Applied Mathematics, Modeling and Computer Simulation
C.-H. Chen et al. (Eds.)
© 2023 The Authors.
This article is published online with Open Access by IOS Press and distributed under the terms of the Creative Commons Attribution Non-Commercial License 4.0 (CC BY-NC 4.0).
doi:10.3233/ATDE231062

Sudoku Solver: A Comparative Study of Different Algorithms and Image Processing Techniques

Swati Hira, Neha Bhagwatkar[1], Kartik Agrawal, and Nikunj Loya
Shri Ramdeobaba College of Engineering and Management, Ramdeo Tekdi, Gittikhadan, India

Abstract. This research paper aims to explore the different types of algorithms used for solving Sudoku puzzles and provide a comparative analysis of their efficiency and accuracy. Sudoku puzzles have become increasingly popular in recent years, and many people enjoy the challenge of solving these puzzles. However, for more difficult puzzles, finding a solution can be time-consuming and require a lot of trial and error. In this paper, we will examine the Backtracking approach, Backtracking + Hashmap approach, Constraint Propagation Algorithm, and Dancing Links Algorithm for solving Sudoku puzzles. We will also provide a detailed analysis of the advantages and disadvantages of each algorithm, based on their performance on different levels of difficulty of Sudoku puzzles. We will also compare the efficiency and accuracy of these algorithms to determine which approach is best suited for solving Sudoku puzzles. The proposed approach will be of interest to puzzle enthusiasts, researchers in the field of artificial intelligence, and anyone interested in exploring different approaches to solving complex problems using algorithms.

Keywords. Sudoku puzzle; image processing; backtracking; constraint propagation.

1. Introduction

SUDOKU is a popular logic-based puzzle game that has gained immense popularity in recent years. The game involves filling a 9 × 9 grid with digits from 1 to 9, in such a way that each digit appears only once in each row, column, and 3 × 3 sub-grid. The puzzle is usually presented with some cells pre-filled, and the challenge is to fill in the remaining cells while adhering to the Sudoku constraints. While Sudoku puzzles are simple to understand, solving them can be quite challenging, even for experienced players. In recent years, researchers and puzzle enthusiasts have developed various algorithms to solve Sudoku puzzles efficiently. These algorithms employ different techniques, including backtracking, constraint propagation, and dancing links, among others, to find the solution to a given puzzle. However, there is no consensus on the most efficient algorithm to solve Sudoku puzzles. In this research paper, we will explore the approach where a

[1] Correspondence author, Neha Bhagwatkar, *Shri Ramdeobaba College of Engineering and Management*; Email: bhagwatkarns@rknec.edu

Sudoku solver takes input as an image, converts it into a 2D array, and then provides a solution to the Sudoku puzzle using four different algorithms. These algorithms are backtracking, backtracking with a hashmap, constraint propagation, and dancing links. In this research paper, we will implement all four of these algorithms and evaluate their performance in solving Sudoku puzzles. We will also conduct a comparative analysis of these algorithms to determine their efficiency in solving Sudoku puzzles. Our experiments will be conducted using Python, and we will use various performance metrics to compare the efficiency of the four algorithms. We will also analyze the strengths and weaknesses of each algorithm and provide insights into the type of Sudoku puzzles each algorithm is best suited for. In conclusion, this research paper provides a comparative analysis of four different algorithms used to solve Sudoku puzzles. Our analysis provides insights into the strengths and weaknesses of each algorithm and provides guidance on which algorithm to use for different types of Sudoku puzzles. Overall, this research paper contributes to the growing body of literature on Sudoku puzzle solving and provides a valuable resource for researchers and puzzle enthusiasts alike.

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

Figure 1. An incomplete sudoku.

2. Literature Review

2.1. Research Papers on Sudoku Solvers

A lot of research has been done in this field. Following are some of the research papers which will be discussed for a better understanding of the importance and the algorithms used to solve sudoku puzzles.

A paper by Maji A K and Pal R K, 2014 [1] provides an overview of the traditional backtracking algorithm used to solve Sudoku puzzles. The authors then propose a modified version of the backtracking algorithm that uses a mini grid- based approach to keep track of the values already present in the mini grids, rows, and columns of the puzzle. This approach aims to reduce the number of backtracking steps required to solve the puzzle by limiting the possibilities of values that can be placed in a particular cell. The proposed algorithm also incorporates a heuristic approach to prioritize the selection of cells that are likely to lead to a faster solution. The heuristic function evaluates the number of possible values that can be placed in a particular cell

and selects the cell with the fewest possibilities. This approach aims to reduce the search space of the algorithm and speed up the process of finding a solution. The results show that the mini grid-based backtracking algorithm outperforms the traditional backtracking algorithm in terms of the number of steps required to solve the puzzles.

Wang C et al. [2] proposes a novel evolutionary algorithm for solving Sudoku puzzles that utilize both global search and local search strategies. The authors describe their proposed algorithm, which uses a genetic algorithm to generate an initial population of candidate solutions and then applies a local search approach to improve the quality of the solutions. The local search approach used in the proposed algorithm operates on both columns and sub-blocks of the puzzle. It involves swapping the values in the same column or sub-block to improve the quality of the solution. The authors also propose a modified version of the local search approach that incorporates a simulated annealing strategy to avoid getting stuck in local optima.

The proposed algorithm in "A Novel Automated Solver for Sudoku Images" by Gupta K, Khatri S and Khan M H [3], uses image processing techniques and computer vision algorithms to extract the puzzle grid from an input image and then applies a combination of rule-based and optimization- based approaches to solve the puzzle. It consists of several stages such as image pre-processing, grid extraction, digit recognition, and puzzle solving. The image pre-processing stage involves applying various filters and transformations to enhance the image quality and improve the accuracy of the subsequent stages. The grid extraction stage involves using edge detection and line segmentation algorithms to identify the boundaries of the puzzle grid and extract it from the input image. The digit recognition stage involves using a convolutional neural network (CNN) to recognize the digits in each cell of the grid. The puzzle-solving stage uses a combination of rule-based and optimization-based approaches to solve the puzzle. The rule-based approach involves applying the basic Sudoku rules to fill in the cells that have a unique solution. The optimization-based approach involves using a genetic algorithm to search for the optimal solution by iteratively generating and evaluating candidate solutions.

The proposed algorithm in "A hybrid backtracking and pencil and paper sudoku solver" by Nwulu, Evarista and Bisandu, 2019 [4] aims to improve the efficiency of the backtracking algorithm by using human-like reasoning and deduction to reduce the number of backtracking steps re- quired to solve the puzzle. The puzzle representation stage involves representing the puzzle in a format that can be easily manipulated by the algorithm. The human-like reasoning stage involves using logic and deduction to fill in cells that can be uniquely determined based on the rules of Sudoku. This stage is inspired by the traditional pencil-and- paper method used by humans to solve Sudoku puzzles. The backtracking stage involves applying a backtracking algorithm to fill in cells that cannot be uniquely determined using the human-like reasoning stage. However, unlike traditional backtracking algorithms, the proposed algorithm uses information gathered from the human-like reasoning stage to guide the backtracking process and reduce the number of backtracking steps required. The solution validation stage involves checking the solution obtained from the algorithm to ensure that it satisfies the rules of Sudoku. The authors also describe several optimizations and heuristics that can be used to further improve the efficiency of the algorithm.

The research by Gummadi Sai Dheeraj, Lakshmi K B V, Anjan Krishna K. 2020 [5] includes a system that builds up a program to identify the Sudoku region, tackle it

and print the answer with expanded reality on a screen. They move just the areas of numbers within the grid and distinguish if encompass a number or are unfilled, if variety is present no action is completed but if the matrix position is vacant, consequently zero is put. The flow consists of steps like loading the input image, localizing the sudoku puzzle in the image, localizing each cell in the sudoku board location, determining if the digit exists, and if yes OCR it, given cell location and digit solves the sudoku using backpropagation and CNN and display the image. The solution depicted overwrites the input image.

Using computer vision and deep learning, the authors, Syed, Azhar and Merugu, Suresh and Kumar, Vijaya (2020) [6] suggest a superior method for identifying Sudoku puzzles and using constraint programming and backtracking to solve them using an algorithm to present the finished puzzle as augmented reality. Along with this study, a performance comparison with earlier work is offered. Methods: Image categorization alone won't be enough to apply augmented reality to the Sudoku puzzle since the solved puzzle must be displayed on top of the area of the unsolved puzzle in the original image. In order to accomplish puzzle detection, they used the CNN and Object Localization algorithms. After the detection, they saved the detected values in each 9 × 9 cell, solved the puzzle using a constraint programming and backtracking technique, and then filled the discovered empty cells with the correct puzzle solution.

'Sudoku Solving Using Patterns and Graph Theory', 2021 [7] presents a novel approach to solving Sudoku puzzles using pattern recognition and graph theory techniques. The authors describe their proposed approach, which involves using pattern recognition techniques to identify recurring patterns in the Sudoku grid. By directing investigations and arranging the perceptions, an awesome strategy for incredibly quick Sudoku solving using recognition of various examples like Naked Singles, Hidden Singles, Locked Candidates, and so forth is examined. Evaluation of the method for solving random arrangements of sudoku puzzles reveals that the rate of resolution can be greatly accelerated. Diagram shading is also the greatest way to understand the simple method to solve the Sudoku, thus we consider this relationship when setting up the Sudoku chart. By conducting experiments and plotting the results, a revolutionary method for solving Sudoku very quickly that makes use of the recognition of different patterns, such as Naked Singles, Hidden Singles, Locked Candidates, etc., is reviewed by Rohit Iyer, Amrish Jhaveri, Krutika Parab in their research "A Review of Sudoku Solving using Patterns", 2013 [8]. The rate of solving can be significantly increased, according to an analysis of the method for solving a random set of Sudoku puzzles. However, just a few patterns have been identified and solved, and additional patterns may be able to be found and solved to increase the rate at which Sudoku can be solved. The performance of solvers was calculated and tested on a set of 1000 puzzles.

In the paper by Musliu N and Winter F, 2017 [9], a novel local search method based on the Sudoku min-conflicts heuristic is presented. Additionally, within the context of iterated local search, they have presented a novel hybrid search method that makes use of constraint programming as a perturbation tool. They have empirically tested the approaches against difficult Sudoku benchmarks and have reported advantages over cutting-edge answers. They also used their strategy on another difficult scheduling problem to demonstrate the generalizability of the suggested methodology. The outcomes demonstrated that the suggested approach is reliable in a different issue domain.

In the case of quantum computing, the problem can be solved in miraculous O(log(n)) time complexity. To answer the sudoku, quantum computing techniques like superposition and entanglement were employed by Varun Singh, Varun Sharma, 2023 [10]. Quantum computers have recently demonstrated promise as a cutting-edge technology for re- solving challenging issues in a variety of industries, including optimization and cryptography. In this paper, they looked into the possibility of using quantum computers to solve Sudoku puzzles. They outlined a quantum method for solving Sudoku problems and assess its effectiveness in comparison to traditional techniques. The findings demonstrate that the quantum algorithm surpasses traditional methods in terms of both speed and accuracy, offering a new method for successfully completing Sudoku problems.

3. Methodology

3.1. Algorithms Used

In our proposed approach, four types of algorithms used—(1) Backtracking; (2) Backtracking + Hashmap; (3) Constraint Propagation Algorithm; (4) Dancing Links/Algorithm X.

(1) Backtracking: To solve a Sudoku puzzle, the back- tracking method checks each empty cell and tests each possible candidate or compatible digit. If a candidate conforms to the rules of a Sudoku grid with no violations, it is placed in the cell. The next empty cell is considered, and if a suitable candidate can be found that also conforms to the rules of a Sudoku grid, it is placed in that cell. However, if no candidate can be found, the backtracking method returns to the previous cells and replaces the candidate to conform to the rules of Sudoku. While the backtracking method guarantees a solution for all valid Sudoku puzzles, it is relatively more time-consuming compared to other methods [11].

(2) Backtracking + Hashmap: Optimization to the Back- tracking approach is by using a Hashmap to make searching efficient first iterate through the array then for each value in the array insert it into the hashmap. When using the backtracking + Hashmap optimization technique for solving Sudoku puzzles, if the algorithm attempts to insert a sequence of strings that is already present in the Hashmap, it indicates an inconsistent state. In such cases, the algorithm backtracks to the most recent consistent state and tries a different pathto continue exploring the search space [12].

(3) Constraint Propagation Algorithm: Constraint Propagation is a technique that involves applying the constraints of the problem to eliminate candidate values from variables. In Sudoku, this means using the rules of the puzzle to propagate constraints and eliminate candidate values. For example, if a cell has a value of 5, all other cells in the same row, column, and 3 × 3 box cannot have a value of 5. By repeatedly applying these constraints and eliminating candidate values, constraint propagation can reduce the search space and make it easier to find the solution to the puzzle. This technique can be applied in combination with other techniques, such as backtracking, to efficiently solve Sudoku puzzles [13].

(4) Dancing Links/Algorithm X: Dancing Links is a popular algorithm used for solving exact cover problems, including Sudoku. It was introduced by Donald Knuth in his paper “Dancing Links” in 2000 [14]. The algorithm uses a data structure called a doubly linked circular list to represent the matrix of the exact cover problem. Each row

and column of the matrix is represented by a node in the list, and each cell that has a 1 in the matrix is represented by a node in both the corresponding row and column lists. The algorithm starts by selecting a column with the fewest 1s in it and covering it by removing it from the header node. Then, for each row that has a 1 in that column, the algorithm recursively covers all other columns that have a 1 in that row. This process continues until either all columns have been covered (meaning a solution has been found), or there are no more rows to cover (meaning no solution exists). If the algorithm reaches a dead end, it backtracks to the previous decision point and tries a different path. This is done by "uncovering" the columns and rows that were previously covered and continuing the algorithm with the next possible column choice. Dancing Links is a powerful algorithm for solving exact cover problems because it is very efficient and has the ability to quickly find all solutions to a given problem. According to a research paper by Harrysson, M., and Laestander in 2014 [15], the Algorithm X implementation based on Dancing Links outperforms other exact cover problem solvers in terms of speed and memory usage on a variety of benchmark instances. The algorithm has been successfully applied to many problems beyond Sudoku, including scheduling, tiling, and more.

3.2. Method and Algorithm

Algorithm design is a crucial component of any computational problem-solving process. It involves the identification and specification of steps that must be taken to solve a problem, including the development of efficient and effective algorithms. In the case of a Sudoku solver using image processing, the goal is to design an algorithm that can accurately and quickly identify the numbers in the Sudoku puzzle from an image, and then solve the puzzle using established Sudoku-solving techniques. This section will outline the design of the algorithm used in the Sudoku solver, highlighting key steps and strategies employed to ensureaccurate and efficient performance.

1. Preprocessing:

a. Read the input image.

b. Convert the image to grayscale.

c. Apply Gaussian blur to the image.

d. Apply adaptive thresholding to the image to obtain abinary image.

e. Find the largest contour in the image and crop it to obtainthe Sudoku grid.

2. Grid Detection:

a. Divide the cropped grid into 81 cells.

b. Determine whether each cell is empty or contains anumber.

c. If a cell contains a number, use OCR (Optical Character Recognition) to detect the number in the cell.

3. Solving the Sudoku:

a. Represent the Sudoku as a 9×9 grid using a 2D array.

b. Select the algorithm to solve the Sudoku.

c. Display the solved Sudoku on the output image. (as shown in figure 4)

Output:

a. Display the input image with the solved Sudoku gridoverlaid on top of it.

b. Save the output image.

Figures 2 and 3 show the Use case diagram and the Landing page UI of the system respectively.

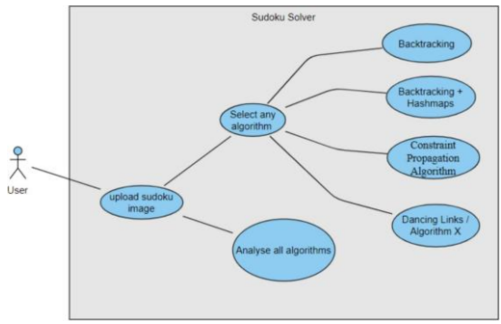

Figure 2. Sudoku solver use case diagram.

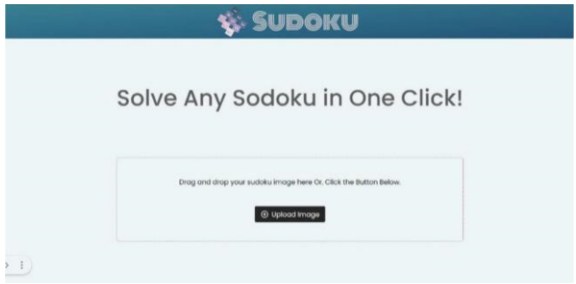

Figure 3. Landing page UI.

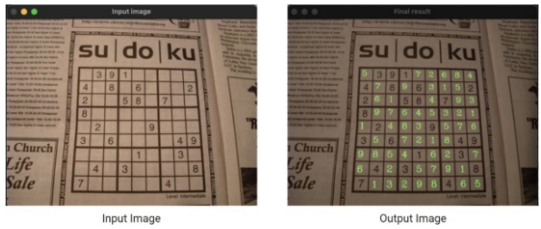

Figure 4. Input and output images.

3.3. Dataset Description and Analysis

The comparison between different algorithms is a crucial aspect of any research paper on the Sudoku solver. In this section, we present the comparison of four different algorithms for solving the Sudoku puzzle. The algorithms we have considered are backtracking, backtracking with hashing, constraint propagation, and dancing links. We tested these algorithms on a sample of 45 Sudoku puzzles with 15 very easy, 15 easy, 15 medium, and 15 hard puzzles. The puzzles were selected based on the number of pre-filled cells, with very easy puzzles having the most cells filled and hard puzzles having the least. The purpose of this comparison is to analyze the performance of each algorithm in terms of time and space complexity and to determine which algorithm provides the best results. The results of this comparison will provide valuable insights into the strengths and weaknesses of each algorithm, which can help in selecting the most appropriate algorithm for solving the Sudoku puzzle in different scenarios.

According to the table, the Dancing Links/Algorithm X algorithm consistently

outperforms the other algorithms in terms of speed, taking only 1.8 seconds to solve a Very Easy puzzle and 0.85 seconds to solve a Hard puzzle. The Back- tracking algorithm is the slowest, taking over 200 seconds to solve a Hard puzzle. The Backtracking + Hashmap and Constraint Propagation Algorithm fall somewhere in between, with the former performing better on easier puzzles and the latter performing better on harder puzzles. It's important to note that the results in the table may vary depending on the specific implementation of each algorithm and the hardware used to run the tests. Additionally, there may be other factors to consider besides speed, such as memory usage or ease of implementation.

3.4. Memory Analysis

In addition to analyzing the speed of algorithms in solving Sudoku puzzles of varying difficulty levels, memory usage is also an important factor to consider. To conduct the memory analysis, the Python memory profiler was used to measure the memory usage of each algorithm during the solving process. The results showed that backtracking used 0 MiB, while Dancing Links used the most memory at 0.5 MiB. Constraint Propagation Algorithm used 0.1 MiB, and Backtracking + Hashmap used 0.2 MiB. These findings provide insight into the memory efficiency of each algorithm and can aid in choosing the most appropriate algorithm for solving Sudoku puzzles.

4. Result and Analysis

4.1. System Setup

The system was set up to accept input in the form of JPEG files containing screenshots of Sudoku puzzles. The solver was able to successfully solve a large number of Sudoku problems using this input format. The system was tested on two different platforms: An Intel(R) Core(TM) i5-8300H CPU running Windows 11 Home (22H2), and an Apple M1 running macOS 13 Ventura. The solver was implemented using Python 3.9 and the following libraries were used: 1. OpenCV version 4.5.3 (for image processing) 2. Numpy version 1.21.2 (for numerical operations) 3. Matplotlib version 3.4.3 (for visualization) 4. React version 17.0.2 (for GUI) 5. Memory profiler version 0.58.0 (for memory analysis).

4.2. Findings Using Various Inputs

The solver was tested with sudoku problems of varying difficulty levels, and the results were compared using different algorithms: Backtracking, Backtracking + Hashmap.

Table 1. Comparison between algorithms.

Difficulty level	Backtracking	Backtracking + Hashmap	Constraint Propagation Algorithm	Dancing Links/Algorithm X (s)
Very Easy	0.009058	0.044428	0.239052	1.282325
Easy	0.006946	0.053653	0.119882	0.855890
Medium	0.018885	0.026188	0.386073	0.056195
Hard	215.0795	7.147690	2.788000	0.858287
Total	215.113389	7.271969	3.533056	2.052697

Constraint Propagation, and Dancing Links. The Dancing Links algorithm outperformed the other algorithms in terms of speed and the number of solutions found.

4.3. Few Points on Memory Analysis

Memory analysis was performed using the Python memory profiler. The results showed that the Dancing Links algorithm used the most memory (0.5 MiB) compared to the other algorithms. Backtracking used the least amount of memory (0 MiB), while Backtracking + Hashmap and Constraint Propagation used 0.2 MiB and 0.1 MiB, respectively. Dancing Links/Algorithm X used the most memory, but was still relatively efficient compared to other algorithms

4.4. Challenges

One of the main challenges faced during the implementation was the image processing phase, where the solver had to identify the grid and extract the digits. The input image, as shown in figure 5, if not a computer-generated image contained some noise which led the model to misread the grid. So this led to the need of preprocessing the image before passing it through the scanner. This required careful tuning of the parameters and handling various edge cases. Another challenge was to implement the different algorithmsefficiently and handle backtracking properly to avoid infiniteloops.

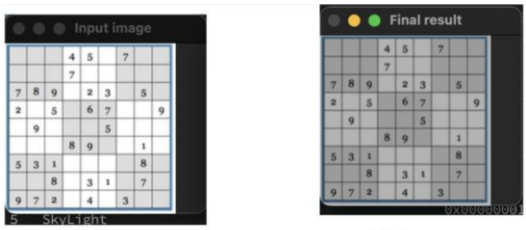

Figure 5. Input and Output for blurred Image.

5. Conclusion

Based on the research and analysis presented in this paper, we can conclude that the designed Sudoku solver using four different algorithms: backtracking, backtracking with a hashmap, constraint propagation algorithm, and dancing link algorithm, is an effective tool for solving Sudoku puzzles. According to the literature review, the backtracking and constraint propagation algorithms are generally the most popular algorithms for solving Sudoku puzzles, as they are efficient and can solve most puzzles quickly. The comparative analysis of these algorithms demonstrates that each algorithm has its strengths and weaknesses in terms of time and space metrics when dealing with different levels of sudoku puzzles. According to the analysis of the dataset, the dancing link algorithm outperforms all the other 3 algorithms in terms of time at the hard level of sudoku puzzles, which are most commonly available. In summary, the designed Sudoku solver provides an efficient and accurate solution to Sudoku puzzles,

and the comparative analysis of the algorithms used can serve as a valuable resource for researchers and practitioners working in this area. Further research can be conducted to explore the potential for combining these algorithms with other approaches to develop more powerful and versatileSudoku solvers.

References

[1] Maji A K, Pal R K. Sudoku solver using mini grid based backtracking [C]. 2014 IEEE International Advance Computing Conference (IACC), Gurgaon, India, 2014, pp. 36-44, doi: 10.1109/IAdCC.2014.6779291.

[2] Wang, C. et al. A Novel Evolutionary Algorithm with Column and Sub-Block Local Search for Sudoku Puzzles [J]. IEEE Transactions on Games, 2023, 1–11, doi: 10.1109/TG.2023.3236490.

[3] Gupta K, Khatri S, Khan M H. A Novel Automated Solver for Sudoku Images [C]. 2019 IEEE 5th International Conference for Convergence in Technology (I2CT), Bombay, India, 2019, pp. 1-6, doi: 10.1109/I2CT45611.2019.9033860.

[4] Nwulu E, Bisandu D, Dunka B. A hybrid backtracking and pencil and paper sudoku solver [J]. International Journal of Computer Applications, 2019, 181, 975-8887. 10.5120/ijca2019918642.

[5] Dheeraj G S, Lakshmi K B V, Krishna K A. Computer Vision based Sudoku Solving with Augmented Reality [J]. International Research Journal of Engineering and Technology (IRJET), 2020, 7(10): 1489–1493.

[6] Syed A, Merugu S, Kumar V. Augmented Reality on Sudoku Puzzle Using Computer Vision and Deep Learning [J]. Augmented Reality on Sudoku Puzzle Using Computer Vision and Deep Learning, 2020, 643: 567–578, doi:10.1007/978-981-15-3125-555.

[7] Sudoku solving using patterns and graph theory [J]. International Journal of Emerging Technologies and Innovative Research, 2021, 8(3): 2219-2226. Available: http://www.jetir.org/papers/JETIR2103276.pdf

[8] Iyer R, Jhaveri A, Parab K. A Review of Sudoku Solving using Patterns [J]. International Journal of Scientific and Research Publications, 2013, 3(5).

[9] Musliu N, Winter F. A Hybrid Approach for the Sudoku Problem: Using Constraint Programming in Iterated Local Search [J]. IEEE Intelligent Systems, 2017, 32(2): 52-62, doi: 10.1109/MIS.2017.29

[10] Singh V, Sharma V, Bachchas V. Sudoku Solving Using Quantum Computer [J]. Ijraset Journal for Research in Applied Science and Engineering Technology, 2023, 128: 8.

[11] Herimanto P, Sitorus P, Zamzami E M. An Implementation of Backtracking Algorithm for Solving a Sudoku-Puzzle Based on Android [J]. J. Phys.: Conf. Ser., 2020, 1566: 012038.

[12] A Survey on Sudoku Solver Using Various Algorithms [J]. International Journal of Emerging Trends Technology in Computer Science (IJETTCS), 2017, 6(4): 123-129.

[13] Norvig P, Russel S. Artificial Intelligence: A Modern Approach. Third Edition [D]. Harlow: Pearson Education, 2016.

[14] Knuth, D. Dancing links [J]. arXiv Preprint, 2000.

[15] Harrysson M, Laestander H. Solving Sudoku efficiently with Dancing Links. (Degree Project in Computer Science, DD143X) [D]. Stockholm, Sweden: Stockholm University, 2014.