A Novel Sudoku Solving Technique using Column based Permutation

Sunanda Jana
Dept. of Computer Sc. & Engg.
Haldia Institute of Technology line
Haldia, India
sunanda_jana@yahoo.com

Arnab Kumar Maji
Dept. of Information Technology
North Eastern Hill University
Shillong, India
arnab.maji@gmail.com

Rajat Kumar Pal
Dept. of Computer Sci. & Engg.
University of Calcutta
Kolkata, India
pal.rajatk@gmail.com

***Abstract*—"Sudoku" is the Japanese abbreviation of "Suuji wa dokushin ni kagiru", which means "the numbers must occur only once". It is a challenging and interesting puzzle that trains our mind logically. In recent years, solving Sudoku puzzles has become a widespread phenomenon. The problem of solving a given Sudoku puzzle finds numerous applications in the domain of Steganography, Visual Cryptography, DNA Computing, Watermarking, etc. Thus, solving the Sudoku puzzle in efficient manner is very important. However, incidentally all the existing Sudoku solving techniques are primarily either guess based heuristics or computation intensive soft computing methodologies. They also solve the puzzle by traversing on each and every individual cell. In this paper, a novel Sudoku solving technique is proposed which solves Sudoku puzzles without guessing a cell and by generating only the desired permutations among columns, which consist of only groups of cells.**

Keywords—Algorithm; backtracking; cell; column; difficulty level; permutation; sudoku

I. INTRODUCTION

A Sudoku puzzle can be characterized as an in part finished N×N grid where the at first characterized qualities are known as givens or clues [5]. A general Sudoku puzzle has nine rows, columns, and minigrids, every having nine cells just. Along these lines, the full grid has 81 cells. Rows, columns, and minigrids are all things considered alluded to as units or extensions. The one decide that can be connected to this puzzle is 'Every digit shows up once in every unit'. Size alludes to the extent of a puzzle or grid. For characterizing the puzzle's span, regularly a composite of 'number of rows times number of columns', assignment is utilized, i.e. size 9×9. The fantastic 9×9 Sudoku configuration can be summed up to a N×N row, column, and grid apportioned into N areas or minigrids, where each of the N rows, columns, and minigrids have N cells and each of the N digits happens precisely once in every row, column, and minigrid.

There is also no known technique in the literature to figure out the number of possible minimal cells that should be there as given clues, which amounts to the ultimate count of distinct Sudoku puzzle instances. It is a challenge that is sure to be taken up in the near future. According to [9], a Sudoku instance must require as minimum as 17 clues to compute a unique solution, if one exists. That means a Sudoku instance with as many as 16 clues must have at least two solutions, if the given instance is a valid Sudoku instance. However, we have observed that a valid Sudoku instance may generate two or more valid solutions [1], even if the number of clues is 17 or more. Apart from 9×9 Sudoku puzzle, other versions of the Sudoku puzzle also exist. They can be briefly classified as follows:

1. **Sub Doku** [2]: It consists of grids smaller than 9×9; 4×4 variant is known as *Children's Sudoku.*
2. **Super Doku** [2]: It is having the grid larger than 9×9.
3. **Prime Doku** [2]: For *N*×*N* grid, if *N* is prime number, then it is known as Prime Doku.
4. **Maximum Su Doku** [2]: If a Sudoku puzzle is having maximum number of independent clues which gives ua a complete and unique solution, then it is known as Maximum Su Doku.
5. **Minimum Su Doku** [2]: If a Sudoku puzzle have the minimum number of that can frame a complete and unique solution.
6. **Proper puzzle** [2]: It is having a unique solution.
7. **Satisfactory puzzle** [2]: If a Sudoku puzzle doesnot require a trial and error method for solving it, then it is known as satisfactory puzzle.
8. **Purely numeric puzzle** [2]: It is a puzzle which can be filled up with numbers only.
9. **Purely literal puzzle** [2]: If instead of numbers, characters are used then it will be known as literal puzzle.
10. **Numeroliteral puzzle** [2]: If combinations of number and literals are used, then only it is known as Numeroliteral puzzle. It is usually seen in 12×12 Sudoku puzzles.
11. **Jigsaw Sudoku** [2]: In Jigsaw Sudoku the constraints for row and column are applied but instead of minigrids it is having grids of Jigsaw shapes.

In our proposed methodology, we have divided a 9×9 grid into nine columns, each having nine rows as shown in Figure 1. Then we have generated permutations among the missing elements in columns based on given clues. Then based on valid permutations amongst columns, the final solution of the Sudoku puzzle has been generated.

II. Literature Survey

In 1979, Howard Garnes, a freelance puzzle constructor, first created a Sudoku puzzle, and it was first published in New York under the title *Number Place*. In 1984, this puzzle was consequently introduced in Japan in the paper *Monthly Nikolist* by Nikoli as "Suuji wa dokushin ni kagiru", which can be translated as "the numbers must occur only once". Later on, the name was abbreviated as Sudoku.

According to Wikipedia [2], Sudoku is a Japanese logical game that has recently become hugely popular in Europe and North-America. Various heuristic algorithms have been used for solving Sudoku puzzles as Sudoku is an NP-complete problem [4]. In literature, Sudoku instances are accessible in view of their grouping of diverse levels of difficulty, as simple, moderate, diabolical, and so on. Now and again we are considering an instance as easier or harder taking into account the quantity of clues given alongside their relative areas in a given case, despite the fact that to bolster such claims there is no verification. By chance, the Sudoku solver grew in this does not separate the examples as far as any level of difficulty, and each of the occasions is similarly simple or difficult to fathom utilizing our methodology, if a sensible number of clues are given.

Table 1 demonstrates an examination outline of the quantity of clues for distinctive difficulty levels [5]. Presently the thing is that the quantity of clues as well as the position of each of the vacant cells additionally influences the level of difficulty. In the event that two puzzles have the same number of clues toward the start of a Sudoku amusement, the puzzle with the givens (or clues) in groups is reviewed in more elevated amount than that with the givens scattered over the space. The lower bound on the quantity of clues is controlled in every line and segment for every difficulty level [5] taking into account the line and section limitations as appeared in Table 2.

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)	36-45
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

The major strategy that has been executed for unraveling Sudoku puzzles is backtracking [6]. It fills in as takes after. The technique spots number 1 in the first exhaust cell. On the off chance that the decision is perfect with the current clues, it proceeds to the second purge cell, where it puts a 1 (in some other line, segment, and minigrid). When it experiences a contention (which can happen rapidly), it eradicates the 1 simply put and embeds 2 or, if that is invalid, 3 or the following lawful number. In the wake of setting the first lawful number conceivable, it moves to the following cell and begins again with a 1. In the event that the number that must be changed is a 9 (which can't be brought by one up in a standard Sudoku matrix), the system backtracks and expands the number in the past cell (the by last number set) by one. At that point it advances until it hits a contention.

In this manner, by utilizing this system, before propelling the method might in some cases backtrack a few times. It is ensured to discover an answer if there is one, basically in light of the fact that it in the long run tries every conceivable number in every conceivable area. Aside from backtracking, some different procedures included:

1. *Elimination based technique* [5] and
2. *Soft computing based technique* [6]

In elimination based approach, a list of possible values for every blank cell is first obtained based on the given clues. Then the following different techniques can be applied to them to eliminate the possibilities of each cell. These techniques can further be classified [7] as follows:

a. Easy Techniques: i) Single Position, ii) Single Candidate Pencil marks.
b. Medium Techniques: i) Candidate Line, ii) Double Pair, iii) Multi-Line.
c. Advanced Techniques: i) Naked Pairs / Triplets ii) Hidden Pairs / Triplets.
d. Master Techniques: i) X-Wing, ii) Swordfish, iii) Forcing Chains.
e. Harder Style: i) Nishio, ii) Guessing.

We are initially numbering each of the minigrids of a 9×9 Sudoku instance as 1 through 9 shown below.

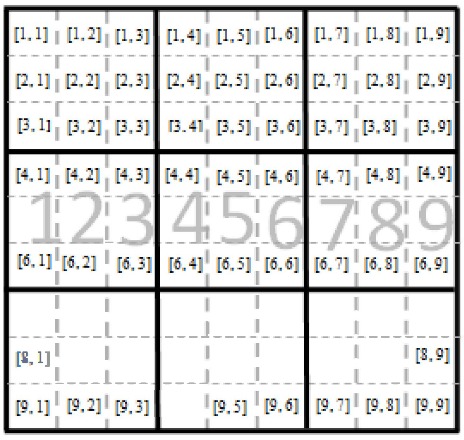


Fig. 1. The structure of a 9×9 Sudoku puzzle (problem) with its nine columns of 9×1 (9 rows for each of the columns) each as numbered 1 through 9. Column number 1 consists of the cell locations [1, 1], [2, 1], [3, 1], [4, 1], [5, 1], [6, 1], [7, 1], [8, 1], and [9, 1], column number 2 consists of the cell locations [1, 2], [2, 2], [3, 2], [4, 2], [5, 2], [6, 2], [7, 2], [8, 2], and [9, 2], and so on.

Single Position: This is the least demanding and a standout amongst the most prevalent procedures which individuals utilize first when finishing Sudoku puzzles. Pick a row, column, or minigrid, and afterward experience each of the numbers that has not as of now been put. In view of different situations, the positions where we could put that number will be constrained. Regularly there will be a few places that are legitimate, yet in the event that we are fortunate, there may be one. On the off chance that we have narrowed it down to one and only legitimate spot where we can put the number, then we can fill that number straight in.

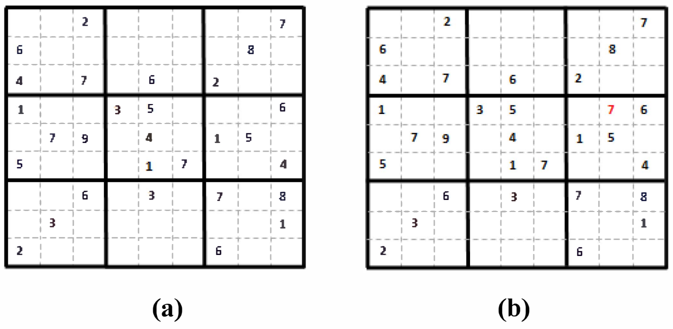


Fig. 2. (a) An instance of the Sudoku problem. (b) An example row (fourth row) of a Sudoku puzzle with a single position, where missing digits are 2, 4, 7, 8, and 9. To find a single position for 7 in this row we start with the left hand minigrid, there is already a 7 in positions [5,2] and [3, 3], which prevent from introducing a 7 in either position [4,2] or [4,3]. Again there is a 7 in position [6,6], that prevents a placement of 7 in the central minigrid. Again in position [4,7] 7 cannot be placed, as 7 is there in position [7,7]. Thus, there is only one place left for 7 in row 4, i.e. position [4,8].

As an example, consider the fourth row of the puzzle in Figure 2(a); —where to put 7 in that row? Starting with the left hand minigrid, there are already 7 in positions [5,2] and [3,3], that prevent us in adding 7 in positions [4,2] and [4,3], respectively. Again there is a 7 in position [6,6], and that prevents the placement of 7 in the central minigrid. Again, in position [4,7], a 7 cannot be placed as 7 is already there in position [7,7]. Thus, there is only one place left for the 7 in the fourth row, i.e. position [4,8].

Single Candidate: This strategy is generally easier; particularly, on the off chance that we are utilizing pencil imprints to store what competitors are still conceivable inside of every cell. On the off chance that we have figured out how to discount every other possibilities for a specific cell (by looking at the encompassing segment, line, and minigrid), so there is stand out number left that could fit there, and after that we put that number there. Investigating segment 3, the main two missing numbers are 2 and 1. 2.

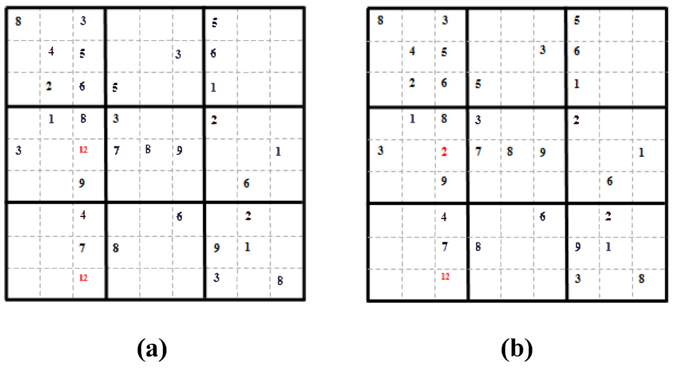


Fig. 3. (a) An instance of the Sudoku problem. (b) An example column (third column) of the puzzle with single candidate, where the missing digits are 2 and 1. Now as there is already a 1 in position [5,9] so the only possible candidate in cell [5,3] is a 2; so we can fill it by **2.**

Candidate Lines: This is the first technique which does not actually tell us where to place a number, but instead helps us to determine places where we cannot place a number. If pencil marks are used, then this information helps us to remove candidates, and from there we should be able to make placements.

Consider the third minigrid of the puzzle in Figure 4. There are only two places where 4 can be placed, and they are in a column. This means that a 4 on this column must be in that minigrid, and cannot be anywhere else on this column.

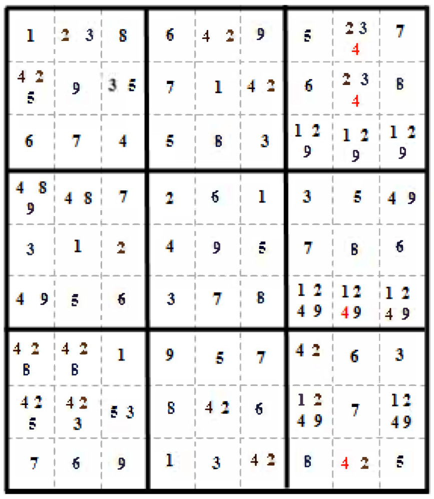


Fig. 4. An instance of a Sudoku puzzle with candidate lines in column eight.

Double Pairs: This method depends on spotting two sets of contender for a value, and utilizing these, guidelines out applicants from different minigrids. For instance, 2 is there at the center segment of minigrids and they just lie along two lines (columns 4 and 6). Presently as the 2's are restricted to these positions in minigrids 5 and 8, it implies that the coulmns 4 and 6 are likewise taken. This implies that any of the contender for 2 in minigrid 2 can be expelled from both of these two columns.

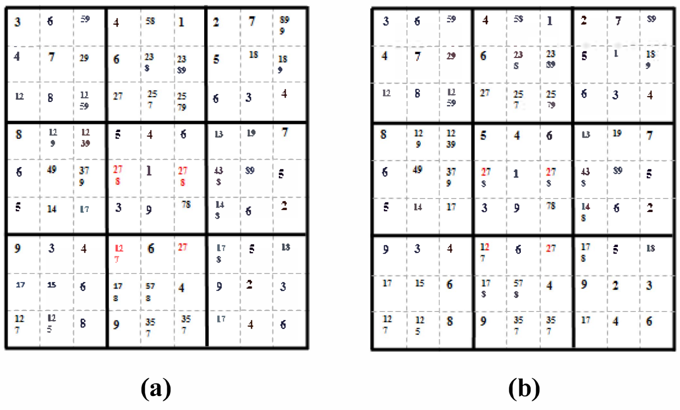


Fig. 5. (a) An instance of the Sudoku problem. (b) Two example minigrids 5 and 8 of this Sudoku puzzle contain double pairs comprising 2 as a probable member, and it only lies along two lines (columns 4 and 6). Now as the 2's are limited to positions [7,4], [7,6], [5,4], and [5,6] in minigrids 5 and 8, the candidates for 2 in minigrid 2 can be removed from either of these two columns.

Multiple Lines: This is very like Double Pairs, however minimal harder to spot. It lives up to expectations in the same way yet the applicants that involve the lines could be spread crosswise over two of the pieces, and there could be a few competitors in every line..

Nishio: "Nishio" intends to discover a cell with only a few applicants, and to pick one of the candidates taking into account speculating. At that point the method wipes out that candidate from the same line, section, and minigrid. Subsequent to disposing of the speculated applicant if the riddle ends up being reasonable, then our conjecture is correct. On the off chance that our supposition isn't right, maybe it would demonstrate to itself up inside only a couple more moves, or perhaps it would take right until the very end before the last number is contradictory. Discovering a contradictory rapidly implies we can erase that alternative, and realize that the other decision was right.

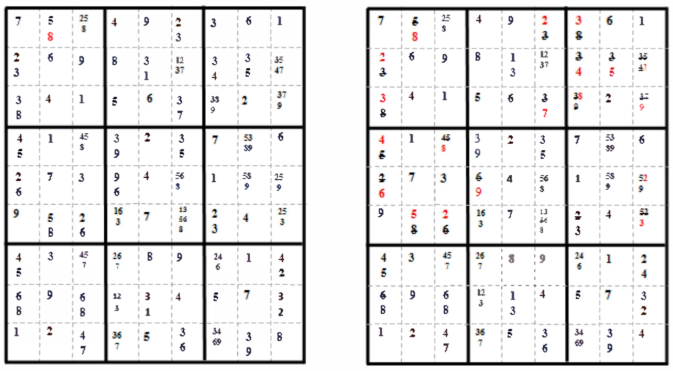

(a) (b)

Fig. 6. (a) An instance of the Sudoku problem. (b) An example cell position [1,2] of a Sudoku puzzle with Nishio, where considering 8 as candidate for that cell (having position [1,2]) and then follow through the values it would force into the nearby cells by crossing out the candidates that are not possible, and highlighting in red the single candidates that would become the new cell contents.

Let us consider the above given puzzle with a cell position [1,2]. Let us further consider the value 8 for that cell, and follow through the values it would force into the nearby cells and so on. Without too much effort we reach this, crossing out the candidates that are not possible, and highlighting in red the single candidates that would become the new cell contents if we follow the results of this 8.

Naked and Hidden Pairs, Triplets: This is one of the cleverer techniques that works by spotting sets of pairs (or triples, or even quads) within an area, i.e. the same two candidates are there in two cells (or, in the case of naked triplets, the same three candidates in three cells, and so forth). A naked pair, triplet, or quad must be in the same virtual line [9]. Figure 7 shows how to use a naked pair. In cells [1,2] and [1,8] the only candidates that appear are a 2 and a 7. Thus, we can say 7 must be in one, and 2 in the other. However, then the 2 and 7 cannot appear in any of the other cells in that row, so 2 can be eliminated as a candidate from locations [1,1] and [1,7], and also 7 can be eliminated as candidate from location [1,1].

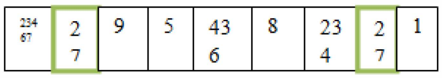


Fig. 7. **Figure 7:** Example cells [1,2] and [1,8], where the only candidates that appear are a 2 and a 7, which construct a naked pair. So, 7 must be in one and 2 in the other. However, then the 2 and 7 cannot appear in any of the other cells in that row, and 2 can be eliminated as a candidate from locations [1, 1] and [1, 7], and also 7 can be eliminated as candidate from location [1, 1].

Like naked pair, for naked triplets, quads, etc. the only requirement is that the three (or four) values be the only values appearing in these squares in some virtual line.

X-Wing: This technique relies on using positions of pencil marks to infer enough to allow us to eliminate some other candidates. X-Wing configuration can be found if candidate's presence can be found exactly two times in two rows and in the same columns of those two rows. In this technique after connecting the possible pairs forms an 'X'. As it looks like X-wing fighters in star wars, so it is called as X-wing.

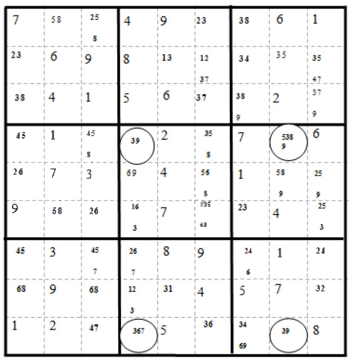


Fig. 8. Candidate 3 occurs exactly twice in rows 4 and 9 and in these two rows, it appears in columns 4 and 8 which construct X-wing.

In the configuration in Figure 8, the candidate 3 occurs exactly twice in rows 4 and 9, and in these two rows, it appears in columns 4 and 8. It does not matter that the candidate 3 occurs in other places in the puzzle.

Swordfish: Swordfish is a type 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.

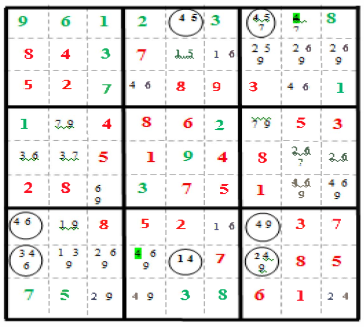


Fig. 9. A Sudoku puzzle with swordfish occurred at the first, fifth, and seventh column.

We can easily see the existing techniques discussed in literature survey are based on guessing. Apart from these techniques other techniques include soft computing based practices, which include Particle Swan Optimization [9], An Colony Optimization [9], Bee Colony Optimization [5], and so on. However, all this techniques are exhaustive in nature and computationally expensive.

III. THE PROPOSEDWORK

All the techniques that are discussed in review are cell based and some measure of speculating is constantly included in every one of the systems. In this paper, an endeavor has been made to add to a calculation which is column based, i.e. we need to exclusively experience nine columns (rather than 81 cells) and perform backtracking just on them, which is less tedious. In addition, no speculating is included and no excess calculation is performed amid the entire calculation. Here "speculating" means attempting to expect a plausible applicant and perceive how far it goes, if a contention happens with the estimate, backtrack from the speculating step and strive for another hopeful; when all competitors are depleted without achievement, backtrack to the past speculating step (if there is one; generally, the riddle demonstrates invalid), and so forth. In our proposed work, we have added to a novel column based speculated free Sudoku solver by executing the idea of stage tree by which we can produce just legitimate changes for each of the succeeding segments of a given Sudoku confound, and settle it thusly, in the event that it has a solution.

A. *Permutation:*

Permutation is a way of ordering of a set of objects or symbols where each object occurs exactly once. As an example, there are two permutations of the set of numbers {1,2}, namely (1,2), (2,1). The number of permutations of k distinct objects can be calculated as $k\times(k-1)\times(k-2)\times \ldots \times2\times1$, which is mainly denoted as "factorial of k" and written as "$k!$".

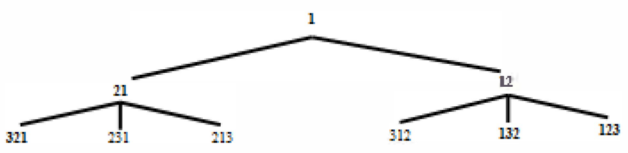


Fig. 10. The permutation tree [15] for three natural numbers 1, 2, and 3.

In the existing literature, there are several permutation generation techniques, like Random Permutation Generation Algorithm [11], Fisher-Yates Shuffle [12], Restricted Permutations [13], Permutation Tree [14, 15], etc. Permutations can be represented by using a tree structure called permutation tree. If S_n is the set of all permutations of set $I = \{1, 2, \ldots, n\}$ (i.e., the set of first n natural numbers), then every permutation in S_n can be represented by a permutation tree. So, we can say that a permutation tree must correspond to a set of all permutations. Various works has been done in the past to present to produce permutations with the assistance of such trees.

In our proposed work we have developed a novel algorithm where by using permutation trees, we have generated all valid permutations only for each of the succeeding columns of a given Sudoku puzzle, and solve it subsequently.

B. *Column-Permuted Sudoku Solver:*

Below we find our proposed algorithm at a glance:

Algorithm: This is a Guessed Free Sudoku Solver that can solve any Sudoku Instance, if it has a valid solution.

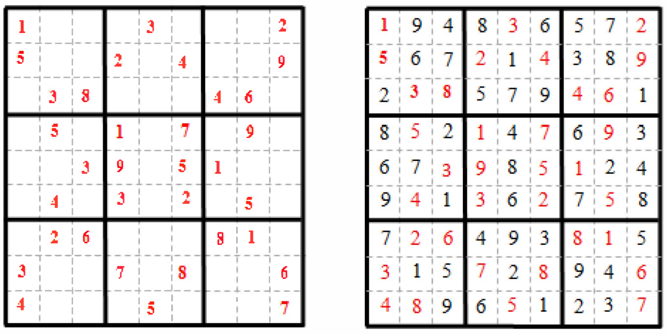
Input: A Sudoku instance, I of size 9×9.

Output: A solution, S of the given Sudoku instance, I.

Data Structure used: Tree structure.

Step 1: Find out missing digits, the number of givens, in each of the columns of I.

Step 2: Process S_C, an arrangement of columns that contains every one of the columns in progression, wherein $C \in S_C$ is the column (and the first part in S_C) with a greatest number of pieces of information and whose column number is least. That implies, in S_C, a part C is a column that contains a most extreme number of intimations and whose column number is least, where $1 \leq C \leq 9$.



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

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

Fig. 11. (a) An instance of the Sudoku problem. (b) A solution of the Sudoku instance shown in 10(a). The proposed algorithm considers each of the columns 1 through 9 as shown in Figure 1. Each column may or may not have some clues as numbers that are given. We first consider a column that contains a maximum number of clues, and if there are two or more such columns, we consider the one with the least column number.

Step 3: Figure every single valid permutation for the missing digits in C, and store them.

Step 4: For the whole remaining columns in progression in SC do the accompanying:

(i) Consider a next column, $N \in S_C$, and register all its substantial permutations for the missing digits in N accepting a legitimate permutation for each of the prior columns up to M, and store them.

(ii) If one legitimate permutation for N is acquired, then consider a next column of N in S_C, if any, and process all its substantial permutations for the missing digits in this column expecting a legitimate permutation for each of the prior columns in S_C, and store them.

Else consider a next legitimate permutation, if any, of the quickly past column of N, and figure all its substantial permutations for the missing digits in N expecting a legitimate permutation for each of the prior columns in SC, and store them.

Step 5: In the event that all the substantial permutations of the prompt successor column of C are depleted to acquire a legitimate mix for all the nine columns in S_C, then consider a next substantial permutation of C and go to Step 4. The procedure is proceeded until a legitimate blend for all the nine columns in S_C is gotten as a fancied arrangement S for I; generally, the calculation announces that there is no substantial answer for the given occurrence I.

We first consider column 4 as both the columns 4 and 6 contain a maximum number of clues (which is five); we may consider any one of them but according to our algorithm we consider the column bearing minimum column number. In this column, the missing digits are 4, 5, 6, and 8. Thus, with the help of permutation tree, the possible valid permutations for column 4 are as shown in Figure 11.

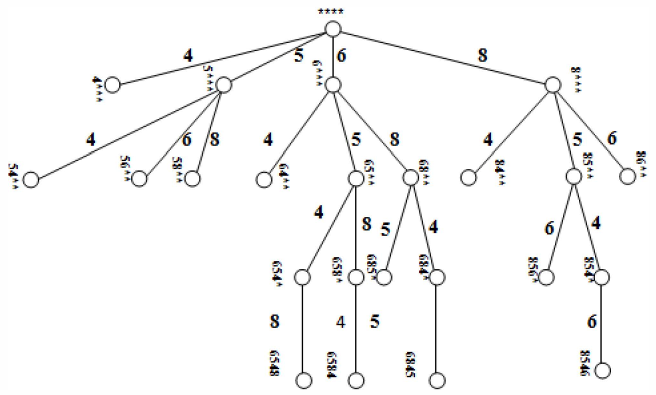


Fig. 12. The permutation tree for generating only valid permutations of the missing digits in column 4 of the Sudoku instance shown in Figure 10(a).

The proposed algorithm considers each of the columns of a given Sudoku puzzle. Each column may or may not have some clues. So, first we have to consider a column that has the maximum number of clues, and if there are several such columns, we consider the one with the least column number. Here columns 4 and 6 contain the maximum number of clues. Hence, at the beginning, we consider column 4 for computing all its valid permutations of the missing numbers (for its blank locations).

Here we mean a cell area of a Sudoku occurrence by [row number, column number]. Thus, the blank locations are [1,4], [3,4], [7,4], and [9,4], and the missing digits are 4, 5, 6, and 8. Here as the quantity of blank location is four, the aggregate number of permutations is 4!, which is equivalent to 24. Presently the calculation considers each of these permutations consistently and distinguishes just the substantial arrangement of permutations taking into account the given hints accessible in different lines, columns, and minigrids. With respect to case, in the event that we consider the last permutation 4568 and as needs be place the missing digits all together in areas [1,4], [3,4], [7,4], and [9,4], which are orchestrated in climbing request, we can find that this permutation is not a substantial permutation. This is on the grounds that the eighth minigrid as of now contains a 8 as a clue in column 6 at area [8, 6]; along these lines, we can't put 8 at area [9,4] as the permutation recommends..

To figure just the valid permutations for the missing digits in column 4, we build a tree structure. Here the missing digits are 4, 5, 6, and 8. The proposed calculation likes to put each of the permutations of these missing digits in the blank areas are [1, 4], [3, 4], [7, 4], and [9, 4]. Actually, as the tree's base structure does not contain any permutation of the missing four digits, it is spoken to by '****'. This root is having four children where the first child prompts produce every single valid permutation gazing with 4, the second child prompts create every single valid permutation gazing with 5, and so on. We extend the tree structure embeddings another missing number at its separate area (for a blank cell) driving from a valid permutation (as vertex) in the past level of the tree. Correspondingly, whether the missing digit could be set at the specific area for a blank cell of the given Sudoku occurrence I or not those we need to confirm. On the off chance that the answer is 'yes', we further extend the vertex. Else, we quit extending the vertex in some prior level of the tree structure before the last level of valid leaf vertices just.

For instance, the vertex with permutation "64**" is not expandable, in light of the fact that we can't put 4 at [3,4] as area [3,7] contains 4 as provided insight. Subsequently, this is the way either a valid permutation is produced from the tree's base structure coming to a bottom-most leaf vertex, or the procedure of extension is ended in some before level of the tree that must create other than valid (or undesirable) permutations as of right now. Number of conceivable permutations of four missing digits is 24, and out of them one and only permutation is valid for column 4 of the Sudoku occasion.

Note that the given clues in I are only imperatives and we should comply with each of them. Along these lines, typically, if there are more clues, I is more obliged and subsequently the quantity of valid permutations is even considerably less, and the arrangement, if one exists, is one of a kind in a large portion of the cases. In actuality, if there are less clues in I, more valid permutations for some column of I could be produced; calculation of an answer for I may take additional time. Regardless, if there is a given's answer Sudoku occurrence, stand out of a few valid permutations (if any) for each of the columns will be chosen taking after the resulting strides of the calculation.

To figure out the following column to be considered, we experience the column in the Sudoku example, and among these columns we find that column 6 contains a greatest number of clues, i.e. 5. Presently the calculation thinks of one as valid permutation of column 4 and creates permutations for column 6 in light of the remaining clues in I, and produces every valid permutation for column 6. In the event that no less than one valid permutation for column 6 is gotten, we continue for producing every single valid permutation for column 1 complying with every provided insight in I and the accepted valid permutations of columns 4 and 6. Something else, a second valid permutation of column 6 is considered, for which similarly, we create every valid permutation for column 1 and so on.

We consider column 1 (as the following likely column) for registering all its valid permutations taking into account one valid permutation of column 6, and afterward one ensuing valid permutation of column 2, as each of the columns 2, 5, 8, and 9 contains same number of clues. Along these lines, we consider that column which is having less column number. This is the means by which the calculation continues and creates every single valid permutation of a column under thought complying with the given clues in I and an arrangement of accepted valid permutations, one for each of the columns considered before in progression, as yet in time. In this way, along these lines by utilizing this speculated free Sudoku solver we can comprehend any Sudoku Instance I in a proficient way.

C. *Computational Time and Space Complexity:*

Let K be the quantity of blank cells in a column and the Sudoku occasion is of size $N \times N$. The computational time and space complexity of speculated free Sudoku solver created in this paper gets to be $(K!-x)^n = O(K^n)$, where x is the quantity of other than valid (or undesirable) permutations in view of the clues given in the Sudoku case I. Subsequently, the experimentation made by algorithm take negligible amount of clock time, of the order of milliseconds.

IV. Conclusion

In this paper, we have studied some existing elimination based methods for solving the Sudoku puzzle. Apart from elimination based schemes, soft computing based techniques are also available. We also discussed different techniques of generating permutations of a given set of numbers, in brief, and we used here the technique of generating only valid permutations for each of the successive columns of a given Sudoku puzzle. We know that the problem of solving a Sudoku puzzle is beyond polynomial time computable; hence, developing heuristic algorithms is one of the solving schemes. Our proposed algorithm can successfully solve a given Sudoku puzzle of size 9×9, if it has a solution. The algorithmic technique we use is backtracking, but this time we apply backtracking column-wise instead of individual blank cells, which is extremely time consuming. The heuristic developed in this paper is also applicable for Super Sudoku, Giant Sudoku, or for any large Sudoku instance, where the size of the instance is bounded by some constant.

References

[1] I. Semeniuk, Stuck on Us. *New Scientist*, Vol. 31, pp. 45-47, 2005.

[2] http://en.wikipedia.org/wiki/Sudoku.

[3] F. Sullivan, Born to Compute. *Comput. Sc. Engg.*, Vol. 8, pp. 88-90, 2006.

[4] T. Yato 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(5), pp. 1052–1060, May 2003.

[5] W. M. Lee, *Programming Sudoku*. USA: Apress, 2006.

[6] N. Jussien, *A-Z of Sudoku*. USA: ISTE Ltd., 2007.

[7] http://www.sudokuoftheday.com

[8] A. K. Maji, S. Jana, and R. K. Pal, An Algorithm for Generating only Desired Permutations for Solving Sudoku Puzzle, Vol. 10, *Procedia Technology Journal* (ISSN: 2212-0173), ScienceDirect, Elsevier, doi: 10.1016/j.protcy.2013.12.375, pp. 392-399, Sep. 2013.

[9] T. Davis, “The Mathematics of Sudoku”. http://www.geometer .org/mathcircles, October 12, 2008.

[10] A. Hård, Sudoku and the Minimum Number of Clues. *Bachelor's Thesis*, Computer Science, Stockholm University, Sweden, 2012.

[11] http://www.chris-j.co.uk/randperm.php

[12] Wikipedia: *The Free Encyclopedia*. wikipedia.org/wiki/Fisher-Yates_Shuffle

[13] M. Bona, *Combinatorics of Permutations*. Chapman Hall-CRC, Paris, 2004.

[14] http://www.techuser.net/randpermgen.html

[15] B. J. Arnow, Representation of Permutations, Combinations and Dihedral Elements as Tree. *Journal of Computers, Mathematics and Its Applications*, Vol. 20, No. 3, pp. 63-67, 1990.