [2]. Let us now focus to review the elimination based approach. In this approach, based on the given clues a list of possible values for every blank cell is first obtained. Then using the following different methods such as *naked single*, *hidden single*, *lone ranger*, *locked candidate*, *twin*, *triplet*, *quad*, *X-wing*, *XY-wing*, *swordfish*, *coloring*, we eliminate the multiple possibilities of each and every blank cell, satisfying the constraints that each row, column, and minigrid should have the numbers 1 through 9 exactly once. An instance of a Sudoku puzzle and its possible values of each blank cell are shown in Figures 3(a) and 3(b), respectively.

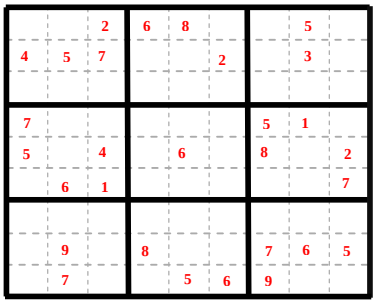
2.2 Naked single

If there is only one possible value existing in a blank cell, then that value is known as a *naked single* [2]. After assigning the probable values for each blank cell, as shown in Figure 3(b), we obtain the naked singles 3, 9, and 3 at locations [5,2], [5,8], and [8,3], respectively. So, we can directly assign these values to these cells. Then we eliminate these digits (or naked singles) from each of the corresponding row, column, and minigrid. Hence, after elimination of these numbers, as stated above, we obtain a modified (reduced) status of each blank cell as shown in Figure 3(c), wherein several other naked singles could be found (and this process is recursive until no naked singles are found).

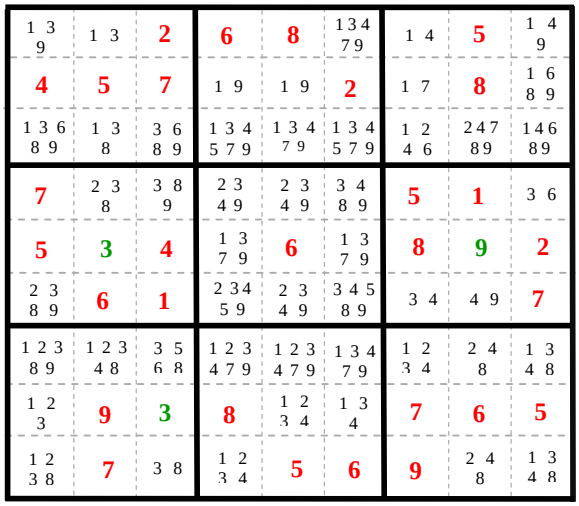
2.3 Hidden single

Sometimes there are blank cells that do, in fact, have only one possible value based on the situation, but a simple elimination of candidate in that cell's row, column and

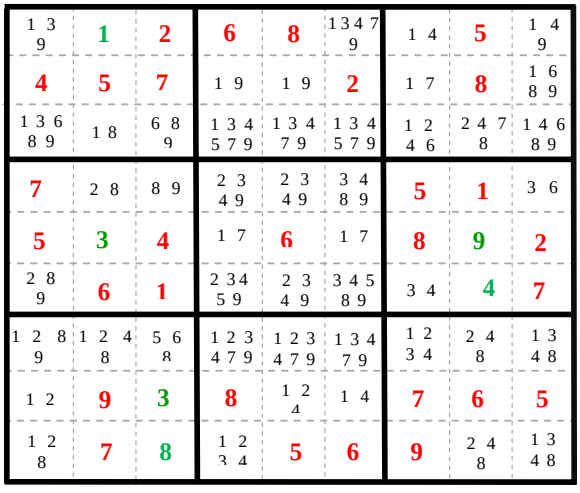
minigrid does not make it obvious. This kind of possible value is known as a *hidden single* [3]. Suppose, if we re-examine the possible values in each cell of Figure 3(b), hidden single can easily be found in cell [7,2] whose value must be 4 as in minigrid numbered 7, 4 is not there as probable values in other cells. Similarly, for cell [4,9], the hidden single is 6 (as in other cells of the same minigrid 6 is not present as probable values). Most of the puzzles ranked as easy, extremely easy, and medium can simply be solved using these two techniques of singles.



(a)



(b)



(c)

Figure 3. (a) An instance of a Sudoku puzzle. (b) Potential values in each blank cell are inserted based on the given clues of the Sudoku instance in Figure 3(a); here green digits are naked singles. (c) The concept of naked singles is preferably used to reduce the domain of probable candidate values in each blank cell, and the process is successive in nature to find out consequent naked singles, as much as possible. As for example, the naked single for cell [9,8] is 2, as 4 and 8 have already been recognized as naked singles along row 9 and column 8; then 8 is a naked single for cell [7,8], as 2 and 4 are already identified naked singles along column 8, and so on.

2.4 Lone ranger

Lone ranger is a term that is used to refer to a number that is one of multiple possible values for a blank cell that appears only once in a row, or column, or minigrid [3]. To see what this means in practice, consider a row of a Sudoku puzzle with all its possibilities for each of the cells (red digits are either givens or already achieved), as shown in Figure 4. In this row, six cells (with red digits) have already been filled in, leaving three unsolved cells (second, eighth, and ninth) with their probable values written in them.

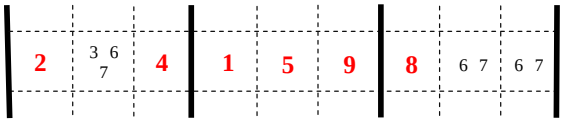


Figure 4. An example row of a Sudoku puzzle with a lone ranger 3 in the second cell.

Notice that the second cell is the only cell that contains the possible value 3. Since none of the remaining cells in this row can possibly contains 3, this cell can now be confirmed with the number 3. In this case, this 3 is known as a lone ranger.

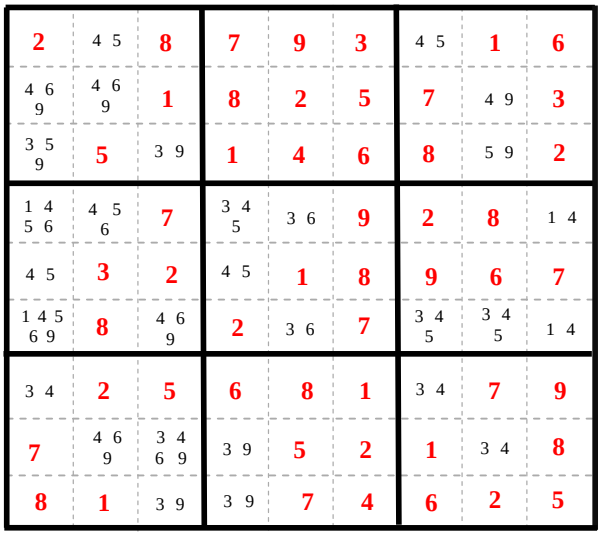


Figure 5. A Sudoku puzzle with probable locked candidates in the last row of minigrid 6 (and here the locked candidates are 3 and 5 in cells [6,7] and [6,8]), in the first column of minigrid 8 (and here the locked candidates are 9 and 3 in cells [8,4] and [9,4]), and so on.

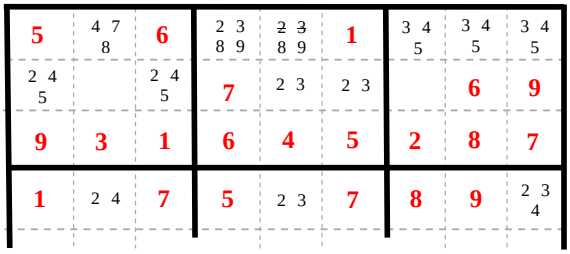
2.5 Locked candidate

Sometimes it can be observed that a minigrid where the only possible position for a number is in one row (or column)

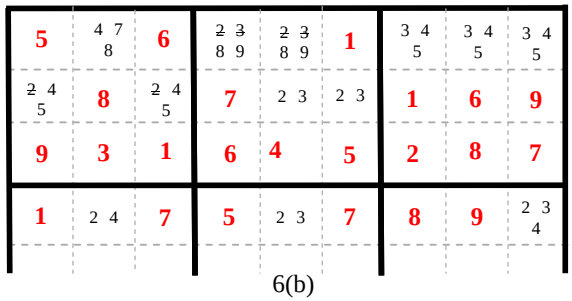
within that block, although the position is not fixed for the number. That number is known as a *locked candidate* [2]. Since the minigrid must contain the number in a row (or column) we can eliminate that number not as a probable candidate along the same row (or column) in other minigrids. Consider the Sudoku puzzle along with its probable assignments for each blank cell, as shown in Figure 5. It can readily be found that minigrid numbered 6 should have 3 in the last row. So we can simply eliminate number 3 from cell [6,5] of minigrid numbered 5. Similarly, minigrid numbered 8 should have 3 in its first column. So, 3 can be eliminated as a possible candidate from cell [4,4].

2.6 Twin

If two same possible values are present for two blank cells in a row (or column) of a Sudoku puzzle, they are referred to as *twin* [3]. Consider the partially solved Sudoku puzzle as shown in Figure 6(a). Observe the two cells [2,5] and [2,6]. They both contain the values 2 and 3 (means either 2 or 3). So, if cell [2,5] takes value 2, then cell [2,6] must contain 3, or vice versa. This type of situation is an example of twin.



6(a)



6(b)

Figure 6. (a) A partial Sudoku instance with presence of twin 2 and 3 in cells [2,5] and [2,6]. (b) Elimination of probable values (that are 2 and 3) based on the twin from the second row (2 is deleted from cells [2,1] and [2,3]) and from the same minigrid (2 and 3 are deleted from cells [1,4] and [1,5]).

Once a twin is identified, these values can be eliminated by striking through from the same row, column, and minigrid as shown in Figure 6(b), as the values can not be probable candidates in other blank cells along the same row (or column) and in the same minigrid.

2.7 Triplet

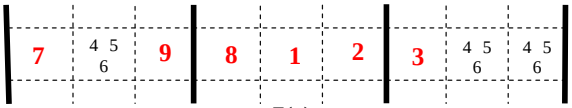
If three cells in a row (or column) are marked with a set of same three possible values, they are referred to as *triplet* [3]. Like twins, triplets are also useful for eliminating some other possible values for other blank cells. Triplet has several variations like the following.

Variety# 1: Three cells with same three possible values, as shown in Figure 7(a).

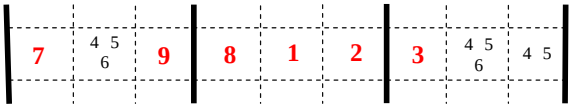
Variety# 2: Two cells with same three possible values and the other cell containing any two of the possible values, as shown in Figure 7(b).

Variety# 3: One cell with three possible values and the two other cells containing two different subsets of two possible values of the former three values, as shown in Figure 7(c).

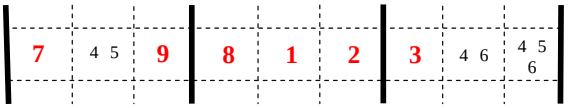
Once a triplet is found, we can eliminate all the values of the triplet that are there as possible candidates in other blank cells along the same row (or column) and in the same minigrid.



7(a)



7(b)



7(c)

Figure 7. Example rows of Sudoku puzzles with different varieties of triplet. (a) A triplet of *Variety# 1* with same three possible values present in three cells. (b) A triplet of *Variety# 2* with same three possible values present in two cells and the other cell containing any two of them. (c) A triplet of *Variety# 3* with three possible values present in one cell and the two other cells containing two different subsets of two possible values of the earlier three values.

2.8 Quad

Analogous to triplet, a *quad* consists of a set of four possible values and these values are present in some form in four blank cells in a row (or column) of the Sudoku instance [3]. That is, if the values only exist in four (blank) cells in a row (or column), while each cell contains at least two of the four values, then other values (or numbers except the specified four values) can be eliminated from each of the assumed cells (forming the quad). Figure 8 shows a row of a Sudoku puzzle where the quad comprising the digits 1, 2, 4, 7 formed by the cells in column four, six, seven, and eight. So other possible values can straightway be eliminated from these cells, as shown by striking through the inapplicable digits in the figure.

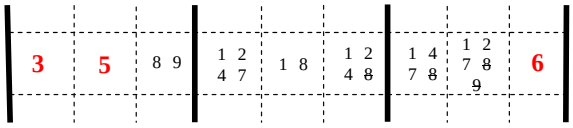


Figure 8. An example row of a Sudoku puzzle with quad comprising digits 1, 2, 4, and 7 present in columns four, six, seven, and eight. To support the digits present in the quad in the stated cells, other probable values (like 8 and 9 in columns six, seven, and eight) are eliminated from these cells of the quad, as these values (that are 8 and 9) cannot be probable digits for the specified cells.

2.9 X-Wing

The *X-Wing* [2] method can be applied when there are two rows and columns for which a given value is possible to assign only to two blank (diagonal) cells. If these four cells are only at the intersections of two orthogonal rows and columns, then all other cells along these rows and columns must not get assigned to this value. Figure 9 shows a Sudoku puzzle where 2 is present as probable value for cells [3,2], [3,8], [7,2], and [7,8]. It forms an *X-Wing*. So 2 can be eliminated as probable candidate value for other cells in the same row and column. In this example we can eliminate 2 as probable candidate for cells [3,1], [3,3], [3,9], [7,1], [7,3], and [7,9].

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

Figure 9. A Sudoku puzzle with X-Wing comprising digit 2 present at the crossings of rows three and seven, and columns two and eight.

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

Figure 10. A Sudoku puzzle with XY-Wing, where x = 5, y = 1, and z = 8.

2. 10 XY-Wing

If a Sudoku puzzle has three cells containing the list of probable candidate values in the following patterns: (1) all cells have exactly two candidates, (2) they share the same three candidates in the form of xy, yz, and xz, and (3) one cell (the 'stem' of Y with candidates xy) shares a group with the other two cells (the 'branches' of Y with candidates xz and yz). Then we can say that they form an *XY-wing* [1]. At that moment any other cell that shares a group with both 'branch' cells can have excluded the 'z' candidate that is common to the 'branch' cells. Consider the Sudoku puzzle shown in Figure 10, as pointed out, the cells [1,2], [1,7], and [6,7] contain the list of probable candidate values as {1,8}, {1,5}, and {5,8}, respectively. So the values of x can be assumed as 5, y as 1, and z as 8. Now, irrespective to cell [1,7] contains the value 1 or 5, the cell either [1,2] or [6,7] will have 8. Therefore, the intersection of these two cells, i.e., [6,2] cannot have 8, i.e., z as a probable candidate. We can easily eliminate z from the intersection of the cell containing xz and yz.

In the case above, the cells XZ and YZ share a common row or a common column with the cell XY. It may also be a case that it shares a minigrid as well. Consider the partial Sudoku puzzle structure shown in Figure 11.

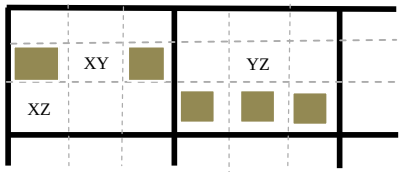


Figure 11. A Sudoku puzzle structure with XY-Wing, sharing minigrid.

Here the cell containing XZ shares a minigrid with cell containing XY, and the cell containing YZ shares a common row with the cell having XY. In this case, all the cells marked grey color cannot contain Z as a probable candidate. Because if X is assigned in cell [2,2], then Z must be assigned to cell [3,1], then cells [3,4], [3,5], [3,6], [2,1], [2,3] cannot have Z. Similarly, if Y is assigned to cell [2,2], then Z also cannot be assigned to all these cells.

2.11 Swordfish

Swordfish [1] is a generalization of X-wing. If a digit is a candidate for at most three cells in each of three different columns and the positions of the cells fall into the same three rows, then this digit cannot be a candidate for any other columns in the three matching rows.

In Figure 12, in the first, fifth, and seventh column the element 4 appears in the list of probable candidate values in the same three rows (i.e., the first, seventh, and eighth row) and 4 is not a member of any other rows of the columns. So, this is a swordfish pattern. Now the digit, i.e., 4, can safely be eliminated from other cells in the same rows. In the above figure, 4 can be removed not as a probable candidate from cell [1,8], and [8,4] as highlighted with green colour.

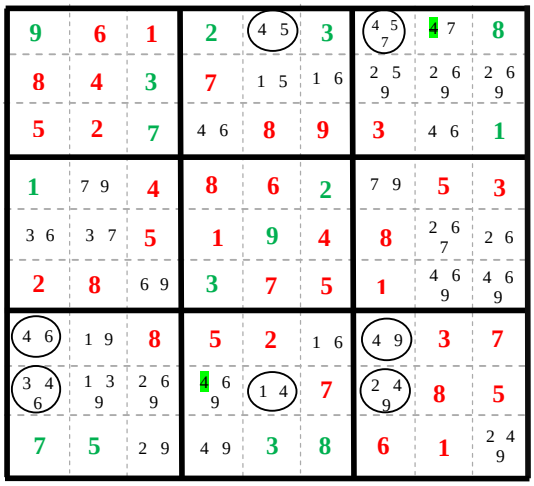


Figure 12. A Sudoku puzzle with swordfish occurred at the first, fifth, and seventh column.

2.12 Colouring

If a digit is a candidate for exactly two cells in a row, in a column, or in a minigrid, then the technique of *colouring* [1] may be very much dominant to make decisions to eliminate a probable candidate from a cell or to confirm a digit for a cell. There are several alternatives to colour a cell starting from single colouring to multi-colouring. Let us state a simple multi-colouring technique. Consider the following Sudoku puzzle as shown in Figure 13.

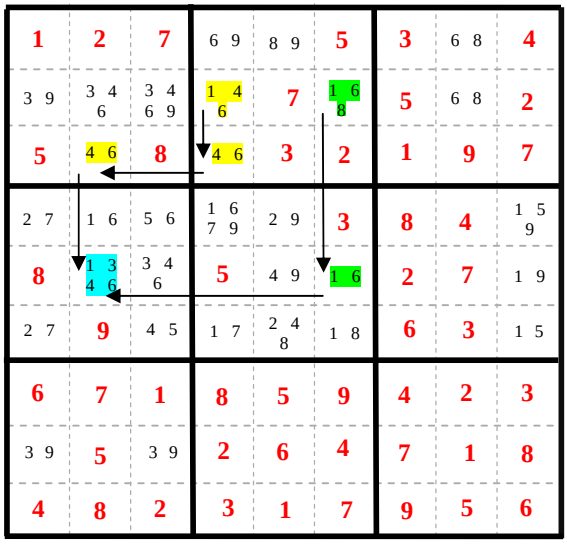


Figure 13. A Sudoku puzzle with multi-colouring for cells [2,4], [2,6], [3,2], [3,4], and [5,6].

Let us consider the second minigrid of the puzzle shown in Figure 13, digits 4 and 6 are candidates for exactly two cells [2,4] and [3,4]. Let us allocate yellow colour for these cells. Then we can also find that in the third row, 4 and 6 also appear two times at row cell number [3,2] and [3,4], and let us again mark these cells with yellow colour. Again we can find that 1 and 6 appear exactly two times in the sixth column. Let us assign a separate colour, say green colour, for these cells.

Now let us follow the path as shown by arrow marks and we can easily find the intersection between these yellow and green coloured cells, and it is marked as blue. We can easily conclude from all these that 6 cannot be present in the blue coloured cell and we can easily eliminate 6 as a probable candidate from the blue coloured cell. The example shows an elimination technique using multi-colouring method. It can also be used to confirm the value for a cell as well.

3. Comparison Among Different Sudoku Solving Techniques

In the preceding section, we have discussed the most fundamental method of backtracking and eleven different elimination based techniques used for solving a given Sudoku puzzle. The basic backtracking method first assigns some random variable to a blank cell, and then it goes on checking with other cells if a value previously assigned to that cell hits a conflict with other cell. If it hits with a conflict, then it backtracks to the previous cell and assign the next value. In this way, the method may sometimes backtrack several times before advancing. But, it is guaranteed to find a solution if there is one, simply because it eventually tries every possible number in every possible location. This method is very much time consuming.

Instead of assigning a value to a cell based on guess, in the elimination based approach a list of probable candidate values are maintained for each of the blank cells. Then the different patterns are searched using the different elimination based techniques such as *naked single*, *hidden single*, *locked candidate*, *lone ranger*, *twin*, *triplet*, *quad*, *X-wing*, *XY-wing*, *swordfish* as described in the previous section. Some of these techniques are able to directly assign some values in a cell. Some of them can only eliminate values from other cells present in the same row, column or minigrid. For example using *naked single*, *hidden single*, or *colouring*, we can directly assign some value for a particular cell. Whereas using the other methods, we can only eliminate the candidate values from other cells, as described in the previous section.

But these elimination based methods are also very much time consuming as each of them has to search for a particular pattern and also several backtracking is needed for solving a puzzle. It may not also be necessary to apply all the methods for solving a Sudoku puzzle. Generally, up to medium difficulty level puzzles are solvable using only *naked single*, *hidden single*, *locked candidate*, *lone ranger*, *twin*, *triplet*, and *quad* based techniques. Rests are advanced techniques, which are essentially used for difficult and evil puzzles.

5. Conclusion

In this paper, we made an attempt to study all the existing elimination based methods for solving the Sudoku puzzle. Apart from elimination based schemes, soft computing based techniques are also available. But all these existing methods are guess based and hence time consuming as well. Based on

the levels of difficulty, different methods are applied one after another to find a valid solution of a Sudoku puzzle, if any. In addition, each of these existing solvers solves an instance of the problem considering the clues one-by-one for each of the blank cell locations. Often guessing may not be guided by selecting a desired path of computing a solution and hence exhaustive redundant computations are involved over there.

References

[1] http://www.geometer.org/mathcircles [Last accessed on Feb. 2014]
[2] Jussien N., A-Z of Sudoku, USA: ISTE Ltd., 2007.
[3] Lee W.-M., Programming Sudoku, USA: Apress, 2006.
[4] Yato T. and T. Seta, "Complexity and Completeness of Finding Another Solution and Its Application to Puzzles," IEICE Transactions on Fundamentals of Electronics, Communications, and Computer Sciences, Vol. E86-A, No. 5, 2003, pp. 1052–1060.

Arnab Kumar Maji completed his B. Tech and M.Tech in the field of Information Technology in the year of 2003 and 2005 respectively. He is currently a senior research scholar of Information Technology department of Assam University, Silchar. He is working as an Assistant Professor of North Eastern Hill University, Shillong, Meghalaya, India since 2006. He has published more than 20 numbers of research paper in the field of algorithm, image processing and e-commerce. He is a professional member of ACM India.

Sunanda jana completed his B.Tech from Biju patnaik University of technology and M.Tech from Berhampur University in the year of 2007 and 2010 respectively. Currently she is working as an Assistant Professor Department of Computer Science, Haldia Institute of Technology, Haldia, Westbengal, India since 2011. Her research interest includes Algorithm, NP complete Puzzles etc.

Dr. Sudipta Roy completed his M.C.A and M.E. in the year of 2002 and 2005 respectively. He obtained his Ph.D. degree in the year of 2010. Currently he is working as Associate Professor in the Department of Information Technology, Assam University Silchar, India. His research interests includes Image Processing, Algorithm, Sensor Networks etc. Dr. Roy published more than 40 research papers in various National and International Journals.

Dr. Rajat K. Pal. He received his B.E. and M.Tech. degree respectively in 1985 and 1988, and awarded Ph.D. degree from IIT, Kharagpur in 1996. He is serving the University of Calcutta as a faculty in the Department of Computer Science and Engineering since 1994, and worked as the Head of the Department during 2005-2007. Dr. Pal has published more than 100 research articles and authored a book entitled "Multi-Layer Channel Routing: Complexity and Algorithms" that has jointly been published from ***NAROSA Publishing House***, New Delhi, ***CRC Press***, Boca Raton, USA and ***Alpha Science International Ltd***, UK, in September 2000. His major research interests include VLSI design, Graph theory and its applications, Perfect graphs, Logic synthesis, Design and analysis of algorithms, Computational geometry, Parallel computation and algorithms.