# Chapter 1

# Introduction to Sudoku Problem

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 $n \times n$, where $n$ is such an integer so that $\sqrt{n}$ is also an integer. This is how the concept of minigrid is formed, which are of size $\sqrt{n} \times \sqrt{n}$ each, wherein each of the numbers ranging 1 through $n$ would occur exactly once. The most common size of such a (square) grid is 9×9 that usually we view in newspapers or magazines. Besides the standard 9×9 grid, variants of Sudoku puzzles include the following:

- 4×4 grid with 2×2 minigrids,
- 16×16 grid (super Sudoku) [1],
- 25×25 grid (Sudoku, the Giant) [2],
- A 3D Sudoku puzzle [3] (being invented by Dion Church was published in *The Daily Telegraph* in UK in May 2005),
- Alphabetical variations that use letters rather than numbers. *The Guardian* (in UK) calls these Godoku [4] while others refer to them as Wordoku [5], etc.

One such problem instance of 9×9 Sudoku Puzzle is shown in Figure 1.1(a) and its solution is shown in Figure 1.1(b). In these figures, we may find the square minigrids of size 3×3 each enclosed by firm lines.

A complete Sudoku solution may be arrived at in more than one ways, as we can start from any of the given clues that are distributed over the minigrids of a given Sudoku instance (to be solved). There is no known technique in the literature that we surveyed to determine how many different starting cells there are. Moreover, a Sudoku starting cell is really only interesting to a mathematician if it follows a minimal route. Removal of a single clue may generate another Sudoku instance for which other solutions may also exist and in general, the solution is no longer unique.

There is also no known technique in the literature to figure out the number of possible minimal cells 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 [6], 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 even if the number of clues is more than 17; as for example, see the Sudoku instance shown in Figure 1.2 with two of its possible valid solutions, where the number of clues given is 32.

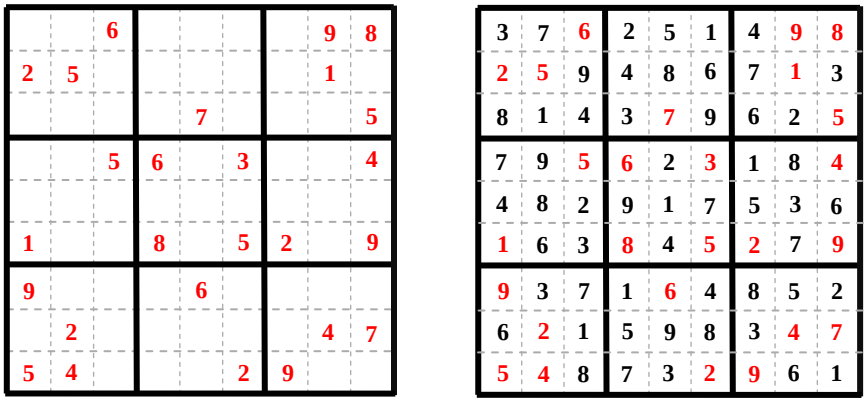
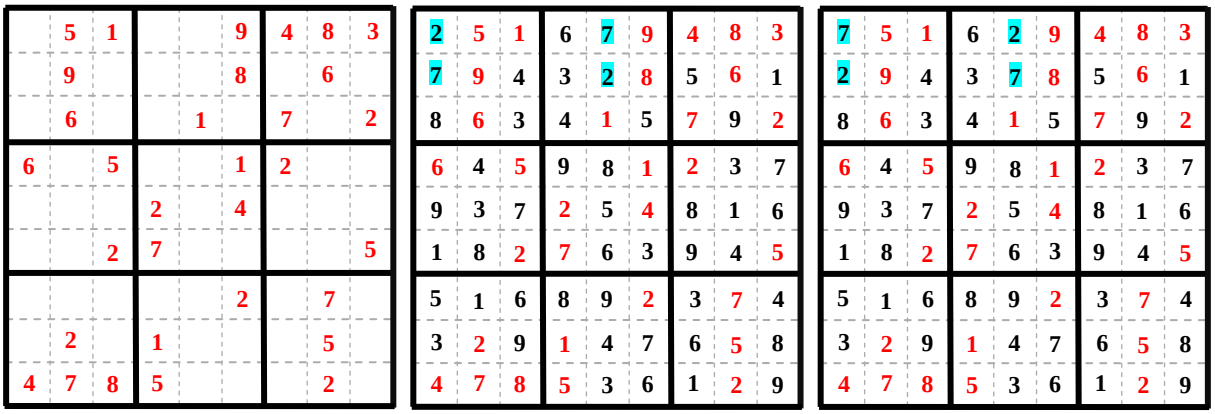



**Figure 1.1: (a)** An instance of the Sudoku problem. **(b)** A solution of the Sudoku instance shown in Figure 1.1(a), where a digit / symbol occurs exactly once in each row, column, and minigrid.

Among the researchers there is a debate that, if a Sudoku instance has more than one solution, then whether this could be considered as a valid Sudoku instance. We can say that, as per the definition of Sudoku, every puzzle must have unique set of numbers in each row, column, and minigrid. There is nothing in the rule that states that every puzzle must have a unique solution [7]. Rather, these are the puzzle creators, who put an additional constraint that tells that a Sudoku instance is valid only if it produces a unique / single solution [8]. But nobody has effectively explored whether a 'well-built' Sudoku instance (of more than 17 clues) has really one and only one solution [9]. Some mathematicians also debated that Sudoku instances can all be solved by certain standard tricks, many of which may result a unique *rational* solution to the integer programming problem [9]. We can find rational solutions with linear programming, and if the rational solution is unique, that type of integer programming problem is not NP-hard; it is in P [9]. However, Sudoku is known to be an NP-complete problem [10]. So, it is not well-off to say

that, every Sudoku instance must have a unique solution, or otherwise, the Sudoku instance is not valid.

We have found several Sudoku instances with more than one solution in literature [9, 11, 12]. So, we can say that, a subset of Sudoku instances may have one and only one valid solution, but in general, a Sudoku instance might have two or more solutions as well. This situation has its own merits and applications. We have already mentioned that solving Sudoku puzzles has an application in Steganographic domain. If a given Sudoku instance is having only one valid solution, then that instance may fail to conceal some information while transmitting the instance that an intruder may attack. On the other hand, if there are two or more solutions for a given Sudoku puzzle, then such attackers may confuse in extracting the hidden information. In any case, there are several Sudoku solvers available in literature tried to solve a Sudoku instance, and based on their capability and solving criteria Sudoku instances are also classified into different levels of difficulty.



**(a)** **(b)**

**Figure 1.2: (a)** A Sudoku puzzle P. **(b)** Two solutions of the Sudoku puzzle P, in which the valid permutations that produce the solutions differ only in the first and second minigrid.

## 1.1 A Brief Background on Sudoku

The name Sudoku or more correctly 数独 comes from Japan and consists of the Japanese characters Su (meaning ‘number’) and Doku (meaning ‘single’) but it was not invented in Japan. Sudoku was originated in Switzerland and then travelled to Japan through America [13].

The origin of *Sudoku* can be tracked down back to 1782 with the Swiss mathematician Leonhard Euler [14]. He introduced a famous problem on a grid: the *officer problem*. Six regiments with

six officers with distinct ranks are considered. The problem consists of placing the 36 officers into a 6×6 square. Each cell of the square contains one officer. In each line and in each column, every regiment and every rank is represented. Euler suspected that it was an unsolvable problem but could not prove it. This result was only proved in 1901 by the French mathematician Gaston Tarry. The constraint preventing the repetition of some element in the grid is the link with Sudoku. In both cases, we are faced with a particular Latin square.

### 1.1.1 Latin Square

Another interesting historical number square is that of the Latin square [14], given its name by Leonhard Euler. A Latin square is square matrix $n \times n$ filled with distinct elements and whose lines (or rows) and columns contain only one instance. Figure 1.3 shows one Latin square with $n = 4$, where the numbers 0 through 3 appear only once in each row and column of the puzzle. Sudoku grids are actually a subset of Latin squares with the additional constraint of uniqueness within minigrids imposed. Because of this property, some of the enumerations and proofs for Latin squares can be readily adapted to hold for Sudoku grids.

| | | | |
|---|---|---|---|
| 0 | 1 | 2 | 3 |
| 1 | 2 | 3 | 0 |
| 2 | 3 | 0 | 1 |
| 3 | 0 | 1 | 2 |

**Figure 1.3:** A Latin square where the numbers 0 through 3 are present in each row and column uniquely.

### 1.1.2 Magic Square

The Magic square [14] is first documented in China two thousand years ago. The puzzle is both a numerical and positional problem, as all the rows, columns, and diagonal lines through the grid must add up to the same number. Just as in Sudoku a number can only be used once in the grid. The aim of the puzzle is to try to devise a new ordering of the numbers to complete the puzzle starting from scratch. A sample 3×3 Magic square is shown in Figure 1.4. In Figure 1.4, we can

easily observe that, if we add all the numbers placed in each row, column, and diagonal, we can get the same number, i.e. 15.

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

**Figure 1.4:** A Magic square where the numbers 1 through 9 are present exactly once in a 3×3 grid such that the sum of the digits in each row, column, and diagonal (or crossway of length three) equals to 15.

### 1.1.3 Modern Sudoku

At the end of the 19th century, ancestors of Sudoku could be found in some French newspapers. In 1892, in the daily newspaper *Le Siècle* [14] an 81-cell grid was given (with identified blocks) but with a 2-digit number instead of our classical digits (from 1 to 9). Figure 1.5 reproduces such a grid.

UN PROBLÈME PAR JOUR

5129. — CARRÉ MAGIQUE DE 9 A COMPARTIMENTS ÉGAUX

Composé par M. le commandant Cocoz

Compléter le carré ci-dessous en employant les 81 premiers nombres de manière que les neuf carrés composés de neuf nombres donnent chacun, comme total, 369. De plus, chacune des deux grandes diagonales, chaque horizontale et chaque verticale du carré complet devra donner 369.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 17 | 20 | 5 | 55 | 67 | 79 | 54 | 36 | 39 |
| 70 | 73 | | | 42 | | | 8 | 11 |
| 45 | | 33 | 2 | | 26 | | | 64 |
| 19 | | 16 | 11 | | 20 | 63 | | 78 |
| 47 | 32 | | | 81 | | | 10 | 22 |
| 75 | | 72 | 13 | | 1 | 35 | | 50 |
| 60 | | | 27 | | 15 | 37 | | 34 |
| 6 | 18 | | | 28 | | | 77 | 62 |
| 31 | 43 | 46 | 80 | 56 | 68 | 12 | 24 | 9 |

**Figure 1.5:** A pre-Sudoku in the newspaper *Le Siècle* published on November 19, 1892 [14].

In 1895, in another newspaper *La France* (see Figure 1.6), a game using digits from 1 to 9 (but with no identified blocks) was given. Those weekly published games appeared in other titles in the French media but did not manage to survive. Modern Sudoku was created by Howard Garns, a retired architect and a mind puzzles fan. He introduced a new dimension in the classical Latin

square and presented it as a partially filled grid that was to be completed. The first grid was published in 1979 in the *Dell Pencil Puzzles and Word Games* magazine under the name *Number Place*. In Japan, Sudoku is called number place. In Japanese, this is pronounced "*nambaapureesu*", which is often abbreviated to *nampure*.

DIVERTISSEMENTS QUOTIDIENS

N° 3879 — CARRÉ MAGIQUE DIABOLIQUE

Par M. B. Meyniel

Compléter le carré ci-dessous en employant les neuf premiers nombres chacun neuf fois de manière que les horizontales, les verticales et les deux grandes diagonales donnent toujours à l'addition le même total.

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

**Figure 1.6:** A pre-Sudoku published in *La France* on July 6, 1895 [14].

In April 1984, Nikoli, a Japanese editor, introduced the game in Japan under the name *Sudoku*. Later on, and this is probably why the game became so popular, the number of givens was limited to 32 and published grids were made symmetrical; the givens (or clues) were evenly distributed around the centre of the grid. Today, most prominent Japanese newspapers publish a Sudoku grid daily. In 1989, the software company LOADSTAR published the *DigiHunt* game for the Commodore 64. This was probably the first piece of software that was able to produce

Sudoku grids. A company still uses this name. In 2005, the *New York Post*, *USA Today*, and the *San Francisco Chronicle* published their grid. In July 2006, the game arrived in France.

### 1.1.4 Sudoku and Medias

The legends say that the Sudoku phenomenon was initiated by Wayne Gould, a retired judge from New Zealand [14]. In a Japanese bookstore, a partially filled grid caught his eye. For more than five years, he spent his time writing computer software to generate such a grid (he had to learn programming from scratch). He promoted his game in *The Times*. The first grid was published on November 12, 2004. A few days later, the *Daily Mail* published its own grid under the name *Codenumber*. Since then the *Daily Telegraph* published its grid in early 2005. At the end of 2005, Wayne Gould was the provider for around 70 daily newspapers all around the world. The British media realized quite early on that having a Sudoku grid in a newspaper was a good thing for sales. Most national newspapers therefore started a regular publication of those grids. Indeed, Sudoku is now in *The Independent*, *The Guardian*, *The Sun* (*Sun Doku*), and *The Daily Mirror*.

Sudoku's popularity is often explained by the fact that people love watching the grid filling itself. For *The Times*, both easy and hard grids are appreciated. Consequently, since June 20, 2005, the two grids have been published side by side. As early as July 2005, Sudoku went to TV on British channels. The first Sudoku show was aired on *Sky One*. Nine teams of nine players (including a celebrity) representing regions in Britain tried to complete a Sudoku grid. Each player had a device that allowed them to fill a digit into one of the four cells they were responsible for collaboration among the players. The audience at home could play an interactive challenge. This popularity burst made people call Sudoku the "Rubik's cube of the 21st century". The game arrived on Indian shores in June 2005 via some daily newspapers, including *The Hindu*, *Hindustan Times*, *Deccan Chronicle*, and *The Asian Age*.

## 1.2 Difficulty Level of the Sudoku 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 1.1 shows a comparison chart of the number of clues for different difficulty levels [13].

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 [15], as shown in Table 1.2.

**Table 1.1:** 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 (Hard) | 28-31 |
| 5 (Evil) | 17-27 |

**Table 1.2:** 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 (Hard) | 02 |
| 5 (Evil) | 00 |

## 1.3 Motivations behind the Present Work

There are several Sudoku solving techniques available in literature that try to solve a Sudoku instance based on the difficulty level. Each of these solvers at a time considers each blank cell of a Sudoku puzzle and tries to put a probable number in its location. If we briefly categorize the different Sudoku solving techniques, they are categorized as follows. (1) Elimination based solving techniques, each of which lists all the possible numbers for blank cells in the Sudoku puzzle. It maintains this list based on entirely guessing. It then tries to minimize the number of possible numbers (or digits) for a cell based on their different patterns and constraints. We have to consider each of the cells in the puzzle, i.e. 81 cells for 9×9 Sudoku puzzles and find out the different patterns on these cells, which is extremely time consuming. (2) On the other hand, all the soft computing based Sudoku solvers either use Genetic algorithm [16] or Bee colony [17],

which is exhaustive and extremely time consuming. Based on different levels of difficulty, the techniques adopted to devise algorithms also change. There are no systematic way-out for solving the puzzle game.

Often guessing technique does not guarantee a valid solution. They cannot be straightway applied to puzzles of higher sizes such as of size 16×16 or 25×25, etc.

Moreover, all of these techniques are not able to identify whether a Sudoku puzzle is valid or not, i.e. whether a given puzzle has really any valid solution.

All the existing Sudoku solvers can compute at most one solution of the puzzle, if one exists. If there are two or more valid solutions, they are incapable to find them out.

We have also revealed several spheres of application of this interesting puzzle game, such as Visual Cryptography [18], Digital watermarking [19], Steganography [20], Encrypting SMS [20], Image authentication [21], Image-Video Encryption [22], DNA computing [23], and so on and so forth. So, it is extremely important to improve the performance of the Sudoku solving techniques, which will indirectly improve the performance of the algorithm.

## 1.4 Objectives of the Present Work

In the current work, we have exhaustively studied each existing Sudoku solving technique that we have obtained along with their difficulty level. We have tried to identify and minimize the drawbacks of existing Sudoku solving techniques in our present work. The major objectives of the proposed Sudoku solvers are highlighted as follows:

(1) To develop a Sudoku solver that is entirely guessed free. We call this solver ‘entirely guessed free’ as not even for a single blank cell the algorithm guesses a (probable) digit that could be assigned to it. It follows some distinct deterministic steps to assign values to each blank location of the Sudoku instance. Whereas the other solvers, we have studied so far, first maintains a probable list of numbers, based on guessing for a blank location and then eliminate the redundant candidates to fix up a distinct value for the said location.

(2) All the different Sudoku solving techniques are applied to the puzzle based on different patterns of the probable list of numbers of a blank cell and difficulty levels of the puzzle. It often does not lead to deterministic steps for finding the solution. As there is no standard definition of

difficulty levels these methods are applied randomly on trial and error basis. Therefore, to develop a technique which will follow some deterministic steps to find out valid solution(s) of the puzzle without depending on various patterns of probable list of numbers and difficulty levels is another important objective of the present work.

(3) Instead of considering every cell of a given Sudoku puzzle, we only consider the minigrids to reach to valid solution(s), as the number of minigrids is much less than the number of cells in a Sudoku puzzle. For example, the number of minigrids of a 9×9 Sudoku puzzle is only 9, whereas the number of cells of a 9×9 Sudoku puzzle in 81. This way, it certainly enhances the performance of the Sudoku solver.

(4) To develop a solver that guarantees a valid solution, if it exists, as none of the existing solvers guarantees a solution.

(5) To develop a methodology, which can right away make a decision whether a given instance is a valid Sudoku instance, i.e. whether the given instance is at all having a solution.

(6) As each of the entire existing Sudoku solvers can compute only one solution of a given Sudoku puzzle, even if the instance might have two or more solutions in reality, thus it is interesting if all of them could be computed without any added cost, and it is also a new objective of the present work.

(7) To develop a Sudoku solver, which can straightway be applied for solving puzzles of larger sizes such as 16×16 or 25×25.

(8) To cultivate an entirely graph theoretic representation of the puzzle game, which is novel in the sense that nobody has ever tried and/or developed such a depiction earlier.

(9) Identifying some new spheres of application of solving a Sudoku puzzle.

(10) To have a brief study on different Sudoku instance generation algorithms.

## 1.5 Achievements Out of the Work Done

In a modest way, the following contributions have been made in this thesis work:

(1) We have studied exhaustively all the different existing Sudoku solving techniques.

(2) We have developed algorithms for solving the Sudoku puzzle in a guessed free manner based on minigrids instead of cells. The algorithm is having four different versions. In the first version (i.e. Version 3.1), minigrids are considered based on the number of given clues. A minigrid with more clues is considered first. In the second version (i.e. Version 3.2), minigrids are taken care of in a sequence, either following spiral, or zigzag, or semi-spiral run for their probable valid combinations such that eventually a desired solution of the Sudoku instance is obtained. In the third version (i.e. Version 4.1 in this thesis), we consider all individual valid solutions of all the minigrids in isolation (based on the given clues in other row and column minigrids) and with the help of a graph theoretic formulation among the individual valid solutions of the minigrids we generate all valid solutions of the given Sudoku instance; this number is either zero (means no solution), or one (means unique solution), or more (means all solutions), if they exist. In the fourth version (i.e. Version 4.2) of the algorithm, we have further compacted the graph structure. So far our study and exploration cover the topic, this is the first attempt in developing a Sudoku solver, which is novel in many ways such as this algorithmic technique is completely guessed free where minigrids are considered as a whole instead of individual blank locations (or cells), it computes all valid solutions of a given Sudoku instance, if there is no unique solution of the given instance, and so on and so forth.

(3) We have exhaustively studied different Sudoku instance generation algorithms based on the different difficulty levels of the puzzle.

(4) We have identified different application domains of the Sudoku puzzle. We also have developed two new algorithms for applying into two recent domains, i.e. Steganography and Biometric.

The proposed method is completely guessed free, as already has mentioned a number of times, which considers minigrids only (instead of (blank) cells) of size 3×3 each that has been developed for the first time in designing such an algorithm for a given Sudoku instance of size 9×9. In different versions of the algorithm, once the minigrids are considered one after another, and then all the minigrids are considered all at once (without imposing any priority on them).

A complete Sudoku solution may be arrived at in more than one way, as we can start from any of the given clues that are distributed over the minigrids of a given Sudoku instance (with blank cells). Nobody has yet succeeded in determining how many different starting cells there are.

Moreover, a Sudoku starting cell is really only interesting to a mathematician if it follows a minimal route. Removal of a single clue may generate another Sudoku instance for which other solutions may exist and the solution is not necessarily unique. In these respects, our algorithmic solution path is unique in determining a valid solution, as we are interested in computing a solution of a given Sudoku instance, if one exists. In addition, as we consider a Sudoku instance minigrid-wise, instead of cell-wise, this algorithm is quicker in computing a solution.

In our proposed algorithm a pre-processing of computing all valid permutations only is there for all the minigrids based on the clues in a given Sudoku instance. It has been observed in most practical situations that the number of valid permutations is significantly 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 other minigrids along the row and column of the said minigrid, particularly help in reducing the eventual number of valid permutations for that minigrid too. Anyway, 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 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 through streets); rather, it is one step above the ground of the puzzle by considering groups of cells or minigrids (and searching the same from a bird's eye view).

In spite of the controversy that a Sudoku puzzle must have only one unique solution, a given Sudoku instance may have two or more valid solutions that finds applications in different fields of research and real life scenario. The proposed Sudoku solver in this thesis is capable of computing all those solutions, if they exist. According to our expertise in the domain of work (i.e. Sudoku puzzle) and the application of graph as a tool to solve this problem, we have extracted a graph structure (that has been shown in Figure 4.1), which is to be present in the overall graph theoretic representation of the problem under consideration and the way we have

anticipated to solve the problem. In fact, the algorithm is capable of computing all those solutions, if there are two or more.

A Sudoku instance might have two or more solutions that have some inherent usefulness in one application where we like to hide some information. Particularly, let us assume a Sudoku instance is sent with some concealed information. An interloper may access the instance and may make one valid solution of this instance. By the way, this solution may not interpret the concealed information as it is kept in some other valid solution of the same instance.

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 proposed method does not incur any additional cost for that; all these solutions can be produced at the same time.

The brilliance of the proposed algorithm is that the same logic can also straight away be applied for larger Sudoku instances such as 16×16, 25×25, or of any other rectangular sizes. Similar logic of extracting an appropriate graph structure (some sort of the graph in Figure 4.6 for a 9×9 puzzle) based on a larger Sudoku instance under consideration is to be executed. It is easy to understand that this task for larger Sudoku instances is extremely difficult, tedious, and time-consuming that such algorithmic line of attack (as has been developed in this thesis) is able to solve in a reasonable amount of time.

The level of difficulty is another important issue that almost all the earlier Sudoku solvers consider while developing an algorithm. There are no hard and fast rules that state the difficulty level of a Sudoku puzzle. A sparsely filled Sudoku puzzle may be extremely easy to solve, whereas a densely filled Sudoku puzzle may in fact be more difficult to solve. From a programming viewpoint, we can determine the difficulty level of a Sudoku puzzle by analyzing how much effort must be expended to solve the puzzle, and the different levels of difficulty as Easy, Medium, Hard, and Evil. Incidentally, the Sudoku solver developed in this thesis is not depicted from the viewpoint of any level of difficulty; rather, all the Sudoku instances are as if belonging to the same difficulty level. In some cases, more valid permutations may be generated for some minigrid, but in general, on an average the number of valid permutations is much less, and the minigrids with smaller number of valid permutations in reality guide to compute all the desired solutions for a given Sudoku instance in due course.

## 1.6 Outline of the Thesis

The thesis consists of seven chapters. In Chapter 2, we have discussed related background of the problem such as nature of the problem is discussed. We have also made an extensive study on different existing Sudoku solving techniques along with their comparative advantages and inadequacies in this chapter.

In Chapter 3, we have developed a new minigrid based Sudoku solver. For this purpose, we have first numbered every minigrid based on their relative locations. Then, valid permutations amongst minigrids are created. Here we have studied several permutation generation algorithms and in the end a novel tree based permutation generation technique is assumed and implemented. The Sudoku solving algorithm has two versions namely Version 3.1 and Version 3.2, based on the selection of minigrids. In Version 3.1 of the proposed algorithm, we have considered the minigrids with more number of clues first, and then all the minigrids are considered one after another based on their relative position in the same column / row / minigrid and the digits (or symbols) of clues given. In Version 3.2 of the proposed algorithm, we have considered the minigrids in a predefined sequence such as zigzag, or spiral, or semi-spiral in row-wise / column-wise fashion.

In Chapter 4, we have developed an exclusively graph theoretic minigrid based approach for solving a Sudoku puzzle. This algorithm is also again having two versions, namely Version 4.1 and Version 4.2. In Version 4.1, we draw a graph derived from the interminigrid based valid permutations (among row and column minigrids), whereas Version 4.2 of the algorithm is a compacted version of the graph drawn in Version 4.1. All these versions are explained with suitable examples. A comparative study has also been made along with other different Sudoku solvers.

In Chapter 5, we have exhaustively discussed different Sudoku instance generation algorithms based on their various levels of difficulty.

In Chapter 6, we have explored several application domains of solving Sudoku instance. We have also discussed two newly developed algorithms in the domain of Steganography and biometric template encryption along with their necessary backgrounds and results.

The thesis is concluded in Chapter 7 with all necessary discussion on the present work. Furthermore, we draw attention to some plausible open problems as further research scopes.