Journal of Education and Learning (EduLearn)
Vol. 19, No. 1, February 2025, pp. 561~569
ISSN: 2089-9823 DOI: 10.11591/edulearn.v19i1.21214

Sudoku solutions: a comparative analysis of breadth-first search, depth-first search, and human approaches

Norizan Mat Diah, Syahirul Riza, Suzana Ahmad, Norzilah Musa, Shakirah Hashim
School of Computing Sciences, College of Computing, Informatics and Mathematics, Universiti Teknologi MARA, Selangor, Malaysia

Article Info

Article history:

Received Aug 22, 2023
Revised Apr 16, 2024
Accepted May 18, 2024

Keywords:

Breadth-first search
Depth-first search
Human
Puzzles
Sudoku

ABSTRACT

Sudoku is a puzzle that has a unique solution. No matter how many methods are used, the result will always be the same. The player thought that the number of givens or clues, the initial value on the Sudoku puzzles, would significantly determine the difficulty level, which is not necessarily correct. This research uses two search algorithms, breadth-first search (BFS) and depth-first search (DFS), to solve a set of Sudoku puzzles based on the number of givens. The Sudoku puzzles are chosen based on the number of givens between 32 and 35. In cases where Sudoku puzzles are considered medium or intermediate difficulty, the solutions generated by both algorithms will be compared with the human-solving approach. The research aims to determine whether humans tend to solve Sudoku puzzles with solutions resembling those generated by BFS or DFS. Furthermore, if all three approaches-human, BFS, DFS-yield comparable solutions, the Sudoku puzzle has only one unique solution.

Corresponding Author:

Norizan Mat Diah
School of Computing Sciences, College of Computing, Informatics and Mathematics
Universiti Teknologi MARA
40450 Shah Alam, Selangor, Malaysia
Email: norizan289@uitm.edu.my

1. INTRODUCTION

Sudoku is a game that challenged player to use common sense and logic in solving number grid puzzles [1]. It was invented by Howard Garnes, a freelance puzzle constructor, in 1979 [2]. The Japanese adopted it as "Suuji Wa Dokushin Ni Kagiru" [3], [4]; then, in 2005, it was abbreviated to just "Sudoku," which gained enormous popularity. Sudoku grew in popularity after being published in newspapers as a regular feature. The 9-by-9 grid has been the most common grid size, with 6,670,903,752,021,072,936,960 possible combinations. The puzzle becomes more complex as the grid size increases, and for a 16-by-16 grid, there are approximately $5.96x^{1098}$ possible combinations [5]. Sudoku is a necessary feature for many newspapers and magazine companies. When the popularity of Sudoku kept increasing, books that included lots of Sudoku puzzles were published and distributed worldwide, which can be found in any bookstore. Sudoku is also available online, either in web-based or even mobile applications.

Generally, the publicly known Sudoku puzzles comprise 81 squares consisting of 9 columns and 9 rows. The big box is then divided into 9 partial squares, each consisting of 3×3 squares as shown in Figure 1(a). Players may fill the boxes with numbers ranging from 1 to 9 [6]–[8]. This Sudoku game begins with a few randomly chosen numbers from 1 to 9, along with boxes that provide clues for solving Sudoku puzzles as shown in Figure 1(b). The player's task is to complete other empty or unfilled squares to fill the entire Sudoku board with numbers [9]. The requirement is that no numbers should be repeated in one column, one row, or any 3×3 partial box [10]. Players are encouraged to continue playing; unlike other number games,

Sudoku provides players' logical intelligence to solve Sudoku by filling in all the boxes without flouting the existing rules. A perfect Sudoku puzzle has just one solution [11], [12]. Sudoku is a puzzle game that focuses on space and circumstances. Sudoku has simple rules, requiring players to fill in an empty box with random numbers. However, even though it is simple in terms of the rules, it turns out that many players cannot complete it. The number of clues on the grid at the beginning of the Sudoku game determines its difficulty level. The fewer clues on the grid, the more difficult the Sudoku will be to solve [13]. The game Sudoku Puzzles requires players to use brain power and concentration in problem-solving strategies [14], [15].

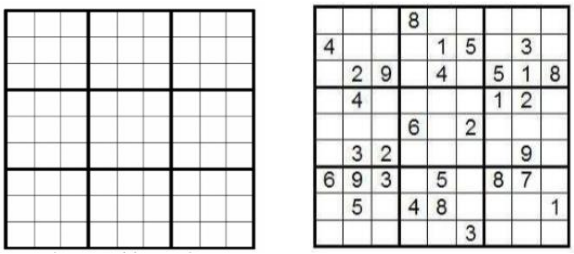


Figure 1. Boxes that provide clues for solving Sudoku puzzle, (a) Example of 9×9 grid of Sudoku and (b) Example of 9×9 grid Sudoku with clues numbers

2. BACKGROUND STUDY

A unique solution has been designed for each regular Sudoku puzzle [16], [17], meaning there is no other possible correct answer for each space in the grid. Therefore, if solvers find themselves guessing between various possible answers, they may guess it incorrectly, or the Sudoku they try to solve is not being developed properly. Research by Chandra and Hareendran [18], and Astrid [19] stated that it is just a myth that the difficulty of Sudoku puzzles is related to the number of clues. There are Sudoku puzzles with many numbers of clues, but they are challenging to solve [20], and there are also Sudoku puzzles with a low number of clues that can be easily solved [21]. Thus, the question is whether the number of clues has a substantial impact on differentiating AI strategies while solving Sudoku puzzles. For this purpose, solving Sudoku is being modeled as binary integer linear programming (BILP) [22] and implemented in three programming languages, Python, Julia and Minizinc. Apparently, computation time for solving BILP problems is getting faster as the difficulty level of Sudoku questions increases, which contradicts with manual Sudoku completion by human. The main reason is that Sudoku with easy difficulty contains more clues, resulting in a higher number of constraints that need to be fulfilled compared to higher difficulty levels with fewer given clues.

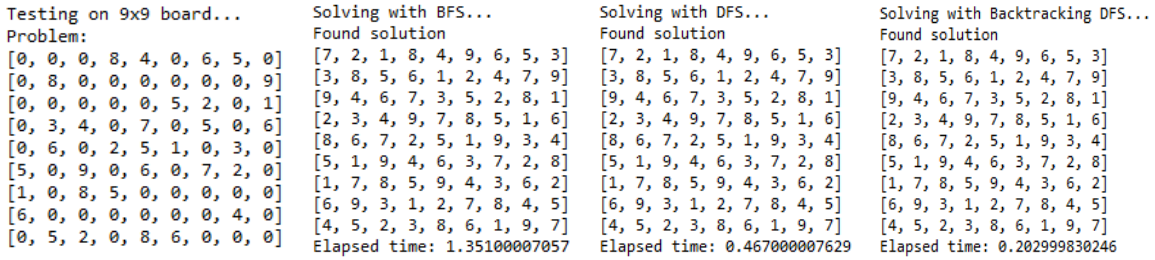
While solving Sudoku involves filling the empty grid with unique numbers in each row, column, and sub-block, diagonal Sudoku requires an additional constraint to ensure that there are no duplicate numbers along the diagonal regions. Research by Ahmad *et al.* [23] proposed algorithm to solve diagonal Sudoku using Java program. Still following backtrack approach, but the duplication checking need to be perform on row, column, sub-block and diagonal region. Unfortunately, this programme cannot differentiate between the given puzzles, whether they have unique or many solutions.

Sudoku is a number-placement puzzle game and can be considered logical. Unlike other strategy games, Sudoku is exceptionally intricate due to its nature as an NP-Complete problem, which cannot be rapidly and easily solved, as indicated by Levonyan *et al.* [24], and Asif and Baig [25]. However, there is no known algorithm for finding a solution to this Sudoku problem. As a result, the computational time needed to solve the problem increases rapidly compared to its size. Sudoku is also a kind of constraint satisfaction problem (CSP). In the context of CSP, the problem necessitates obtaining a solution set for variables and the imposition of constraints on the available state of values [26], [27].

A basic technique for solving a Sudoku puzzle involves searching for the answer one cell at a time [28], using the most fundamental search strategy: depth-first search (DFS) and breadth-first search (BFS). Both algorithms employ backtracking, returning to a previously explored branch in their search route to expand and explore additional pathways when necessary. When these algorithms are performed on specific test problems, DFS works well on low- and medium-complexity puzzles but struggles with many steps on the harder ones. On the other hand, BFS works well on 4×4 problems but is highly sluggish on all other puzzles since it tests every intermediate alternative to the solution.

A Python program to compare algorithms between BFS, DFS, and backtracking depth-first search (BDFS) has been developed and make available at github.com [29]. Figure 2 shows the stages of solving

sudoku, of which Figure 2(a) shows the initial stage of 9×9 grid of Sudoku problem while Figures 2(b) to (d) show Sudoku grid solved with BFS, DFS and BDFS respectively. Solving Sudoku with BDFS and DFS algorithm is significantly fast, which take less than one second compared to BFS, requires more than one second. However, the outcomes may sometimes vary due to more than one possible solution in Sudoku.



(a) (b) (c) (d)

Figure 2. Stages of solving sudoku, (a) Sudoku initial stage puzzle, (b) Solving with BFS, (c) Solving with DFS, and (d) Solving with BDFS

A comparison between uninformed and AI algorithms for N puzzle solutions has been conducted in [30] using BFS, DFS, A* Search, Best First Search, and Hill Climbing Search to compare the results of the N puzzle. The study was conducted to determine the extent of each algorithm in solving the puzzle. Best First Search is suitable for a shorter solution, while A* Search is better for a long and complex solution. Meanwhile, Everitt and Hutter [31] conducted an analysis of the algorithm selection problem using tree search for BFS and DFS, while Everitt and Hutter [32] focused on graph search methods. A breadth-first algorithm for BFS and depth-limited search (DLS) was proposed in the Sudoku Game [11] aiming to find Sudoku solutions and evaluate the effectiveness of the two algorithms. The findings have demonstrated that DLS is faster and more efficient than BFS, whereas BFS offers more structured and systematic search system capable of finding all potential numbers in each box.

Solving Sudoku using searching by column and sub-block technique is proposed in [33] with the integration of genetic algorithm (GA) which significantly improved searching speed, resulting in shorter Sudoku solving times. Backtracking algorithm has also proven effective in solving Sudoku as shown in [34]. The algorithm is only considering potential elements that leads to success. However, a thorough testing with more complex cases, such as a more intricate artificial intelligence game, is advisable to ensure the algorithm performs optimally.

Numerous research studies have explored the application of both BFS and DFS algorithms in accelerating the resolution of Sudoku. Notably, [35] specifically investigated the impact of BFS and DFS algorithms on execution time and explored their interactions with nodes during the Sudoku-solving process. The findings from these studies significantly contribute to our understanding of search algorithms and their practical applicability in efficiently solving Sudoku problems in real-world scenarios.

Furthermore, the research presented in [36]–[38] delves into formulating Sudoku problems using the entropy concept, particularly focusing on solving extremely challenging Sudoku puzzles. Beyond being a recreational game, Sudoku holds substantial importance in various real-world applications in daily life and industrial engineering. These applications span diverse fields such as data encryption [39], radar waveform design [40], and education [41]. Despite the multitude of methods explored for solving Sudoku problems, it is noteworthy that algorithms involving human input have not yet been fully implemented. The study emphasizes that in real-life situations, the contribution of human thinking to solve Sudoku is more significant than in a purely digital environment, underscoring the complexity and relevance of Sudoku-related problem-solving across different domains.

3. METHODOLOGY

This study has proposed an algorithm to solve Sudoku based on the number of givens (clues). Currently, the graphical user interface (GUI) only uses the integrated development environment (IDE). Hence, every input and output of the application developed will be displayed on IDE. The development process is divided into three stages: the coding stage, the testing session, and the comparison of the results. Figure 3 shows the flow chart of the proposed algorithm to solve Sudoku, while Figure 4 is a snippet code of a search class for

the proposed algorithm. The search class is the parent for BFS and DFS; the function inside is to validate what numbers can be filled in the current cell.

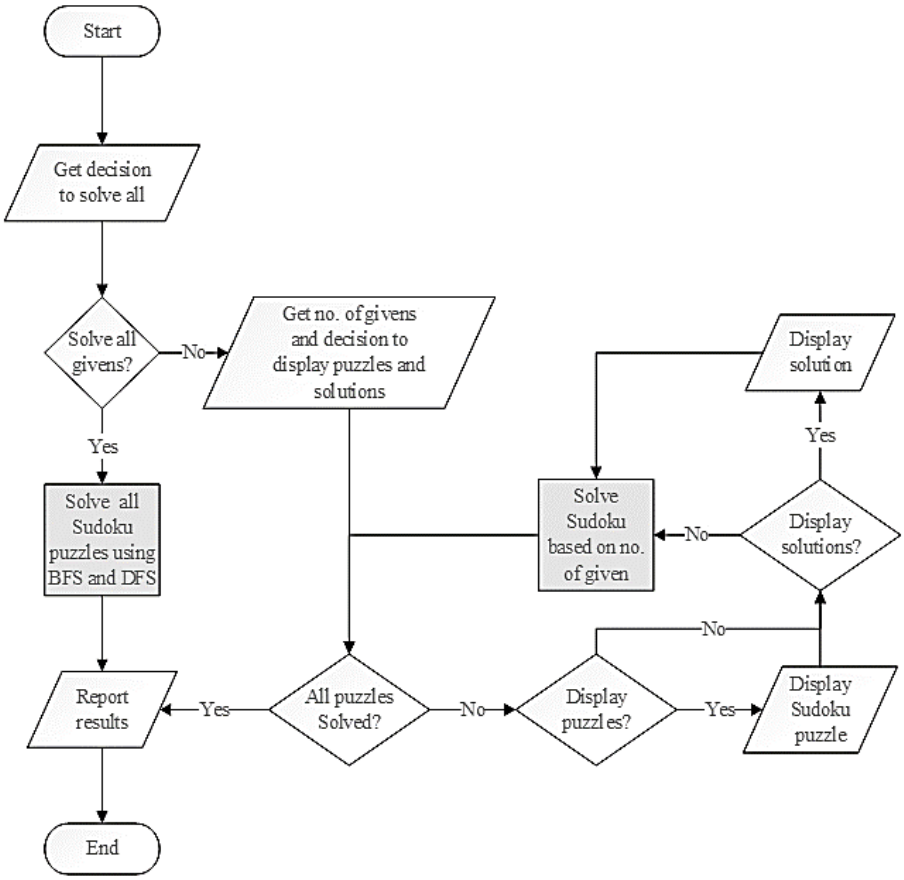


Figure 3. System flow

```
class Search:
    def __init__(self, initial):
        self.initial = initial

    @staticmethod
    # Return set of valid numbers that can be fill in current cell
    def valid_numbers_in_cell(num_set, used):
        return [number for number in num_set if number not in used]

    @staticmethod
    # Return row and col value for the first empty cell found
    def first_empty_cell(puzzle, state):
        for row in range(puzzle):
            for col in range(puzzle):
                if state[row][col] == 0:
                    return row, col

    @staticmethod
    def is_total_45(state):
        # Each row, column and box can only be filled from 1 - 9 with the sum of 45
        for row in range(9):
            if (len(state[row]) != 9) or (sum(state[row]) != 45):
                return False
            for col in range(9):
                if (len(state[col]) != 9) or (sum(state[col]) != 45):
                    return False

        # Check sum of the numbers in each box
        for column in range(0, 9, 3):
            for row in range(0, 9, 3):
                boxes = 0
                for box_row in range(0, 3):
                    for box_col in range(0, 3):
                        boxes += state[row + box_row][column + box_col]
                if boxes != 45:
                    return False
        return True
```

Figure 4. Coding example of search class

4. RESULTS AND DISCUSSION

The testing of these applications will be divided into two types: i) unit testing: referring to the testing of each function or procedure and ii) correctness testing: used to determine which techniques give the closest solutions like humans. Figure 5 refers to the application solving a specific set of Sudoku puzzles based on the number of givens. The user must select 'no' or 'n' before inputting a number of highlighted givens. This action will solely determine whether the Sudoku is solved or not. Figure 6 displays soduku solution puzzle, Figure 6(a) shows the initial Sudoku puzzle that has the same number of gifts Figure 6(b) displays the solution of the puzzle. After both actions displayed Sudoku puzzles, their solutions were successfully executed without any issues. Figure 7 shows the outcomes achieved by combining both functions.

When all the functions have been completed, Figure 8 corresponds to when the user intends to solve all existing puzzles without specifying a particular number of givens. This function usually takes more time, especially as the number of Sudoku puzzles increases. The Figure 9 show subsequent step involved a correctness test to identify solutions resembling how humans solve Sudoku puzzles. Figure 9(a), a human manually solves the Sudoku. Simultaneously, Figures 9(b) and 9(c) present the solutions to the same puzzle.

Upon examination of the figures above, no difference is noted between them. Hence, we can infer that this Sudoku puzzle is considered good, as a good Sudoku puzzle should have only one unique answer, regardless of the techniques used. This upcoming testing phase will compare the performance of BFS and DFS, as detailed in the next section.

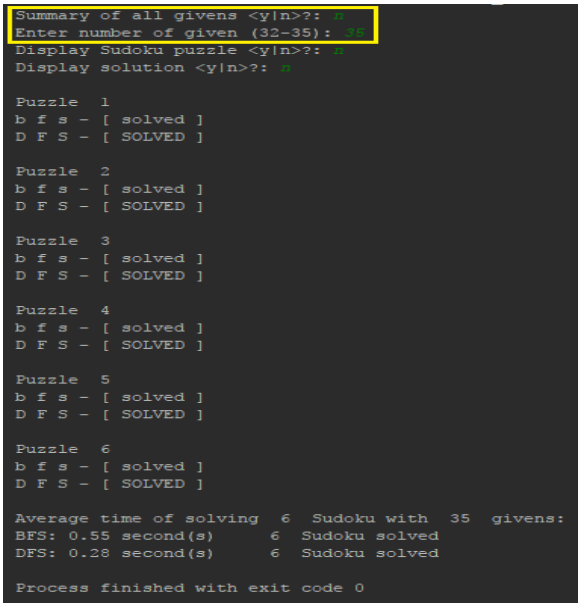
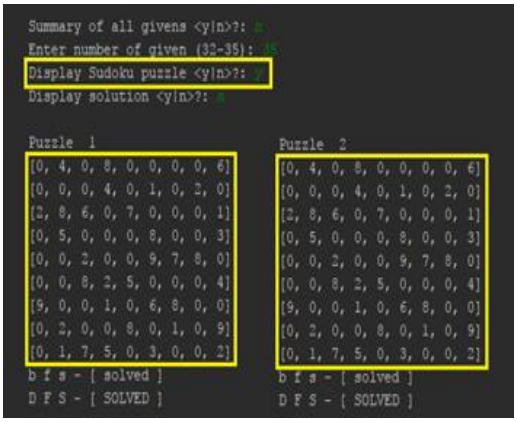
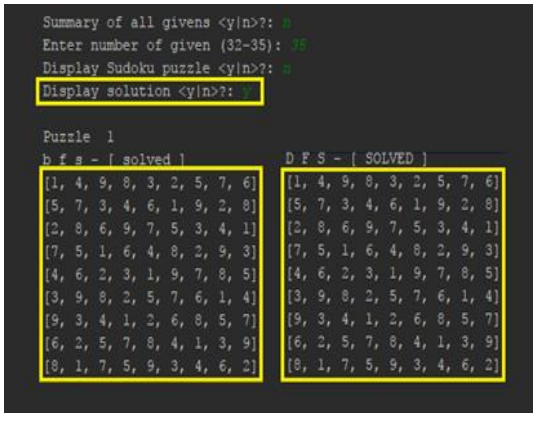


Figure 5. Solving sets of Sudoku puzzles based on several givens



(a)



(b)

Figure 6. Soduku solution puzzl, (a) display initial puzzles and (b) display solution

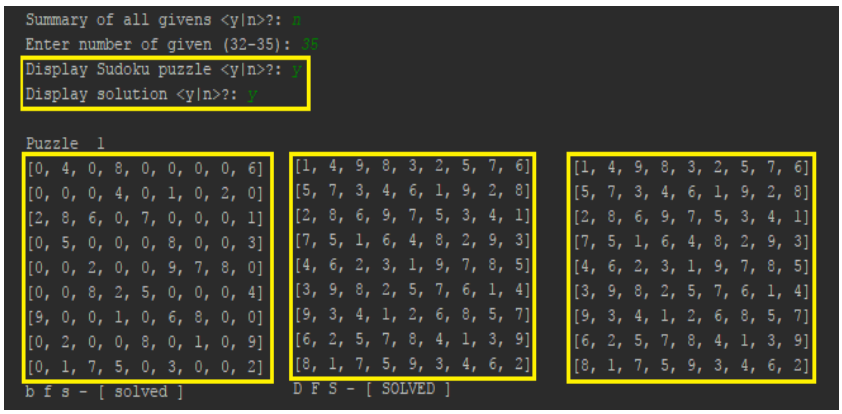


Figure 7. Display the initial puzzle and its solutions

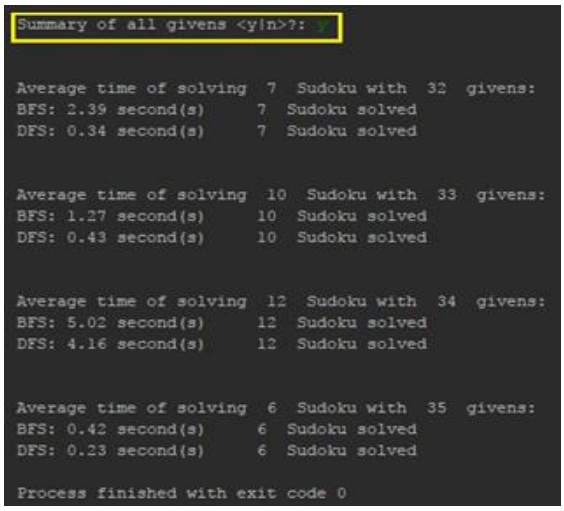
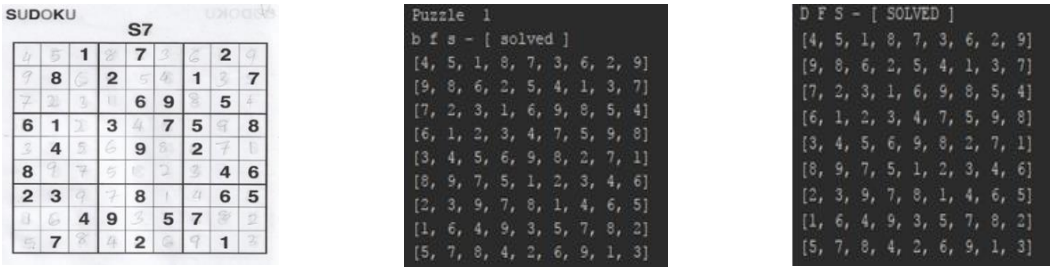


Figure 8. Solving every puzzle in each given



(a) (b) (c)

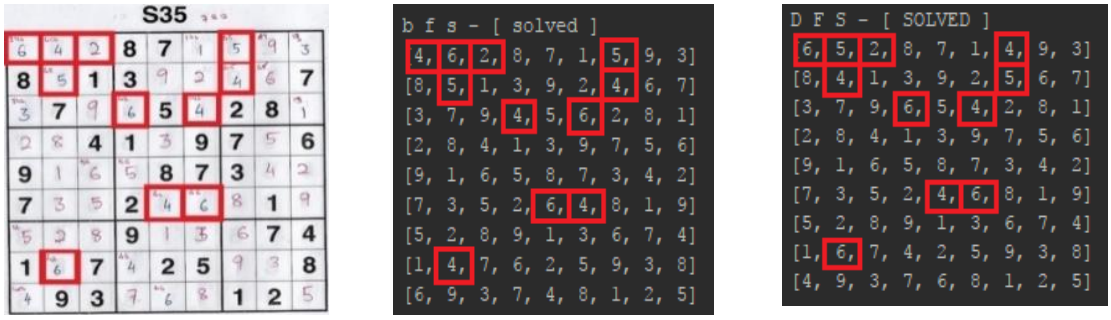
Figure 9. Step involves a correctness test to identify solutions resembling how humans solve sudoku puzzle, (a) Sudoku solved by human, (b) Sudoku solved by BFS, and (c) Sudoku solved by DFS

The results of solving Sudoku puzzles using both techniques are shown in Table 1. The comparison is based on how many puzzles each technique successfully solved and how much their solutions resemble human answers. A total of 33 Sudoku puzzles were employed in this research. The average time taken to solve a Sudoku puzzle per given indicates that DFS results in a faster solving time. The human solution can align with either BFS, DFS, or none. For cases where there is no alignment, refer to Figure 10, Soduko that has been solved, Figure 10(a) for the human solution, Figure 10(b) for the BFS solution, and Figure 10(c) for the DFS solution for a clearer understanding. The distinctions between the three have been highlighted in red boxes.

In cases where the human solution and algorithms yield similar results, it indicates that the puzzle being solved is a valid Sudoku. A valid Sudoku is one where the solution is unique, irrespective of the method employed; the solution remains consistent. Nearly half of the tested Sudoku puzzles in this research demonstrate a unique solution.

Table 1. Comparison results of solutions resembling human

No. of Givens	Total puzzles solved	Avg. time of solving a Sudoku (seconds)		Total solutions resembling human			
		BFS	DFS	BFS	DFS	Both	None
32	7	2.26	0.27	2	2	1	2
33	10	1.55	0.55	2	2	5	1
34	11	1.23	0.34	1	1	7	2
35	5	0.42	0.22	0	1	3	1
Total	33			5	6	16	6



(a) (b) (c)

Figure 10. Soduko that has been solved, (a) Human solution, (b) BFS solution, and (c) DFS solution

5. CONCLUSION

This research compares solutions generated by BFS or DFS to align more closely with human solutions. Additionally, it seeks to discern if there is a relationship between the number of givens and the likelihood of solutions resembling either BFS or DFS. The process involved in this study encompasses identifying the algorithms used to design the system and implementing these algorithms. Subsequently, the unit testing results were displayed, followed by correctness tests comparing human solutions to algorithmic solutions in solving Sudoku puzzles. The comparison revealed that when both algorithms and humans arrived at similar solutions, it indicated careful construction of the Sudoku puzzles, ensuring they possess only one unique solution.

ACKNOWLEDGEMENTS

Acknowledgement is extended to the College of Computing, Informatics, and Mathematics at Universiti Teknologi MARA, Shah Alam, Selangor, Malaysia, for their support in facilitating this work.

REFERENCES

[1] Y. L. T. Ting, "Learning to hypothesize with confidence through sudoku game play," in *English teaching forum*, 2009.

[2] S. Jana, A. K. Maji and R. K. Pal, "A novel Sudoku solving technique using column-based permutation," in *2015 International Symposium on Advanced Computing and Communication (ISACC)*, Silchar, India, 2015, pp. 71-77, doi: 10.1109/ISACC.2015.7377318.

[3] A. Maji, S. Jana, and R. K. Pal, "An algorithm for generating only desired permutations for solving sudoku puzzle," *Procedia Technology*, 2013, pp. 392–399, doi: 10.1016/j.protcy.2013.12.375

[4] K. A. Putrilani *et al*., "Effectiveness of sudoku game media in memorizing kana letters (using quasi-experimental method for japanese club students at UPI Pilot Laboratory Middle School) (in Indonesian: *efektivitas media permainan sudoku dalam menghafal huruf kana (menggunakan metode eksperimen quasi terhadap siswa Japanese club SMP Laboratorium Percontohan UPI)*", *JAPANEDU: Jurnal Pendidikan dan Pengajaran Bahasa Jepang*, vol. 1, no. 3, pp. 34-43, 2016.

[5] Y. Wu, J. P. Noonan, and S. Agaian, "Binary data encryption using the Sudoku block cipher," 2010 IEEE International Conference on Systems, Man and Cybernetics, Istanbul, Turkey, 2010, pp. 3915-3921, doi: 10.1109/ICSMC.2010.5641801.

[6] J.P. Delahaye, "The science behind sudoku," *Scientific American*, vol. 294, no. 6, pp. 80–87, June 2006, doi: 10.1038/scientificamerican0606-80.

[7] F.A. Rahman; D. Anubhakti, "implementation of the backtracking algorithm in the sudoku game (in Indonesian: *implementasi algoritma backtracking pada permainan sudoku*)", *MEANS (Media Informasi Analisa dan Sistem)*, vol. 5, no.1, pp. 67-71, 2020.

[8] K. Nishikawa and T. Toda, "Exact method for generating strategy-solvable sudoku clues" *Algorithms*, vol 13, no. 7, 171, 2020 doi: doi.org/10.3390/a13070171.

[9] D. Deodhare, S. Sonone and A. Gupta, "A generic membrane computing-based sudoku solver," *2014 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT)*, Ghaziabad, India, 2014, pp. 89-99, doi: 10.1109/ICICICT.2014.6781258.

[10] N. Kitsuwan, P. Pavarangkoon, H. M. Widiyanto, and E. Oki, "Dynamic load balancing with learning model for Sudoku solving system," *Digital Communications and Networks*, vol. 6, no. 1, pp. 108–114, feb 2020, doi: 10.1016/j.dcan.2019.03.002.

[11] J. Espasa, I. P. Gent, R. Hoffmann, C. Jefferson, A.M. Lynch, A. Salamon and M. J. McIlree, "Using small MUSes to explain how to solve pen and paper puzzles", *arXiv:2104.15040*, January 2021 doi: 10.48550/arXiv.2104.15040.

[12] T. N. Lina and M. S. Rumetna, "Comparison analysis of breadth first search and depth limited search algorithmsin sudoku game," *Bulletin of Computer Scienceand Electrical Engineering*, vol. 2, no. 2, pp. 74-83, December 2021, doi: 10.25008/bcsee.v2i2.1146.

[13] B. V. Indriyono, N. Pamungkas, Z. Pratama, E. Mintorini, I. Dimentieva and P. Mellati, "Comparative analysis of the performance testing results of the backtracking and genetics algorithm in solving sudoku games" *Journal of Artificial Intelligence & Robotics (IJAIR)*, vol. 5, no. 1, pp. 29-35, May 2023, doi: doi.org/10.25139/ijair.v5i1.6501.

[14] R. Sivakumar, "Effectiveness of memory game on academic performance of primary school students," *Global and Lokal Distance Education- Glokalde*, vol. 8, no. 1, pp. 15-23, April 2022.

[15] N. Siricharoen, "Creative brain training apps and games can help improve memory, cognitive abilities, and promote good mental health for the elderly," *EAI Endorsed Trans Context Aware Syst App*, vol. 9, no 1, July 2023, doi: 10.4108/eetcasa.v9i1.3524.

[16] A. Mohanty, A. Alam, R. Sarkar and S. Chaudhury, "Design and development of digital game based learning software for incorporation into school syllabus and curriculum transaction," *Design Engineering*, vol. 8, pp. 4864-4900, October 2021.

[17] R. A. Bailey, J. P. Cameron and R. Connelly, "Sudoku, gerechte designs, resolutions, affine space, spreads, reguli, and hamming codes", *The American Mathematical Monthly*, vol 115, no 5, 383-404, January 2018 doi: 10.1080/00029890.2008.11920542.

[18] S.S. Chandra and S. Hareendran. Artificial intelligence: principles and applications. Delhi India: PHI Learning Pvt. Ltd., 2020.

[19] E. Astrid. *The language of gaming*. London United Kingdom: Palgrave Macmillan Publishing, 2017.

[20] P. Kai, "An ACT-R model of skill development in the case of sudoku," Master dissertation, Faculty of Economics and Social Sciences, Albert-Ludwigs-Universität, Germany, 2018. [Online]. Available: https://osf.io/27zvc/download

[21] G. McGuire, B. Tugemann and G. Civario, "There Is No 16-clue sudoku: solving the sudoku minimum number of clues problem via hitting set enumeration, experimental," *Mathematics*, vol. 23, no 2, pp. 190-217, June 2014, doi: 10.1080/10586458.2013.870056.

[22] F. Bukhari, S. Nurdiati, M. K. Najib, and N. Safiqri, "Formulation of sudoku puzzle using binary integer linear programming and its implementation in Julia, Python, and Minizinc," *Jambura Journal of Mathematics*, vol. 4, No. 2, pp. 323–331, 2022, doi: 10.34312/jjom.v4i2.14194.

[23] N. I. S. Ahmad, N. Ismail, A.K. Khalid, and M.S.A. Halim, "Solving diagonal sudoku puzzle 9x9 grid by java programming" *Algorithm. Journal of Information System and Technology Management*, vol 4, no. 3, pp. 71-83, June 2019.

[24] K. Levonyan, J. Harder and F. D. M. Silva, "Automated graph genetic algorithm based puzzle validation for faster game design," *2022 IEEE Congress on Evolutionary Computation (CEC), Padua, Italy, 2022*, pp. 1-8, doi: 10.1109/CEC55065.2022.9870402.

[25] M. Asif and R. Baig, "Solving NP-complete problem using ACO algorithm," *2009 International Conference on Emerging Technologies*, Islamabad, Pakistan, 2009, pp. 13-16, doi: 10.1109/ICET.2009.5353209.

[26] A.S. Praneeth, "AI for solving sudoku puzzles," *Students Journal of AI*, vol. 1, no. 1, pp. 1-4, 2014.

[27] K. R. Chowdhary, "Constraint satisfaction problems," in *Fundamentals of Artificial Intelligence*, New Delhi India: Springer, 2020.

[28] B. De Kegel and M. Haahr, "Procedural puzzle generation: a survey," in *IEEE Transactions on Games*, vol. 12, no. 1, pp. 21-40, March 2020, doi: 10.1109/TG.2019.2917792.

[29] Tphanco, "Sample: solving sudoku using BFS, DFS, and backtracking DFS," [Online]. Available: https://kandi.openweaver.com/python/tphanco/sample, accessed October 18, 2022.

[30] K. Mathew, M. Tabassum and M. Ramakrishnan, "Experimental comparison of uninformed and heuristic AI Algorithms for N puzzle solution," *International Journal of Digital Information and Wireless Communications*, vol. 4, no. 1, pp. 143-154, January 2014.

[31] T. Everitt and M. Hutter, "Analytical results on the BFS vs. DF algorithm selection problem: part i: graph search," in *Australasian Joint Conference on Artificial Intelligence*. Canberra, Australia, 2015, pp. 157–165, doi: 978-3-319-26350-2_14.

[32] T. Everitt and M. Hutter, "Analytical results on the BFS vs. DFS algorithm selection problem: part ii: graph search," In *Australasian Joint Conference on Artificial Intelligence. Canberra*, Australia, 2015, pp.166–178, doi: 10.1007/978-3-319-26350-2_15.

[33] C. Wang *et al.* "A novel evolutionary algorithm with column and subblock local search for sudoku puzzles," *IEEE Trans. Games*, early access, Jan. 12, 2023, doi: 10.1109/TG.2023.3236490.

[34] Herimanto, P. Sitorus and E. M. Zamzami, "An implementation of backtracking algorithm for solving a sudoku-puzzle based on android," *Journal of Physics: Conference Series 1566 (2020) 012038*, IOP Publishing, doi: 10.1088/1742-6596/1566/1/012038.

[35] P. Garg, A. Jha and K. A. Shukla, "Randomised analysis of backtracking-based search algorithms in elucidating sudoku puzzles using a dual serial/parallel approach," *In Inventive Computation and Information Technologies*, Tamil Nadu, India, 2022, pp 281–295, doi: 10.1007/978-981-16-6723-7_21.

[36] J. Gunther, and T.K. Moon, "Entropy minimization for solving sudoku," *IEEE Transactions on Signal Processing*, vol. 60, pp. 508–513, 2012, doi: 10.1109/TSP.2011.2169253.

[37] N. Pathak, and R. Kumar, "Entropy guided evolutionary search for solving sudoku," *Prog Artif Intell*, vol. 12, pp. 61–76, 2023, doi: 10.1007/s13748-023-00297-7.

[38] A. Burlats and G. Pesant, "Exploiting entropy in constraint programming," in: Cire, A.A. (eds) *Integration of Constraint Programming, Artificial Intelligence, and Operations Research. CPAIOR 2023*. Lecture Notes in Computer Science, vol 13884. Springer, Cham, doi: 10.1007/978-3-031-33271-5_21.

[39] S. Jana, A. K. Maji, and R. K. Pal, "A novel SPN-based video steganographic scheme using Sudoku puzzle for secured data hiding," *Innovations Syst. Soft. Eng*., vol. 15, no. 1, pp. 65-73, Jan. 2019.

[40] R. M. Nicholas, D. B. Travis, and M. N. Ram, "Sudoku based phase coded radar waveforms," *Radar Sensor Technology XXV*, vol. 11742, pp. 117420L, April 2021, doi: 10.1117/12.2588316.

[41] S. Jose and R. Abraham, "Influence of chess and sudoku on cognitive abilities of secondary school students," *Issues and Ideas in Education*, vol. 7, no.1, pp. 27-34, Jul. 2019.

BIOGRAPHIES OF AUTHORS

Norizan Mat Diah is an Associate Professor at the School of Computing Sciences, College of Computing, Informatics, and Mathematics, Universiti Teknologi MARA, Shah Alam, Selangor. She holds a Ph.D. in information sciences from the National University of Malaysia. Her research focuses are on gamification, game engines, real-time feedback (algorithm), handwriting recognition (algorithm), and game theory. She can be contacted at email: norizan289@uitm.edu.my.

Syahirul Riza received her Bachelor degree in Information Systems (Hons.) Intelligent Systems Engineering at Universiti Teknologi MARA (UiTM), Shah Alam, Malaysia. He has work experience as an intern at TIME dotCom Berhad. He can be contacted at email: reza.53iky0@gmail.com

Suzana Ahmad is a Senior Lecturer at Department of Computing Sciences, Faculty Computer and Mathematical Sciences, Universiti Teknologi MARA, Shah Alam, Malaysia. She has graduated with Ph.D. in Information Sciences from Universiti Teknologi MARA. Her research focuses on Interactive Computing, Data Policies and Collaborative System. She can be contacted at email: suzana235@uitm.edu.my.

Norzilah Musa is a senior lecturer at the Department of Computer Science, University Teknologi MARA, Malaysia. She has been a faculty member since 2007. Norzilah holds a Bachelor of Science in Computer Science with Honors from University Teknologi Malaysia and Master of Science in Computer Science from Universiti Putra, Malaysia. In 2019, she received her doctorate in Information Technology and Quantitative Sciences from Universiti Teknologi MARA, Malaysia. Her primary research interests are in collaborative media, artificial intelligence, and data visualization. She can be contacted at email: norzi105@uitm.edu.my.

Shakirah Hashim is a senior lecturer at the School of Computing Sciences, College of Computing, Informatics, and Mathematics, Universiti Teknologi MARA, Shah Alam, Selangor. She graduated with Ph.D. in Electronic and Electrical Engineering from the University of Sheffield, UK. Her research interests are hardware encryption and machine learning. She can be contacted at email: shakirahhashim@uitm.edu.my.