Chapter 7

Conclusion and Future Scopes

This chapter concludes the thesis by summarizing the works, findings, and contributions of the thesis. It also gives some directions about the future scopes of possible works in this specified field. In the beginning of the chapter, we have discussed about the problems associated with the earlier Sudoku solvers existing in literature, then in the next section, we have discussed about the summary of the current work. In the later part of this chapter we have discussed about the future directions of potential research.

7.1 Status of the Problems Prior to this Work

All the earlier existing Sudoku solvers that are available in literature (including Internet) are entirely guess based heuristics and hence extremely time consuming. In addition, each of these existing solvers solves an instance of the problem considering the clues one-by-one for each of the blank locations in a given Sudoku puzzle. Often guessing may not be guided by selecting a desired path of computing a solution and hence exhaustive redundant computations are involved over there.

Usually Sudoku instances are available in literature based on their classification of different levels of difficulty like easy, moderate, diabolical, etc. These classifications are based on the algorithmic approaches developed and executed in solving different Sudoku instances. Sometimes, it is told that the instances are easier or harder based on the number of clues given along with their relative locations in a given instance but there is no proof to support such claims. All the existing Sudoku solvers solve the puzzle based on the different difficulty levels. As there is no clear definition of difficulty levels, it is unjustified to implement an algorithm based on the different levels of difficulty.

There was no generalized Sudoku solver that can straightaway be applied for 9×9, 16×16, or 25×25 Sudoku puzzles or more in a modular fashion.

Also, at the time of Sudoku instance generation, it is very much needed to detect whether the generated instance is having a valid solution. If there is any valid solution, it is also necessary to

make sure for multiple solutions, and if any, how many. There was no solver, which can expose whether a Sudoku puzzle is really an instance. If there are more than one solution for a given puzzle, that also cannot be established by any of the existing solvers. Anyways, there were a number of lacunas that we observed in this field of study before we started to work on it.

7.2 Summary of the Work Done

Here, in this section, we briefly discuss the coverage of this thesis; status of the earlier work in the form of literature survey and our contribution that are devised and illustrated in five major chapters excluding the introductory chapter and this one. To the point, these have been highlighted as follows.

In Chapter 2, we have exhaustively studied different existing Sudoku solving techniques. We have found out their relative advantages and shortcomings as well in this chapter.

In Chapter 3 of this thesis, we have developed a guessed free Sudoku solver that naturally has grown in the form of two successive versions, and ultimately a graph theory based generalized Sudoku solver is obtained (in Chapter 4). This Sudoku solver is minigrid based instead of cell based as earlier all the existing Sudoku solvers did.

All the versions of the algorithm, we have developed, are exclusive minigrid based Sudoku solver, which is completely guessed free. The solver considers each of the minigrids (instead of blank cells in isolation) of size 3×3 that has been developed for the first time in designing such an algorithm for a given Sudoku puzzle of size 9×9. In our algorithm, a preprocessing is there for computing only all valid permutations for each of the minigrids based on the clues in a given Sudoku puzzle.

It has been observed in most practical situations that the number of valid permutations is appreciably less than the total number of possible permutations for each of the minigrids, and even if there are less clues in some minigrid of an instance, clues present in four adjacent row and column minigrids severely help in reducing the ultimate number of valid permutations for that minigrid too. Anyways, this approach of minigrid-wise computation of valid permutations and checking their compatibility among row minigrids and column minigrids is absolutely new and done for the first time in this domain of work.

As we consider minigrids for finding only the valid solutions of a given Sudoku puzzle instead of considering the individual (blank) cells, therefore, the computations involved in the algorithm is significantly reduced. In the case of a 9×9 Sudoku puzzle, there are 81 such cells with some clues (which is less) and the remaining blank cells (which is more), whereas there are only nine such minigrids each of which consists of 3×3 cells. Here the observation to a Sudoku puzzle is not by searching of missing numbers cell-by-cell (as if searching for an address by moving on the earth); rather, it is one step above the ground of the puzzle by considering groups of cells or minigrids (as if searching is made from a bird's eye view flying in the sky).

The brilliancy of the algorithm developed in this chapter is that the same logic can also be straightaway applied for larger Sudoku instances such as 16×16, 25×25, or of any other rectangular sizes (with their respective objective functions).

In Chapter 4 of this thesis, we have designed two more versions of the same algorithm in the form of an exclusively graph theoretic Sudoku solver, which is entirely guessed free. The solver considers each of the minigrids and plots a graph structure based on the compatibility amongst minigrids. In the end, a reduced graph structure is obtained based on the constraints given as clues and the objective functions we are supposed to meet, which counts the exact number of valid solutions possible for a given Sudoku puzzle.

In spite of the controversy that a Sudoku puzzle must have only one unique solution, we show that a given Sudoku instance may have two or more valid solutions and it finds applications in different domains as well. The Sudoku solver developed in this chapter (i.e. in Chapter 4) is capable of computing all these solutions, if there exists at least one. Even that, if there is no solution of an assumed Sudoku instance, can also be identified using the same cost of algorithmic time and space.

We have observed that, even if there are more clues, a Sudoku puzzle may produce more solutions. If there are many solutions for some given Sudoku instance, our method does not incur any additional cost for that; all these solutions can be produced at the same time.

The novelty of the algorithm devised herein is that the same logic can also be straightaway applicable for larger Sudoku instances such as 16×16, 25×25, or of any other rectangular size (with their respective objective function). Similar logic of extracting an appropriate graph structure (some sort of induced subgraph as shown in Figure 4.1 for a 9×9 puzzle) based on a

larger Sudoku instance under consideration is to be executed. We may right away guess that the associated tasks for larger Sudoku instances are extremely difficult, tedious, and time consuming that such algorithmic methodology (as developed in Chapter 4) may solve in a reasonable amount of clock time.

The level of difficulty is another important issue that is considered by all the earlier Sudoku solvers while developing a Sudoku algorithm. There are no hard and fast rules that define the difficulty level of a Sudoku puzzle. A sparsely filled Sudoku puzzle may be extremely easy to solve, whereas densely filled Sudoku puzzles are actually more difficult to crack. From programming point of view, we can determine the difficulty level of a Sudoku puzzle by analyzing the effort required to solve the puzzle, and the different levels of difficulty are defined as Easy, Medium, Hard, Evil, etc. However, the Sudoku solver designed in this chapter does not categorize the instances to any level of difficulty; rather, all the Sudoku instances represent the same level of difficulty (if a minimum number of 17 clues are there in a given Sudoku puzzle). This is true for solving a Sudoku puzzle of any level of difficulty. This is because our devised solver follows the same algorithmic technique. In some cases, more valid permutations may be generated for some minigrid, but in general, the number of valid permutations is much less. The minigrids with smaller number of valid permutations indeed lead to compute all the desired solutions for a given Sudoku instance in due course of time.

In Chapter 5, we have studied and explored different existing Sudoku instance generation means, and devised algorithms for doing the same. We have analyzed their relative advantages and limitations though the said means are not explicitly available and portrayed in existing literature. We have also observed that each of those ways likes to check whether a generated Sudoku instance is valid or not. That means whether a created instance has any unique solution. We have already mentioned that none of the existing solvers is able to check, if there is a solution, then how many different solutions are there for a generated Sudoku instance. By the way, there was no such existing technique that can effectively endorse the validity of a created Sudoku instance. Here we observed the beauty and novelty of our devised algorithms that have taken role to judge the validity of a newly generated instance. In fact, we have been introduced a scheme in generating Sudoku instances where we are successful to certify all created instances and categorize them. Our Sudoku instance generator is a global one, which is rather a scheme comprising a collection of algorithms, that takes the help of a combination of two or more (up to

six) techniques in different ways such that instances could be evolved in a randomized way. In addition, our unbeaten Sudoku instance generator is simple as well, which is able in creating thousands of Sudoku instances.

In chapter 6, we have explored some extremely important spheres of application of the Sudoku puzzle. We have proposed two new algorithms, which can be straightaway applied in the domains of *Steganography* and *Biometrics*.

7.3 Future Scope of the Work

Now we like to point out a few probable extensions of our work where the researchers interested to explore in this field may get some clue; some of these are as follows:

(i) In this thesis, we have developed a guessed free Sudoku solver that contains four versions one after another elaborated in two chapters of this thesis. In future, interested researchers may explore other possible versions of the same or may design a new guessed free solver.

(ii) The algorithms devised herein or would be designed in future could be executed for other Sudoku instances larger than that of size 9×9. That means execution of the same algorithm for Sudoku instances like 16×16, or 25×25, or 36×36, or more where minigrids are also square in dimension, or instances of other rectangular shapes could be a part of future research.

(iii) Larger Sudoku instances can also be generated using the algorithmic notions accumulated and devised in this thesis. Also an average variation in computation time for different Sudoku sizes could be interesting to visualize for a fixed but large number of Sudoku instances that are to be generated in a similar fashion.

(iv) Future researchers may also consider Sudoku instances in the third dimension that are termed as *cubic* Sudoku [13]. Furthermore, cubic Sudoku can also be expanded in modular fashion in the third dimension, if computations are somehow bounded within a reasonable amount of CPU time.

(v) In this thesis we have talked about some important fields of application where solving of Sudoku instances is a must as an intrinsic part of the same. Besides, we have explored two new applications in the fields of Steganography and Biometric. We strongly feel that there are scopes in discovering inventive domains of application, either in real life or in some research problem,

where solving Sudoku puzzles has its inseparable role. In this respect, the larger Sudoku instances that we talked about above, either in 2D or in 3D, might require to generate and solve as a part of future research, where smaller solved Sudoku puzzles might not be placed aside to obtain a larger space. In such a case, repetition of the same smaller puzzles with biased nature could be avoided.