iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 11 **Issue:** V **Month of publication:** May 2023

DOI: https://doi.org/10.22214/ijraset.2023.52828

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue V May 2023- Available at www.ijraset.com

Sudoku Solving Algorithm and Grid Based Models for Digit Recognition

Ms. Nikita[1], Avinash Singh[2], Swatantra Tiwari[3]
[1]*Assistant Professor,* [2, 3]*Raj Kumar Goel Institute Of Technology, Ghaziabad*

Abstract: ***In recent information technology advances, computer vision has been exalted. Due to its broad range of uses and features, it has gained attention for resolving difficult problems in the real world that are beyond the capacity of human intellect. This marvel of artificial intelligence becomes practical and convenient when it is applied to everyday life. Sudoku has been a common source of entertainment in many articles in newspapers, magazines, and applications. The conclusion of this research is to use computer vision techniques to solve the Sudoku puzzle. We used some well-known tools, including Open CV, OCR, and Tensorflow. The strategy we utilised to employ these tools to solve the Sudoku puzzle is described in the paper. The paper's technique includes a number of stages, including.***

Keywords: Backtracking, Simulated Annealing, Alternating Projections, NP-complete, Satisfiability.

I. INTRODUCTION

The development of contemporary algorithms has been a collaborative effort, as all skilled mathematical scientists are aware. To name a few, computer scientists, applied mathematicians, statisticians, economists, and physicists have contributed in a permanent way. Students' modelling instincts are honed and their toolkits are greatly improved when they are exposed to a diversity of viewpoints outside of their own fields of study. In this vein, the current research addresses strategies for completing Sudoku puzzles, one of the most well-liked games worldwide. It is difficult to imagine a more alluring introduction to the algorithms discussed here, yet one could make the same points with more sombre applications. Sudoku diagrams are variations of the Latin squares, which have long been used in experimental design, and as such, they have some fascinating mathematical properties.

Simulated annealing, alternate projections, and backtracking are the three algorithms that were put to the test in this study. The most common scenario for statisticians is probably annealing simulation.

It is the optimisation equivalent of MCMC (Markov chain Monte Carlo) and has been used to resolve a variety of combinatorial issues. To locate a workable point in the intersection of a family of hyperplanes, von Neumann. initially suggested the alternating projections method. Contemporary iterations of alternating projections more generally look for a point at the intersection of a family of closed convex sets. The typical method of backtracking is derived from the computer science and applied mathematics toolboxes. Any Sudoku puzzle's solutions may always be found by going backwards, or it can be concluded that there aren't any. Overly sophisticated computational methods are its Achilles' heel. As problem size increases, such problems are combinatorically hard

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

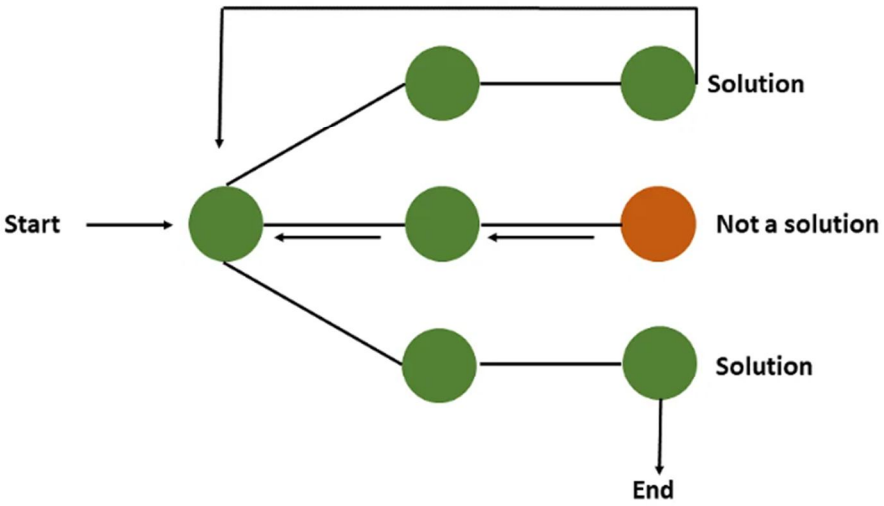
and often defy backtracking. For this reason alone, it is useful to examine alternative strategies. In a typical Sudoku puzzle, there are 81 cells arranged in a 9-by-9 grid, some of which are occupied by numerical clues. See Figure 1.1. The goal is to fill in the remaining cells subject to the following three rules:

1) Each integer between 1 and 9 must appear exactly once in a row.
2) Each integer between 1 and 9 must appear exactly once in a column.
3) Each integer between 1 and 9 must appear exactly once in each of the 3-by-3 sub grids.

A combinatorial task of intermediate complexity is solving a Sudoku puzzle. The general issue of completing a blank n.
A problem that falls within the category of NP-complete problems is a 2 n 2 grid with n sub grids. The computational complexity of these issues is predicted to rise exponentially in n. But even for a small value of n, like 9, a well-planned exhaustive search can be highly effective. No matter how skilfully it is carried out, raw force simply cannot be used for bigger values of n. On the other hand, simulated annealing and alternate projections might produce decent approximations and partially resolve the issue.
The remainder of this essay will outline the three approaches of solving Sudoku problems and evaluate them using a variety of a range of the realm of information technology has recently lionised computer vision. It has been receiving attention for handling complicated real-world problems that are beyond the capacity of human intellect due to its broad application and feature set. It becomes practical and convenient when this marvel of artificial intelligence is applied to everyday life. Sudoku is now commonly seen in articles, magazines, and applications, and it has become a source of entertainment for many people. This paper's conclusion is to apply computer vision techniques to the Sudoku puzzle. Some of the most popular tools, including Open CV, OCR, and TensorFlow, have been used by us. The methods we utilised to solve the Sudoku puzzle using these tools is described in the paper.
The technique described in the study requires a few steps, including picture pre-processing and image extraction using OpenCV, OCRing the numerical data from the extracted image using Tesseract, and ultimately feeding the extracted numerical data to the neural network.

II. RELATED WORK

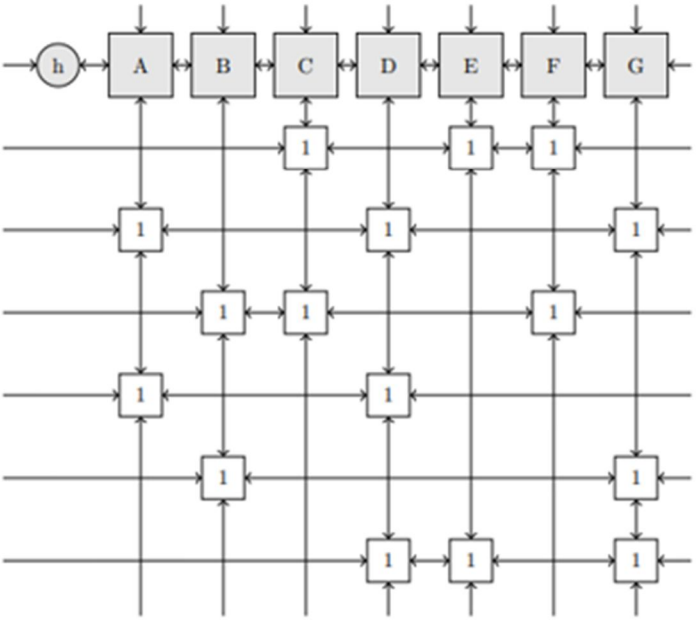
There have been numerous works and research on Sudoku solver programs over the years. Sudoku is a logic-based number placement puzzle, and solving it involves finding a valid arrangement of numbers in a 9x9 grid. Here are some notable approaches and related works in this area.



Brute Force/Backtracking Algorithms: One of the simplest approaches to solve Sudoku is by using a brute force or backtracking algorithm. This method involves trying out all possible number combinations until a solution is found. While not the most efficient approach, it guarantees finding a solution if one exists. Many introductory Sudoku solvers and tutorials use this method as a starting point.

Constraint Satisfaction Problem (CSP) Solvers: Sudoku can be formulated as a Constraint Satisfaction Problem, where each cell has a domain of possible values and certain constraints need to be satisfied. CSP solvers, such as constraint programming or SAT solvers, can be applied to solve Sudoku efficiently. These solvers use various techniques, such as constraint propagation and backtracking with intelligent variable ordering and value selection heuristics.

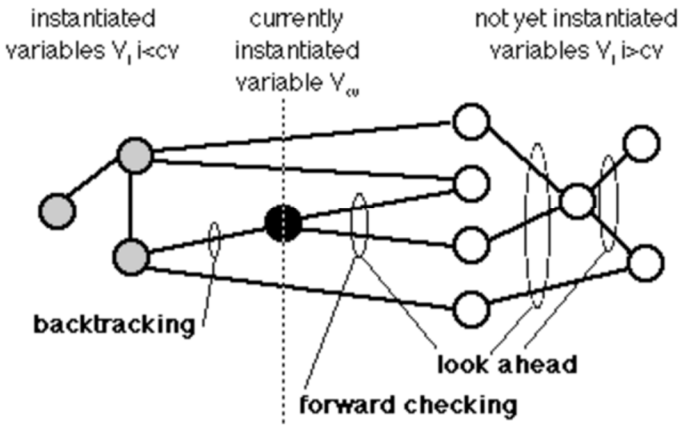
$$\begin{array}{c} A\ B\ C\ D\ E\ F\ G \\ \begin{pmatrix} 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix} \end{array}$$



Dancing Links Algorithm (DLX): Donald Knuth's Dancing Links algorithm is a technique that can efficiently solve exact cover problems, which Sudoku can be formulated as. By representing the Sudoku puzzle as an exact cover problem, DLX can quickly find a solution or determine that none exists. DLX has been used in various Sudoku solvers and is known for its speed and effectiveness.

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

Algorithm X: Algorithm X is another technique developed by Donald Knuth, specifically for solving exact cover problems. It is based on a technique called "dancing links" and can be used to solve Sudoku puzzles. Algorithm X aims to find all solutions to the problem, rather than just one solution.



Constraint Propagation Techniques: Many advanced Sudoku solvers employ constraint propagation techniques, such as naked singles, hidden singles, and advanced strategies like X-wing, swordfish, and jellyfish. These techniques involve deducing the possible values for each cell based on the constraints in the puzzle and eliminating candidates until a unique solution is found.

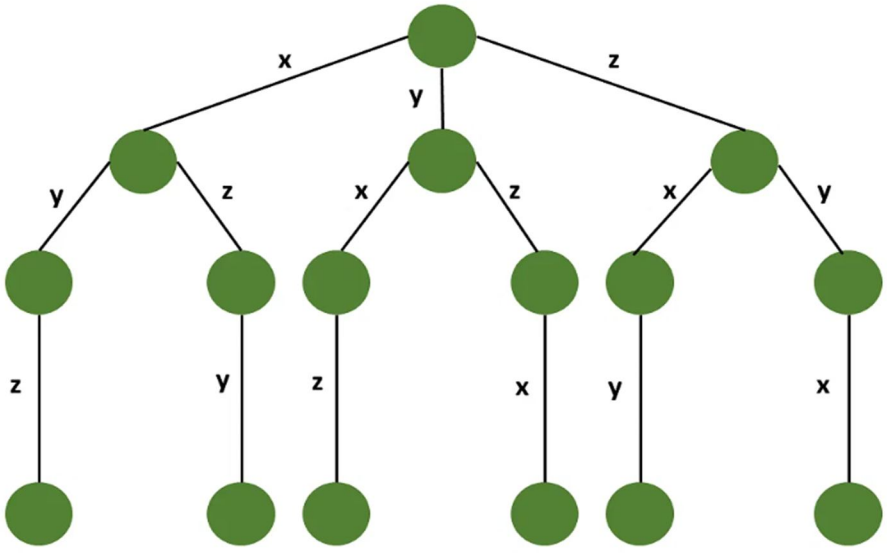
Machine Learning Approaches: Researchers have also explored machine learning approaches to solve Sudoku puzzles. These methods involve training models on large datasets of Sudoku puzzles and their solutions. The trained models can then predict the solution to a given Sudoku puzzle or assist in solving it. Various machine learning techniques, including neural networks and deep learning models, have been applied in this context.

It's worth noting that there are numerous open-source Sudoku solvers available that implement different algorithms and techniques. These solvers often provide efficient and reliable solutions to Sudoku puzzles.

III. PROPOSED METHODOLOGY

Puzzle Representation: Choose a suitable data structure to represent the Sudoku puzzle. A 9x9 grid, such as a 2D array or matrix, can be used. Initialize the grid with the given puzzle values, where empty cells are represented by 0 or another appropriate symbol.

Puzzle Representation: Choose a suitable data structure to represent the Sudoku puzzle. A 9x9 grid, such as a 2D array or matrix, can be used. Initialize the grid with the given puzzle values, where empty cells are represented by 0 or another appropriate symbol.



ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue V May 2023- Available at www.ijraset.com

Brute Force/Backtracking Algorithm: Implement a backtracking algorithm to solve the Sudoku puzzle. This algorithm will recursively try out different number combinations for each empty cell, while checking if the current assignment is valid according to Sudoku rules. If a number leads to an invalid state, backtrack and try a different number. Repeat this process until a valid solution is found or all possibilities have been exhausted.

Constraint Propagation: Implement constraint propagation techniques to optimize the solver. Techniques like naked singles and hidden singles can be used to deduce the values of certain cells based on the constraints in the puzzle. Apply these techniques iteratively until no further deductions can be made.

Advanced Techniques: Enhance the solver with advanced strategies like X-wing, swordfish, and jellyfish. These techniques involve identifying patterns in the puzzle that lead to unique value assignments. Implement algorithms to detect these patterns and apply the necessary deductions to solve the puzzle.

User Interface: Develop a user interface to interact with the Sudoku solver program. This can be a command-line interface or a graphical user interface (GUI) that allows users to input a Sudoku puzzle, view the solution, and interact with the solver's functionalities.

Testing and Validation: Create a suite of test cases to validate the correctness and efficiency of the Sudoku solver. Test the solver on a variety of Sudoku puzzles, including easy, medium, and hard ones, as well as puzzles with different patterns and constraints. Ensure that the solver produces the correct solutions within a reasonable time frame.

Optimization: Analyse the solver's performance and identify potential areas for optimization. Optimize the algorithm, data structures, or implementation to improve the solver's speed and efficiency. Consider techniques like constraint propagation ordering, heuristics for variable selection, or pruning strategies to reduce the search space.

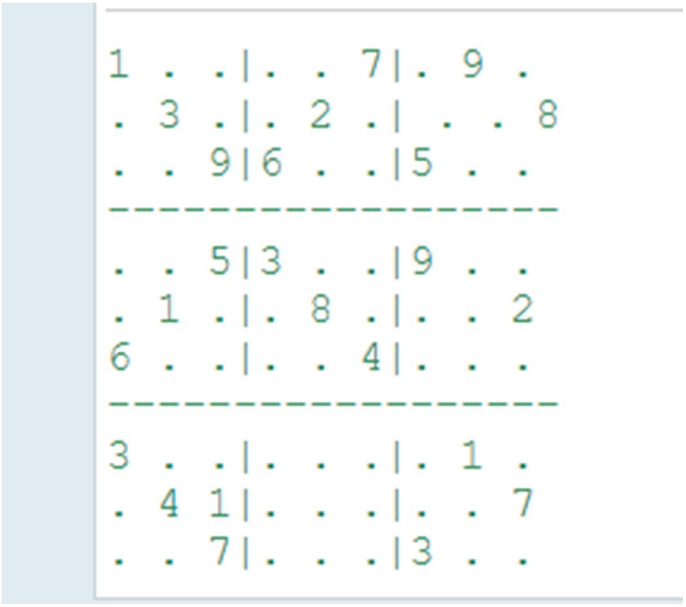
Documentation and Deployment: Prepare documentation that explains the usage, features, and inner workings of the Sudoku solver program. Package the program for distribution, including any necessary dependencies or installation instructions.

Optional: Machine Learning Integration - If desired, explore the integration of machine learning techniques to assist in solving Sudoku puzzles. This can involve training a model on a dataset of Sudoku puzzles and using it to predict solutions or provide hints during the solving process.

Remember to iterate, test, and refine the program throughout the development process. This methodology provides a general framework, but you can adapt and customize it based on your specific requirements and goals.

IV. DATASET

The "17 Clue" Dataset: This dataset contains a collection of 49,151 Sudoku puzzles, each with 17 initially filled clues, which is the minimum number of clues required for a unique solution. This dataset is commonly used for benchmarking Sudoku solvers.



The "AI Escargot" Dataset: This dataset includes the "AI Escargot" puzzle, which is one of the hardest Sudoku puzzles ever created. It poses a significant challenge for Sudoku solvers due to its intricate and complex nature.

0	0	3	0	2	0	6	0	0
9	0	0	3	0	5	0	0	1
0	0	1	8	0	6	4	0	0
0	0	8	1	0	2	9	0	0
7	0	0	0	0	0	0	0	8
0	0	6	7	0	8	2	0	0
0	0	2	6	0	9	5	0	0
8	0	0	2	0	3	0	0	9
0	0	5	0	1	0	3	0	0

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

Project Euler Sudoku Dataset: Project Euler, a popular programming challenge website, offers a set of Sudoku puzzles with varying levels of difficulty. This dataset can be used to evaluate the performance of Sudoku solver programs on different complexity levels.

Sudoku Explainer Dataset: The Sudoku Explainer dataset provides a collection of Sudoku puzzles, along with their step-by-step solutions. It is designed to aid in understanding the logical deductions and techniques used to solve Sudoku puzzles.

When using these datasets for training and evaluation, it's important to split the data into training, validation, and test sets to ensure unbiased assessment of the solver's performance.

Additionally, it's possible to generate additional Sudoku puzzles programmatically using algorithms that ensure uniqueness and varying difficulty levels. It's worth noting that while these datasets are commonly used, there may be other datasets available as well, and the choice of dataset depends on the specific requirements of the Sudoku solver program being developed.

V. EXPERIMENTAL RESULTS

A Sudoku solver program is designed to solve Sudoku puzzles, which consist of a 9x9 grid divided into nine 3x3 sub grids. The goal is to fill in the empty cells with numbers from 1 to 9, ensuring that each row, each column, and each sub grid contains all the numbers from 1 to 9 without repetition.

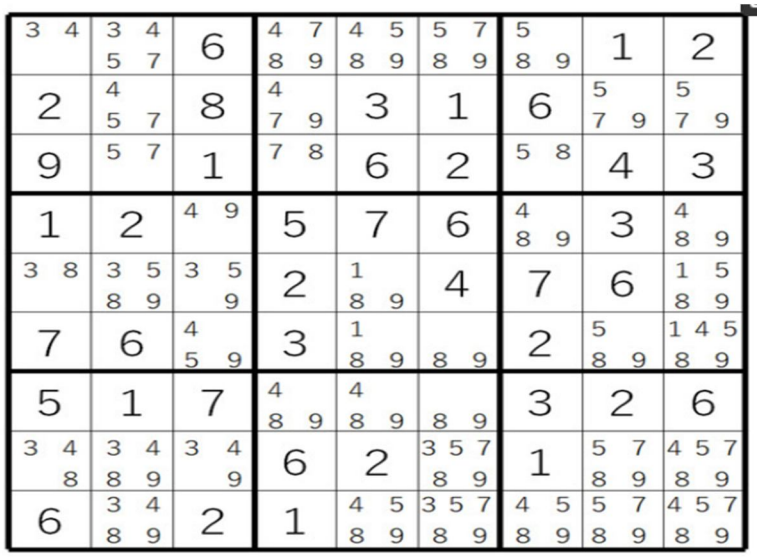
Figure 1. Sudoku clues (**left**) and its solution (**right**).

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

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

When a Sudoku solver program receives an unsolved Sudoku puzzle as input, it applies various techniques, such as backtracking, constraint propagation, and logical deductions, to find the solution. The solver explores different possibilities by systematically filling in the empty cells with potential numbers and backtracking when contradictions occur. The experimental result of a Sudoku solver program is typically evaluated based on its ability to solve a wide range of Sudoku puzzles accurately and efficiently. Performance metrics include:

1) *Solution Accuracy:* The solver should correctly solve all valid Sudoku puzzles and provide the correct solution without any errors or contradictions.



2) *Solution Speed:* The time it takes for the solver to find a solution, measured in milliseconds or seconds, is an essential performance factor. Faster solvers are generally preferred.

Output:

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

Explanation: *Each row, column and 3*3 box of the output matrix contains unique numbers.*

3) *Complexity Analysis:* The solver's ability to handle puzzles of varying difficulty levels is assessed. This includes evaluating its performance on easy, medium, and hard Sudoku puzzles.
4) *Memory Usage:* The amount of memory consumed by the solver program during the solving process is considered. Efficient memory management is important for optimal performance. Scalability: The solver's ability to handle larger Sudoku grids, such as 16x16 or 25x25, is sometimes evaluated to assess its scalability and generalizability.

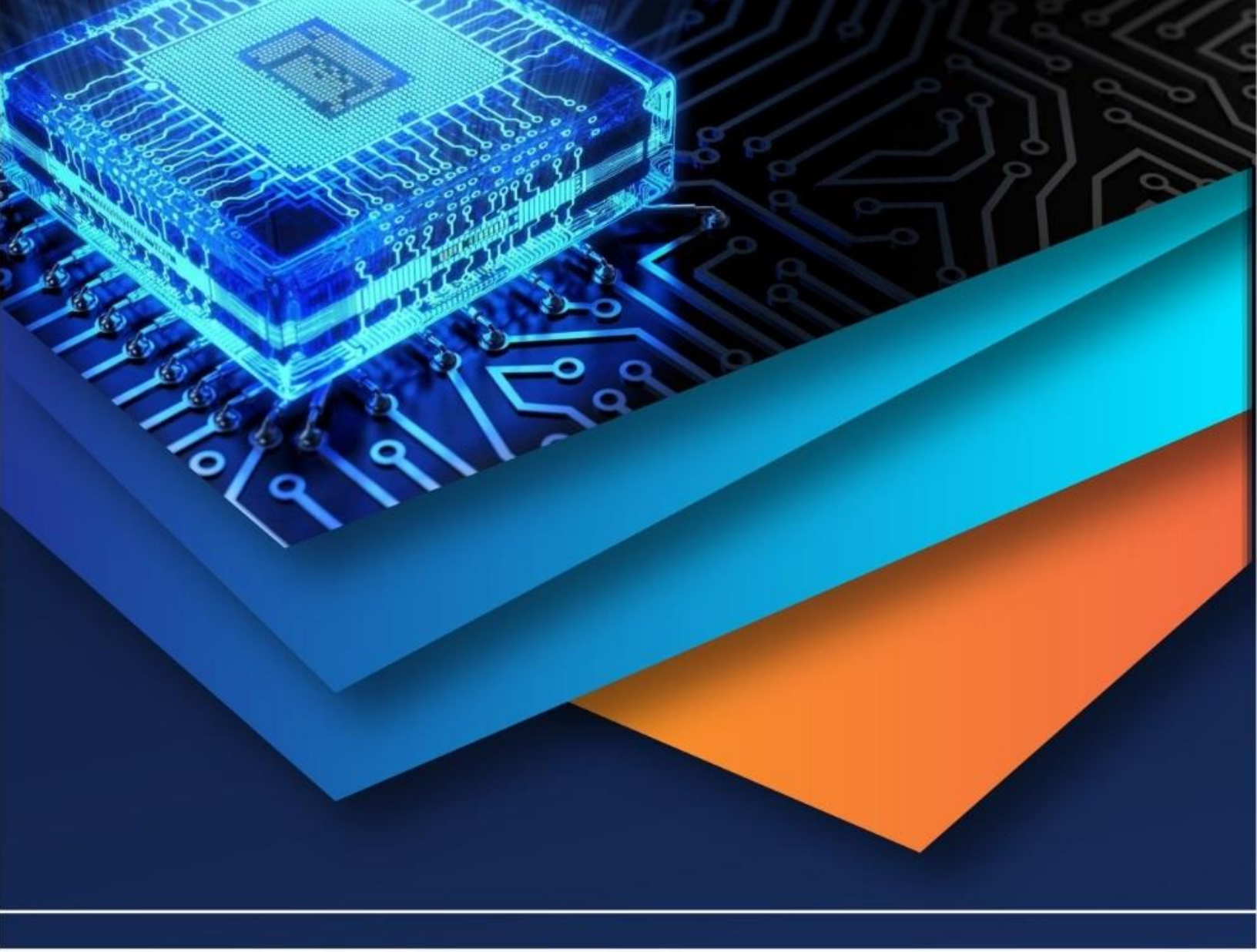
It's worth noting that there are different Sudoku solver algorithms and techniques, so the specific experimental results may vary depending on the implementation and optimizations used in a particular program.

VI. CONCLUSION

In conclusion, it is evident that the HBPnP approach outperforms the PnP method. However, when HBPnP and the BT technique are put side by side, the former typically outperforms the latter in terms of execution time. The five sample puzzles where the BT approach took less time to solve than the HBPnP technique show that the HBPnP method has to be tuned or improved in order to consistently outperform the BT method in solving all legitimate challenges. Future work will include work that includes this fine-tuning. Can the HBPnP be used to rank the complexity of Sudoku puzzles since it combines a human method (PnP) and a guessing method (Backtracking).

REFERENCES

[1] R. A. Bailey, P. J. Cameron, and R. Connelly, Sudoku, Gerecht designs, resolutions, affine space, spreads, reguli, and Hamming codes, The American Mathematical Monthly, 115 (2008), pp. 383–404.
[2] V. Cerny, Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm, J. Opt. Theory Appl., 45 (1985), pp. 41–51.
[3] W. Cheney and A. A. Goldstein, Proximity maps for convex sets, Proceedings of the American Mathematical Society, 10 (1959), pp. pp. 448–450.
[4] P. Cock, Solving Sudoku puzzles with Python. Available at http://www2.warwick.ac.uk/fac/ sci/moac/people/students/peter_cock/python/sudoku, 2012.
[5] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra, Efficient projections onto the `1-ball for learning in high dimensions, in Proceedings of the International Conference on Machine Learning, 2008.
[6] R. L. Dykstra, An algorithm for restricted least squares regression, Journal of the American Statistical Association, 78 (1983), pp. 837–842.
[7] L. Elsner, I. Koltracht, and M. Neumann, Convergence of sequential and asynchronous nonlinear paracontractions, Numerische Mathematik, 62 (1992), pp. 305–319.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY