Volume 1, No. 3, Sept-Oct 2010
ISSN No. 0976-5697
International Journal of Advanced Research in Computer Science
RESEARCH PAPER
Available Online at www.ijarcs.info

Yet Another Organized Move towards Solving Sudoku Puzzle

Arnab K. Maji*
Department Of Information Technology
North Eastern Hill University
Shillong 793 022, Meghalaya, India
e-mail: arnab.maji@gmail.com

Rajat K. Pal
Department Of Information Technology
Assam University, Silchar
Cachar 788 011, Assam, India
e-mail: rajatkp@vsnl.net

Abstract: "Sudoku" is the Japanese abbreviation of a longer phrase, "Suuji wa dokushin ni kagiru", meaning "the digits must remain single". It is a challenging numeric puzzle that trains our logical mind. Solving a Sudoku puzzle requires no math, not even arithmetic. Even so, the game poses a number of intriguing mathematical problems. This paper describes an algorithm to solve the Sudoku puzzle in a systematic way. It guarantees a unique solution to the problem for all difficulty levels.

Keywords: Sudoku, Puzzle, Grid, Clue, Difficulty level, Logic, Algorithm, Backtracking.

I. INTRODUCTION

A Sudoku puzzle is a grid of *n* rows and *n* columns, in which some pre-assigned *clues* or *givens* have been entered. The size of the grid can be *n*×*n*, where *n* is an integer. The most common size of such a (square) grid is 9×9. We have divided a 9×9 grid into nine 3×3 minigrids. We have labeled each minigrid from 1 to 9, with minigrid 1 at the top-left corner and minigrid 9 at the bottom-right corner as shown in Figure 1. We refer to each cell in the grid by its column number followed by its row number. Figure 2 shows the coordinates of each cell in the grid.

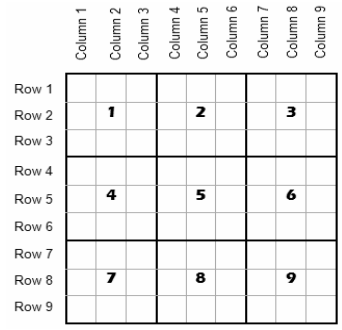

Figure 1. The rows and columns in a Sudoku puzzle.

(1,1)	(2,1)	(3,1)	(4,1)	(5,1)	(6,1)	(7,1)	(8,1)	(9,1)
(1,2)	(2,2)	(3,2)	(4,2)	(5,2)	(6,2)	(7,2)	(8,2)	(9,2)
(1,3)	(2,3)	(3,3)	(4,3)	(5,3)	(6,3)	(7,3)	(8,3)	(9,3)
(1,4)	(2,4)	(3,4)	(4,4)	(5,4)	(6,4)	(7,4)	(8,4)	(9,4)
(1,5)	(2,5)	(3,5)	(4,5)	(5,5)	(6,5)	(7,5)	(8,5)	(9,5)
(1,6)	(2,6)	(3,6)	(4,6)	(5,6)	(6,6)	(7,6)	(8,6)	(9,6)
(1,7)	(2,7)	(3,7)	(4,7)	(5,7)	(6,7)	(7,7)	(8,7)	(9,7)
(1,8)	(2,8)	(3,8)	(4,8)	(5,8)	(6,8)	(7,8)	(8,8)	(9,8)
(1,9)	(2,9)	(3,9)	(4,9)	(5,9)	(6,9)	(7,9)	(8,9)	(9,9)

Figure 2. The coordinates of cells in a Sudoku grid.

Besides the standard 9×9 grid, variants of Sudoku puzzles include the following:

- 4×4 grid with 2×2 minigrids,
- 5×5 grid with *pentomino* [6] regions published under the name *Logi-5*. A *pentomino* is composed of five congruent squares, connected orthogonally. *Pentomino* is seen in playing the game *Tetris* [13],
- 6×6 grid with 2×3 regions,
- 7×7 grid with six *heptomino* [7] regions and a disjoint region,
- 16×16 grid (super Sudoku),
- 25×25 grid (Sudoku, the Giant),
- A 3D Sudoku puzzle [8] (being invented by Dion Church was published in the *Daily Telegraph* in the U.K. in May 2005),
- Alphabetical variations, which use letters rather than numbers. The *Guardian* (in the U.K.) calls these Godoku [11] while others refer to them as Wordoku [12], etc.

A complete Sudoku solution grid may be arrived at in more than one way, as we can start from any given clues that are distributed over the minigrids of a given incomplete grid. Nobody has yet succeeded in determining how many different starting grids there are. Moreover, a Sudoku starting grid is really only interesting to a mathematician if it is minimal, i.e., if removing a single number means that the solution is no longer unique. No one has figured out the number of possible minimal grids, which amounts to the ultimate count of distinct Sudoku puzzles. It is a challenge that is sure to be taken up in the near future.

Another problem of minimality also remains unsolved; to wit, what is the smallest number of digits a puzzle maker can place in a starting grid and still guarantee a unique solution? The answer seems to be 17. Gordon Royle of the University of Western Australia has collected more than 38,000 examples that fit this criterion and cannot be translated into one another by performing elementary operations [5]. Gary McGuire of the National University of Ireland, Maynooth, is conducting a search for a 16-clue puzzle with a unique solution but has so far come up empty-handed [1]. It begins to look as if none exists. On the other hand, Royle and others working independently have managed to find one 16-clue puzzle that has just two

solutions [4]. Searchers have not yet uncovered any additional examples.

Is anyone near to proving that no valid Sudoku puzzle can have only 16 clues? Still the answer is "no". If we could analyze one grid per second, looking for a valid 16-clue puzzle within it, then the total time for searching all the grids might take 173 years [10]. Even using a fast computer we cannot achieve it in a reasonable amount of time. Even a distributed computing environment of 10,000 fast computers, might take time of about one year. It depicts the amount of computation involved in this problem, and a breakthrough (in developing a much better algorithm for searching) is necessary in our understanding to make it feasible to search all the grids in some realistic time.

Mathematicians do know the solution to the opposite of the minimum number of clues problem: What is the maximum number of givens that do not guarantee a unique solution? The answer is 77 [10]. It is very easy to see that with each of 80, 79, or 78 givens, if there is a solution, it is unique. But the same cannot be guaranteed for 77 givens.

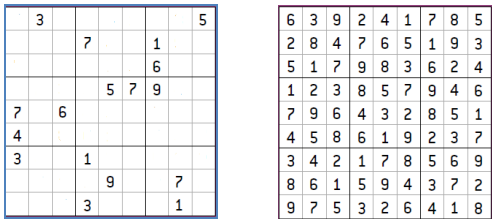

Figure 3. Sudoku example with 17 clues and its unique solution.

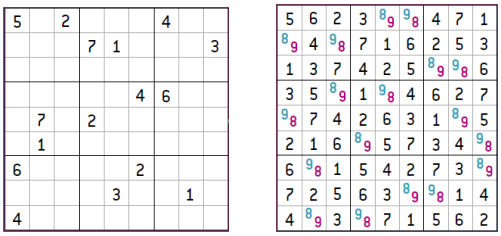

Figure 4. Sudoku example with 16 clues and its non-unique solution.

The minimum number of clues that a 9×9 Sudoku puzzle can start with and still yield a unique solution seems to be 17; an example is shown in Figure 3. One particular filled-in grid, known to Sudoku aficionados as "Strangely Familiar", or SF, hides 29 in equivalent 17-clue starting boards, an unusually high number; this means that from an SF grid we can deduce as much as 29 number of Sudoku puzzles with only 17 givens. SF is once considered the grid most likely to harbour a 16-clue puzzle with a unique solution, but an exhaustive search has dashed that hope. The only known 16-clue Sudoku having just two solutions appears in Figure 4; the final grids interchange the 8's and 9's.

II. LITERATURE SURVEY

An $n^2 \times n^2$ Sudoku grid (consisting of $n \times n$ blocks) is an NP-complete problem [3]. So, it is unlikely to develop a polynomial time algorithm to solve this problem. Hence, in this paper we have developed a heuristic algorithm that follows a set of organized moves to solve a given Sudoku puzzle. There are quite a few logic techniques people use to solve this problem. Some are basic simple logic, some are more advanced. Depending on the difficulty of the puzzle, a mixture 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 1 shows a comparison chart of the number of empty cells for different difficulty levels [2].

Table 1. Number of empty cells for each difficulty level.

Level	# of Empty Cells
1 (Easy)	40 to 45
2 (Medium)	46 to 49
3 (Difficult)	50 to 53
4(Extremely Difficult)	54 to 58

Now we review on the backtracking technique that has been adopted for solving Sudoku puzzles [3]. 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. If the number that has to be changed is a 9 (which cannot be raised by one in a standard Sudoku grid), the program backtracks and increases the number in the previous cell (the next-to-last number placed) by one. Then it moves forward until it hits a conflict.

In this way, the program 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 square. This means that there are roughly 9^{81-n} possible states that might need to be searched, where n is the number of given values. 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* [9], *forward checking* [9], and *choosing most constrained value first* [9] are some of them.

III. THE PROPOSED METHOD

The algorithms discussed above are very lengthy as well as time consuming as so much of trial and errors as well as guessing works are involved in those methods. In the proposed approach, we have tried to solve the Sudoku puzzle in an organized way where a limited number of guessing is involved. The proposed method is based on several elimination techniques. First we scan each of the cells in rows, columns, minigrids, and try to find out possible unique values for the empty cell. If unique values are found, we place one of them on the empty cell. Otherwise, note down the list of possible values for each of the empty cells. We called it as *Elimination Technique-I (ET-I)*, as it eliminates most of the values that are not valid for a minigrid.

We try to differentiate the minigrids by *solitary number*, where in a cell only a unique number could be placed. If found, put the solitary numbers in the empty cells and again apply *ET-I* in the way as stated above. If the puzzle is not solved, then try to find out the minigrids that are *clones* and *triads*, and eliminate some of them based on the same logic.

After elimination in the same way, we try again to find out the solitary number, if any. Then again apply *ET-I*. Note that for each valid assignment of a number to a minigrid, a clone may be converted to a solitary number and a triad may be converted to a clone. Now we discuss the proposed approach of *ET-I* as follows.

A. Elimination Technique-I (ET-I)

Most Sudoku puzzles can be solved by a process of elimination. In this technique, first we scan every column of the puzzle, then every row and minigrid. The values already present in the cell are kept aside. From the list of unassigned value, if we can get one unique value, satisfying the constraint of Sudoku puzzle, we assign that value for a particular cell. Otherwise, we list out the possible values for an empty cell. The method is expressed in the form of an algorithm below.

Elimination Technique-I (ET-I)

Scan each cell of column number 1 through 9
For each cell
Set possible values for each cell to 123456789;
Eliminate the values already present;
Scan each cell of row number 1 through 9
For each cell
Set possible values for each cell to 123456789;
Eliminate the values already present;
If there is only one possible value for the cell, the value is assigned to the cell;
Otherwise, list down the possible values for the cell until no more cells can be confirmed.

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

Figure 5. A given Sudoku puzzle

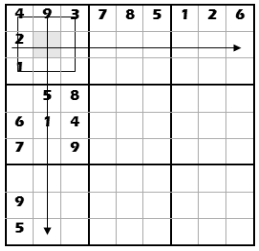

Figure 6. Scanning by column, row, and minigrid

Now we consider an example to explain *ET-I*. Figure 5 shows a partially filled Sudoku puzzle. Scanning each cell from left to right, and then from top to bottom, the first empty cell we encounter is (2,1). Examining the column, that is the second column, we find the possible values left for this cell are 2, 3, 4, 6, 7, 8, and 9. However, scanning the row, that is the first row, we find the only possibility is 9, because all the other values have already been present there.

Continuing with the scanning, the next empty cell is (2,2); see Figure 6. Scanning by its column and row, the possible values obtained are 3, 4, 6, 7, and 8; any one of them can be placed over there. By scanning the minigrid next we see that it already has the values of 1, 2, 3, 4, and 9, so the possible values are now reduced to 6, 7, and 8. Hence

4	9	3	7	8	5	1	2	6
2	678	567	13469	13469	13469	345789	345789	345789
1	678	567	23469	23469	23469	345789	345789	345789
3	5	8	12469	124679	124679	24679	14679	12479
6	1	4	23589	23579	23789	235789	35789	235789
7	2	9	134568	13456	13468	34568	134568	13458
8	3467	1267	1234569	12345679	1234679	2345679	1345679	1234579
9	3467	1267	1234568	1234567	1234678	2345678	1345678	1234578
5	3467	1267	1234689	1234679	12346789	2346789	1346789	1234789

Figure 7. Possible values for empty cells of a Sudoku puzzle.

We cannot achieve a unique value for this position from the present status of the puzzle; we just list out the possible values for the same.

In this way, scanning through each column, row, and minigrid, we ultimately list out the possible values for each empty cell, as shown in Figure 7.

At this stage of the algorithm, when multiple possible values are present for an empty cell in the puzzle, we can eliminate some of the possibilities already computed above by applying the following elimination techniques one after another:

i) Solitary number based,
ii) Clone based, and
iii) Triad based.

Finally, the puzzle can be solved by applying all these elimination techniques repeatedly.

B. Elimination Technique-II (ET-II): Solitary Number based Elimination

Solitary number is a term that is used to refer to a number that is one of multiple possible values for a cell but appears *only once* in a row, column, or minigrid. To see what this means in practice, consider the row shown in Figure 8. In this row, six cells have already been filled in, leaving three unsolved cells (second, eighth, and ninth) with their possible values written in them (derived after applying *ET-I*). Notice that the second cell is the only cell that contains the possible value 8. Since no other cells in this row can possibly contain the value 8, this cell can now be confirmed with the value 8. In this case, the 8 is known as a solitary number. Stepwise this method is as shown below.

1	348	5	9	6	7	2	34	34

Figure 8. An example row of the Sudoku puzzle with a solitary number.

Elimination Technique-II (ET-II)

Scan every cell of each minigrid number 1 through 9
For each cell
Find the solitary numbers and place them in respective minigrids;
Apply ET-I again, if any change is made;
Scan every cell of row number 1 through 9

For each cell
Find the solitary numbers and place them in respective minigrids;
Apply ET-I again, if any change is made;
Scan every cell of column number 1 through 9
For each cell
Find the solitary numbers and place them in respective minigrids;
Apply ET-I again, if any change is made until no more changes can be confirmed.

6	245	7	2389	2389	1	345	345	345
245	8	245	7	23	23	1345	6	9
9	3	1	6	4	5	2	8	7

Figure 9. Possible values for the cells after applying *ET-I* for a portion of the Sudoku puzzle.

Solitary numbers are extremely useful for solving a Sudoku puzzle. Figure 9 shows a partial Sudoku puzzle after applying *ET-I*. One interesting observation is found by looking at the third minigrid. If we observe cell (7,2), one of the possible values is 1, along with the other numbers like 3, 4, and 5. However, the number 1 appears as a possible value only for (7,2) and not for the other cells within the minigrid. Logically, we can now conclude that *as long as a number appears only once (as a possible value) within the minigrid, that number can be confirmed as the number for the cell*. This is logical, because cells (7,1), (8,1), and (9,1) do not contain the value 1, and hence only (7,2) can contain 1. Following this argument, we can now put a 1 in (7,2), as shown in Figure 10.

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

Figure 10. Confirming the value for cell (7,2).

C. *Elimination Technique-III (ET-III): Clone based Elimination*

If two same possible values are present for two cells in a Sudoku puzzle, they are referred as *clone*. Consider the partially solved Sudoku puzzle after applying all those previously stated techniques shown in Figure 11. Observe the two cells (5,2) and (6,2). They both contain the value 23 (means either 2 or 3). So, if cell (5,2) takes the value 2, then cell (6,2) must contain 3, and vice versa. This type of situation is known as clone. Our proposed clone based elimination technique is as shown below.

6	245	7	2389	2389	1	345	345	345
245	8	245	7	23	23	1	6	9
9	3	1	6	4	5	2	8	7
1	24	6	5	23	7	8	9	234

Figure 11. A partially solved Sudoku puzzle with the possible values for empty cells.

Elimination Technique-III (ET-III)

Scan each cell of row number 1 through 9
Find out the clones present in any two cells;
Eliminate the component of the clone find in any other cell;
Scan each cell of minigrid number 1 through 9
Find out the clones present in any two cells;
Eliminate the component of the clone find in any other cell;
Scan each cell of column number 1 through 9
Find out the clones present in any two cells;
Eliminate the component of the clone find in any other cell;
Now apply ET-II and ET-I once again for each cell in the grid until no more cells can be confirmed.

6	245	7	2389	2389	1	345	345	345
245	8	245	7	23 23		1	6	9
9	3	1	6	4	5	2	8	7
1	24	6	5	23	7	8	9	234

Figure 12. Identification of clones.

6	245	7	2389	2389	1	345	345	345
245	8	245	7	23	23	1	6	9
9	3	1	6	4	5	2	8	7
1	24	6	5	23	7	8	9	234

Figure 13. Elimination of 2 and 3 as possible values for other cells in the same row as clones.

Figures 12 and 13 show identification of clones and elimination based on clones in the second row, respectively. As shown in Figure 13, we can eliminate 2 and 3 as possible values for cells (1,2) and (3,2) in the row, as clones are found in cells (5,2) and (6,2), as shown in Figure 12. Now we apply the same elimination techniques for minigrids as well as for columns, respectively, as shown in Figures 14 and 15. As shown in Figure 14, that clones are found in cells (5,2) and (6,2), we can eliminate 2 and 3 as possible values for cells (4,1) and (5,1) for the second minigrid. Similarly, 2 and 3 can be eliminated as possible values for cells (5,5), (5,6), (5,7), (5,8), and (5,9), as shown in Figure 15.

6	245	7	89	89	1	345	345	345
45	8	45	7	23	23	1	6	9
9	3	1	6	4	5	2	8	7
1	24	6	5	23	7	8	9	234

Figure 14. Elimination of 2 and 3 as possible values for other cells in the same minigrid.

6	245	7	89	89	1	345	345	345
45	8	45	7	23	23	1	6	9
9	3	1	6	4	5	2	8	7
1	24	6	5	23	7	8	9	234
				1 8 8 6 89				
				1 8 8 6 89				
				1 8 8 5 6789				
				1 8 8 5 6789				
				1 8 8 5 6789				

Figure 15. *Scanning the columns for the clones and* elimination of numbers based on them.

D. Elimination Technique-IV (ET-IV): Triad based Elimination

If three cells are spotted with a set of three same possible values, they are referred as *triads*. Like clones, triads are also useful for eliminating some other possible values for the cells. Triads have several variations like the following.
Variety# 1: Three cells with same three possible values, as shown in Figure 16.
Variety# 2: Two cells with same three possible values and the other cell containing any two of them, as shown in Figure 17.
Variety# 3: One cell with three possible values and two cells containing two different subsets of two possible values of the earlier three values, as shown in Figure 18.

456	456	456
×	×	×
×	×	×

Figure 16. Variety# 1 triad.

456	456	46
×	×	×
×	×	×

Figure 17. Variety# 2 triad.

45	46	456
×	×	×
×	×	×

Figure 18. Variety# 3 triad.

Our proposed triad based elimination technique is as shown below.

Elimination Technique-IV (ET-IV)

Scan each cell of minigrid number 1 through 9
Find out the triads present in any three cells;
Eliminate the components of the triads found in any other cell;
Scan each cell of row number 1 through 9
Find out the triads present in any three cells;
Eliminate the components of the triads found in any other cell;
Scan each cell of column number 1 through 9
Find out the triads present in any three cells;
Eliminate the components of the triads found in any other cell;
Now apply ET-III, ET-II, and ET-I once again for each cell in the grid, until no more cells can be confirmed.

Figures 19 and 20 show triads and their elimination, respectively. As shown in Figure 19, triads are present in cells (1,8), (1,9), and (2,9). So we can eliminate the values belonging to those triads from cells (2,7), (3,7), and (3,8); see Figure 20.

1	246789	246789	
246	3	246789	
246	246	5	

Figure 19. A partial Sudoku puzzle with triads

1	246789	246789	
246	3	246789	
246	246	5	

Figure 20. Elimination based on triads.

E. Complexity Analysis

An $n^2 \times n^2$ Sudoku grid (consisting of $n \times n$ blocks) is an NP-complete problem [3]. So, development of a heuristic algorithm is the only way out. In this paper we have developed a heuristic algorithm that follows a set of organized moves to solve a given Sudoku puzzle. Needless to mention that the space complexity of the algorithm is $O(m^2)$, as a constant number of elimination techniques has been adopted on the given grid structure of size $O(m^2)$, where $m = n^2$. On the other hand, for each of the $O(m^2)$ cells $O(m)$ alternatives are applied to find out the desired unique entries, if any, in the worst case. This results the overall time complexity of the algorithm is $O(m^3)$, where $m = n^2$.

IV. EXPERIMENTAL RESULTS

In this paper, we have developed a systematic heuristic approach to solve Sudoku puzzle. Incidentally, there are several existing Sudoku solvers, like versions V2.0, V1.01, etc. In doing experimentation, we have executed all these including the algorithm developed by us for a valid set of 60 Sudoku puzzles of different difficulty levels using a platform of Pentium Dual Core 2.40 GHz Processor with 1.0 GB RAM in Java, which results are shown in Figure 21 using bar charts; note that the average computation time (in milliseconds) taken using our approach is almost one-third to that taken by Sudoku solver V1.01 and half to that of Sudoku solver V2.0. So, on an average we can state that our Sudoku solver is much faster to each of such solvers (like V2.0 and V1.01) to solve a valid Sudoku puzzle.

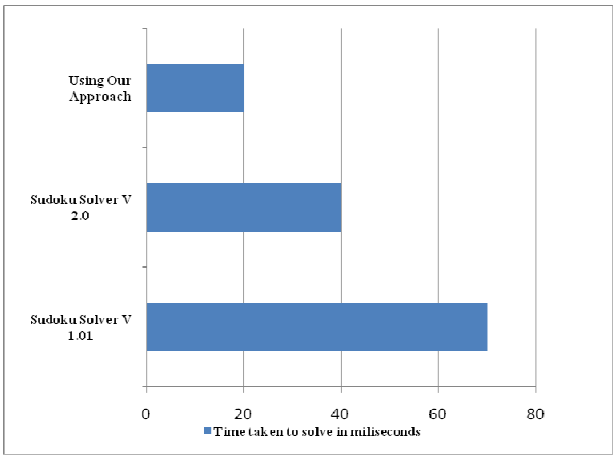

Figure 21. Average times (in milliseconds) taken in the experimentation to solve a set of 60 valid Sudoku instances using Sudoku solvers V2.0 and V1.01, and using our approach.

V. CONCLUSION

The Sudoku is an NP-complete problem. So, developing heuristic algorithms is the only way out. There are several existing solvers for solving the problem including backtracking; the algorithm is highly time consuming. In this paper we have developed an organized method to solve the problem. It takes $O(m^3)$ time and $O(m^2)$ space, where $m = n^2$, for solving a Sudoku grid of size $n^2 \times n^2$ (consisting of $n \times n$ blocks). Our proposed algorithm solves a $9(3^2) \times 9(3^2)$ Sudoku puzzle for all difficulty levels, and the results obtained using our approach is highly interesting in terms of computation time in comparison to that of other existing Sudoku solvers.

In future our objective is to further tune and generalize the algorithm developed in this paper for any larger values of *n*, and execute thousands of instances using the newly developed version of the algorithm. In addition, we like to implement other existing Sudoku solvers and algorithms including backtracking to compare the results obtained in terms of CPU time and required memory for any generalized Sudoku puzzle.

We have calculated the difficulty of the puzzle based on the number of empty cells. But the difficulty level does not always depend on that; it depends on the positioning of the empty grids too. There exist many easy to solve grids with 17 givens only, whereas really hard to solve grids can exist with more than 30 givens. In future, we also like to make all these issues explicit.

VI. REFERENCES

[1] Berthier D., The Hidden Logic of Sudoku, Second Edition, Lulu, Morrisville, France, 2007.

[2] Lee W.-M., Programming Sudoku, First Edition, Apress Inc., New Jersey, USA, 2006.

[3] Narendra J., A to Z Sudoku, Second Edition, ISTE Limited Publications, United Kingdom, 2007.

[4]http://www-imai.is.s.u-tokyo.ac.jp/~yato/data2/MasterThesis.pdf.

[5] http://www.csse.uwa.edu.au/gordon/sudokumin.php.

[6] http://www.en.wikipedia.org/wiki/Pentomino

[7] http://mathworld.wolfram.com/Heptomino.html

[8] http://www.sudoku.org.uk/PDF/Dion_Cube.pdf

[9] http://www.afjarvis.staff.shef.ac.uk/sudoku/sudoku.pdf

[10]http://www.userweb.cs.utexas.edu/~kuipers/readings/Sudoku-sciam-06.pdf

[11] http://www.goduku.com

[12] http://www.wordoku.biz

[13] http://www.en.wikipedia.org/wiki/Tetris