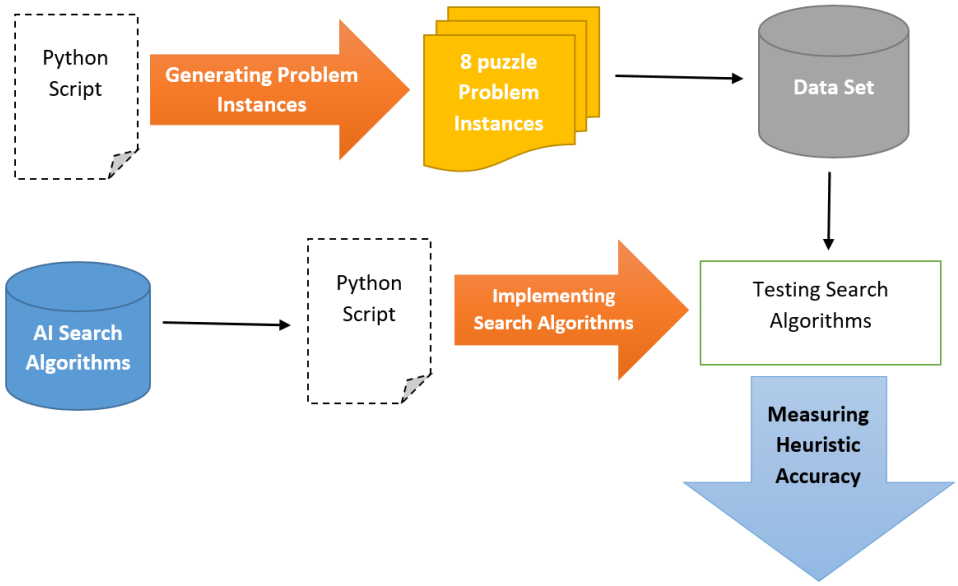
Measuring Heuristic Accuracy on the Performance of Search Algorithms in Solving 8-Puzzle Problems

S.D.T. Jananji and D.D.A. Gamini

Department of Computer Science, University of Sri Jayewardenepura, Nugegoda, Sri Lanka

Date Received: 23-12-2023 Date Accepted: 29-06-2024



Heuristic function	Percentage of reduction in average nodes generated	Percentage of reduction in average nodes expanded	Percentage of reduction in average EBF
Manhattan	99.12%	99.19%	21.11%
Euclidean	98.76%	98.86%	19.71%

Abstract

The goal of this paper is to examine the effect of heuristic accuracy on the performance of search algorithms in solving 8-puzzle problems. The 8-puzzle is a popular benchmark search problem in Artificial Intelligence, and numerous search strategies have been developed to solve it. This research evaluates the performance of search algorithms using different heuristics to determine the optimal level of heuristic accuracy for solving 8-puzzle problems. Several search algorithms, including uniform cost search and A* search, were implemented and tested using different heuristics. Manhattan and Euclidean distances were chosen as heuristics for A* algorithm. Heuristics play a crucial role in guiding these algorithms towards the solution, and their accuracy can greatly impact the performance of the search. Results revealed that reduction attained in both time and space complexity by A* search with Manhattan heuristic is over 99.1% for the average case, while the reduction in the effective branching factor is over 21.1%. These results from the experiments provide insights into the trade-off between heuristic accuracy and search efficiency, and contribute to a better understanding of the behavior of search algorithms in complex problems.

Keywords: *Heuristic search, A* algorithm, time complexity, space complexity, 8-puzzle problems, effective branching factor*

*Correspondence: gamini@sjp.ac.lk

1. Introduction

The field of computer science is currently focused on developing various artificial intelligence (AI) search algorithms. Search algorithms are divided into two broad categories; uninformed and heuristic. Uninformed search algorithms rely on available operators or actions and the structure of the search space to arrive at a solution, without utilizing any additional domain specific knowledge. On the other hand, heuristic search algorithms employ additional knowledge in terms of heuristic evaluation functions, to guide the search process. These heuristics are estimates of the cost of reaching the goal state and are used to prioritize the search and make more informed decisions about which nodes to explore next. This paper focuses on the application of two search algorithms, namely uniform cost search (an uninformed search algorithm) and A* search (a heuristic search algorithm), on the 8-puzzle problem domain.

Informed or heuristic search algorithms are generally more efficient compared to uninformed search as they utilize reliable knowledge to make decisions and prioritize search directions (Russell and Norvig, 2022). By employing different heuristic functions, informed search algorithms can improve the performance in terms of speed, accuracy, and efficiency. Hence, heuristic accuracy significantly influences search algorithm performance. The goal of our research is to analyze the impact of heuristic accuracy on solving 8-puzzle problems.

In the field of AI research, there have been extensive discussions regarding commonly used algorithms such as genetic algorithms, path finding algorithms, and problem reduction algorithms, to explore their impact on problem-solving (Kallem, 2012). Search algorithms aim to identify the most appropriate solution from a set of intended solutions, and different types of search algorithms, including uninformed and heuristic approaches, have their own advantages and limitations. Consequently, several researchers have conducted comparisons between these types of algorithms (Hidayat et al., 2021; Menon et al., 2018; Mishra and Siddalingaswamy, 2017; Pathak et al., 2018).

Uninformed search strategies such as depth-first search, breadth-first search (Igwe et al., 2013; Menon et al., 2018; Pathak et al., 2018), and uniform cost search (Pathak et al., 2018) have been commonly employed in previous researches. In addition, informed search algorithms, including A*, hill climbing, and steepest ascent hill climbing (Menon et al., 2018; Jain and Patel, 2023) have also been utilized. Notably, A* algorithm has shown superior performance in path finding and graph traversal problems according to comparative studies (Igwe et al., 2013; Menon et al., 2018; Pathak et al., 2018). Another research (Igwe et al., 2013) evaluated the performance of A* algorithm using genetic programming (GP) for solving the 8-puzzle problem, demonstrating its better performance.

Furthermore, numerous researchers have focused on assessing the performance of A* algorithm using different heuristic functions (Iordan, 2016, 2021). A comparison was made (Iordan, 2016, 2021) between several heuristic functions, such as Hamming, Manhattan, and Chebyshev heuristics, when applied to A* algorithm for solving puzzle games. Search algorithms are evaluated based on criteria such as time complexity, space complexity, optimality, and completeness (Pathak et al., 2018) in order to assess their performance. Time and space complexity have been widely used as performance criteria in comparative analyses of heuristics (Iordan, 2016, 2021). Various metrics have been employed to measure time complexity and space complexity, including running time (Iordan, 2016), average number of nodes expanded and explored (Hidayat et al., 2021), number of edges and vertices (Menon et al., 2018), as well as the number of nodes generated, number of nodes expanded, and effective branching factor (Iordan, 2016, 2021). Various extensions and expansions to 8-puzzle can be found in the literature, including diagonal moves, resulting in a bigger search space, and much larger 15-puzzle (Johnson and Story, 1879). An analysis of the former reveals that a significant fraction of solvable 8-puzzle positions is pathological with regard to a number of parameters (Piltaver et al., 2011).

Intelligent search mechanisms are employed in applications for solving a wide range of problems, from simple ones like the problem of block architecture (Rahim et al., 2006) to more complex ones like polyhedra puzzles (Iordan, 2021). Gaming is a prominent application area for

*Correspondence: gamini@sjp.ac.lk

intelligent search, including games like TicTacToe (Fathi et al., 2019), "Congklak" (a popular traditional Indonesian game) (Rahim et al., 2018), general path finding tasks, and Sudoku (Lina and Rumetna, 2021). These examples serve as considerations for researchers and game developers, emphasizing the importance of implementing heuristically enriched search algorithms to efficiently utilize memory and computational resources when solving the problems of similar nature (Hidayat et al., 2021).

2. Methodology

Python was selected as the main programming language for implementing and conducting experiments on the algorithms. The research comprises three main steps, involving multiple sub-tasks. Step one generates solvable random instances of the 8-puzzle problem. Step two implements and solves the problem instances using two search algorithms. Step three evaluates the performance of the search algorithms based on the results obtained.

2.1 Generating problem instances

The 8-puzzle problem is a benchmark problem in artificial intelligence that involves a 3x3 board with eight tiles and an empty square. The goal of this puzzle is to arrange the tiles in ascending order, with the empty square in the bottom-right corner as shown in Figure 1.

1	2	3
4	5	6
7	8	

Figure 1*:* Goal state

There are 9! (362,880) possible permutations of the tiles, and half of them are solvable (Reinefeld, 1993). By using the concept of "inversion,"(Ishan, 2022) it is possible to determine solvable problem instances.

Rule: “Odd permutation inversions of the puzzle are impossible to solve, all even permutations are solvable” (Bhasin and Singla, 2012).

In this research, a set of 100,000 random configurations were generated out of the total of 181,440 solvable instances using Python code.

2.2 Implementing search algorithms

Both informed and uninformed search algorithms were used in this research. The uninformed search algorithm chosen was the uniform cost search (UCS), while the informed search algorithm selected was the A* algorithm, which incorporated Manhattan and Euclidean heuristics.

Uniform cost search

UCS is a popular uninformed search algorithm in AI used for path finding problems. It explores all possible paths from the initial state and evaluates the total cost of each path until reaching a goal state. The algorithm operates similarly to breadth-first search, utilizing a priority queue and increasing the path cost by 1 for each state expansion. The path cost equation $g(n)$ represents the total cost of reaching node *n* from the starting node.

$$path\ cost = g(n) \qquad \text{----------(1)}$$

A algorithm*

A* is the most often used heuristic search algorithm for path finding and graph traversal problems. It combines uniform cost search with a heuristic evaluation function, which provides an estimated cost

*Correspondence: gamini@sjp.ac.lk

to reach a goal from the current state, to obtain optimal solutions efficiently. In the 8-puzzle problem, A* examines the initial state and generates successor states by moving one tile at a time. It computes the $f(n)$ value for each successor state using the path cost equation:

$$f(n) = g(n) + h(n) \quad \text{----------(2)}$$

where $g(n)$ is the cost from the initial state and $h(n)$ is the heuristic cost to the goal state. A* selects the state with the lowest $f(n)$ value and expands it until reaching the goal state. For the 8-puzzle, Manhattan and Euclidean heuristics are employed in this research, each providing a unique approach to estimating the cost to reach the goal.

Manhattan heuristic

The widely used admissible heuristic evaluation function recorded in the literature so far solving 8-puzzle problem is Manhattan distance heuristic (Iordan, 2016), which is calculated as follows:

$$h_M(S) = \sum ManhattanDistance(k), where\ k \in \{1,2,3,\dots,N\} \quad \text{----------(3)}$$

N represents the number of tiles and $ManhattanDistance(k)$ represents the distance of k number position (on vertical/horizontal axis) in state S state towards its position in the goal state which is calculated with the following relation (Iordan, 2016):

$$ManhattanDistance(k) = |x_k - x_{kg}| + |y_k - y_{kg}| \quad \text{----------(4)}$$

where (x_k, y_k) represents the coordinates of k number position in current state and (x_{kg}, y_{kg}) represents the coordinates of k number position in goal state.

Euclidean heuristic

Euclidean heuristic is a heuristic function for estimating the distance between a state and the goal state in the 8-puzzle problem. It uses the Pythagorean Theorem to calculate the straight-line distance between two points on a plane. The Euclidean heuristic function and distance can be calculated using below equations.

$$h_E(S) = \sum EuclideanDistance(k), where\ k \in \{1,2,3,\dots,N\} \quad \text{----------(5)}$$

$$EuclideanDistance(k) = \sqrt{(x_k - x_{kg})^2 + (y_k - y_{kg})^2} \quad \text{----------(6)}$$

where (x_k, y_k) represents the coordinates of k number position in current state, (x_{kg}, y_{kg}) represents the coordinates of k number position in goal state and N represents number of tiles.

2.3 Performance criteria

The performance of search algorithms for solving the 8-puzzle problem are evaluated based on a set of parameters chosen carefully. This research assessed the effectiveness of heuristic functions by considering two measures; space complexity and time complexity. Space complexity deals with the memory needed by an algorithm to carry out the search. In this study, it was measured by the number of nodes expanded. Time complexity concerns with the time consumed by an algorithm to complete the search, often represented in big-O notation. For the 8-puzzle problem, time complexity was approximated using the number of nodes generated, which depends on the branching factor and depth of the goal state. Furthermore, time complexity can be expressed as $O(b^d)$, where b is the effective branching factor and d is the depth of the goal state (Iordan, 2016). This indicates that the time

*Correspondence: gamini@sjp.ac.lk

complexity increases exponentially with the depth and effective branching factor, which was numerically calculated using equation (7).

$$N = b^* + (b^*)^2 + (b^*)^3 + \cdots + (b^*)^d \quad \text{----------(7)}$$

where N is the total number of nodes generated by the root, b^* is the effective branching factor, and d is the depth of the search tree.

3 Results

This research involved generating and solving 100,000 problem instances from the space of solvable 8-puzzle problems, which accounts for over 50% of all solvable instances. These instances were carefully selected to provide a comprehensive and representative sample. The analysis covered diverse problem instances and calculated the average depth of the problem space as 22. By assessing the performance of search algorithms and heuristics, the research provided a comprehensive evaluation of their effectiveness in solving the 8-puzzle problem.

3.1 Graphical comparison of nodes generated in search algorithms

To assess the time complexity of search algorithms, the number of nodes generated was measured and the average number of nodes for each depth was computed. The results clearly demonstrate that the use of heuristic functions resulted in a significant reduction in the number of nodes generated, as depicted in Figure 2.

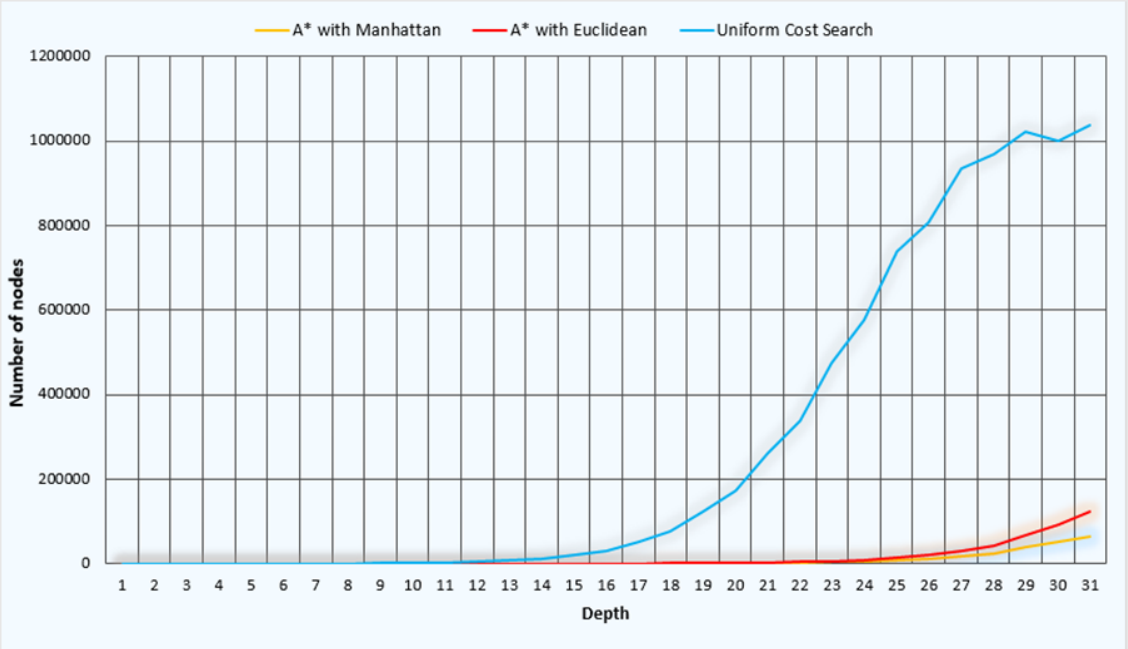


Figure 2: Average number of nodes generated

3.2 Graphical comparison of nodes expanded in search algorithms

To evaluate space complexity in the context of the 8-puzzle problem, the number of nodes expanded during the search process was measured. This metric provides insights into the efficiency of algorithms regarding their space requirements. Figure 3 graphically illustrates the average number of nodes expanded for UCS, A* with Manhattan heuristic, and A* with Euclidean heuristic. The figure clearly demonstrates that UCS expanded the highest number of nodes, whereas A* search algorithms

*Correspondence: gamini@sjp.ac.lk

with Manhattan and Euclidean heuristics expanded significantly fewer nodes. This outcome emphasizes the importance of selecting an appropriate heuristic function to minimize the number of nodes expanded and improve the space efficiency of search algorithms.

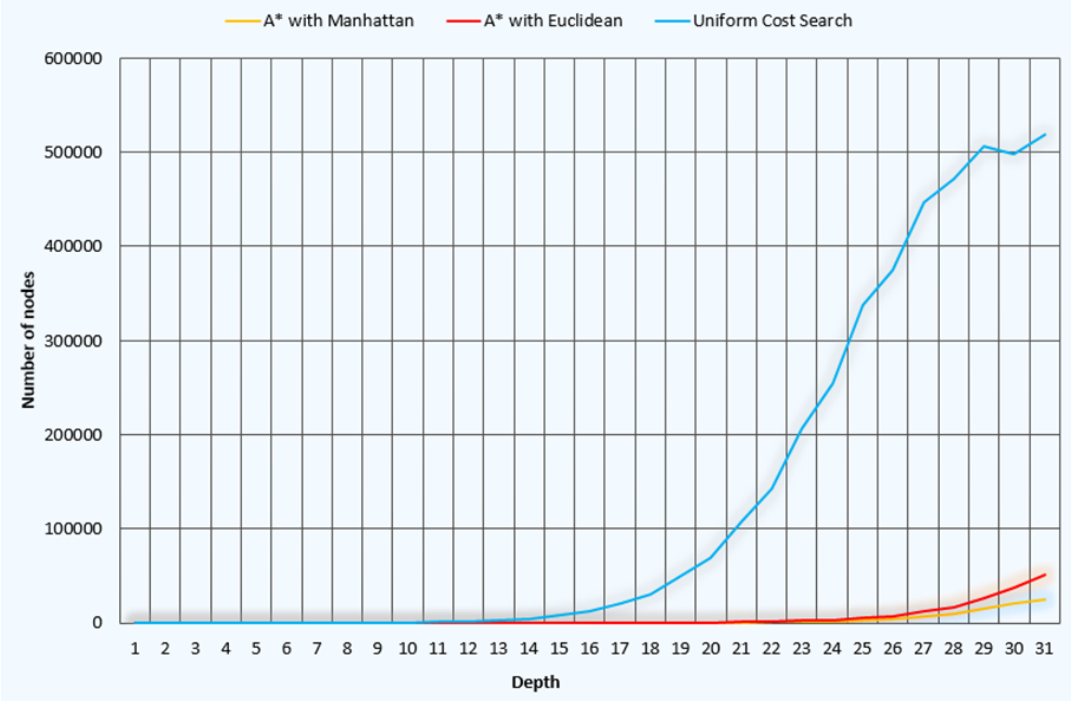


Figure 3: Average number of nodes expanded

3.3 Graphical comparison of the effective branching factor

The effective branching factor (EBF) is a valuable metric for measuring the time complexity of search algorithms, providing insights into their performance and time requirements. The average EBF was calculated numerically for each depth and search algorithm, and Figure 4 displays the results. The performance of Manhattan and Euclidean heuristics in improving algorithm efficiency was similar, as evidenced by their closely aligned lines in the graph. Additionally, as the depth increases, the difference in EBF between UCS and A* algorithms decreases, as depicted in the graph.

The percentage reduction of these metrics was calculated at each depth, providing a clear measure of how heuristic functions improve search algorithm performance. Furthermore, average reduction percentages for nodes generated, nodes expanded, and effective branching factor were determined. These results, including both Manhattan and Euclidean heuristics, are presented in Table 1 for problem instances with a depth of 22, which represents the average solution depth.

Table 1: Percentage reduction of heuristic functions

Heuristic function	Percentage of reduction in average nodes generated	Percentage of reduction in average nodes expanded	Percentage of reduction in average EBF
Manhattan	99.12%	99.19%	21.11%
Euclidean	98.76%	98.86%	19.71%

*Correspondence: gamini@sjp.ac.lk

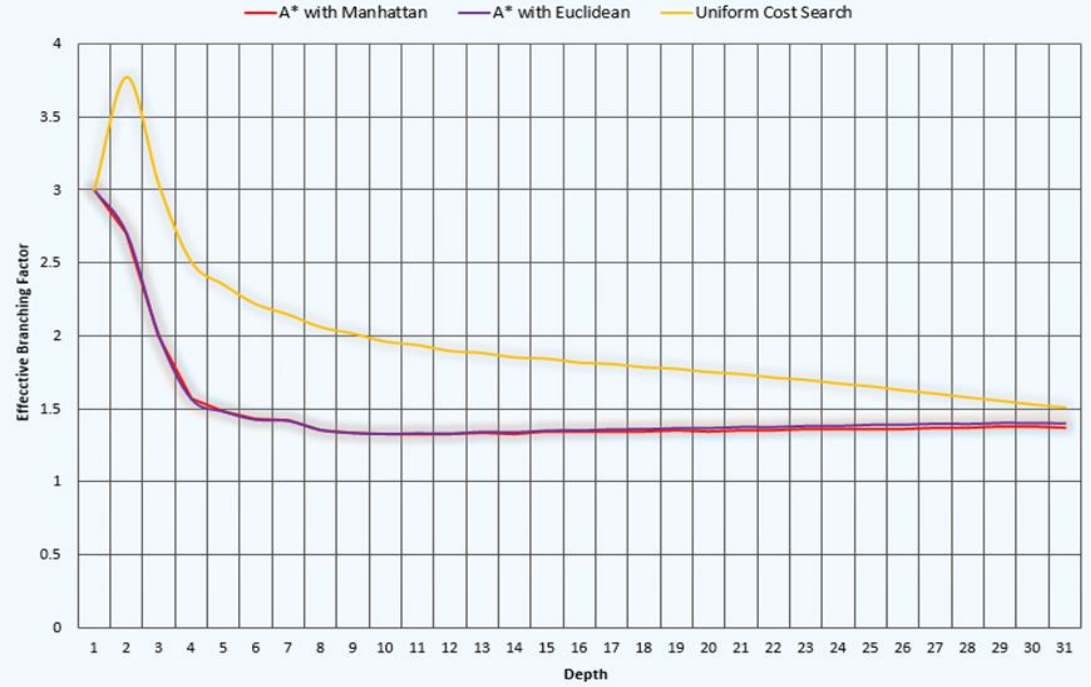


Figure 4: Average effective branching factor

4 Discussion

The findings of this study demonstrate the effectiveness of heuristic functions in enhancing the space and time complexity of search algorithms for solving 8-puzzle problems. The analysis of metrics such as the number of nodes generated, nodes expanded and the effective branching factor provides accurate acumens into the performance of each heuristic function. Furthermore, the average reduction percentages of nodes generated, nodes expanded, and effective branching factor for problem instances with a depth of 22, show that both Manhattan and Euclidean heuristics yield similar improvements in space and time complexity, with Manhattan heuristic performing slightly better. The reductions achieved were significant, exceeding 98.7% for both complexities and heuristics. Manhattan heuristic also resulted in a reduction of over 21% in the average effective branching factor.

The use of heuristic functions allows for a notable reduction in computation time. For example, when employing Manhattan heuristic as opposed to employing none, solving problems with an average depth of 22 becomes equivalent to solving problems with an average depth of 12 in terms of time requirements. Similarly, Euclidean heuristic offers substantial time reduction, making problems with an average depth of 22 equivalent to solving problems with a depth of 13.

The findings highlight the substantial enhancements brought by heuristic functions to search algorithm performance, particularly in terms of time and space complexity. Utilizing heuristics effectively reduces the number of nodes generated and expanded, as well as the effective branching factor. As a result, search times are accelerated, and memory usage becomes more efficient.

5 Conclusions

This research concludes that heuristic algorithms, specifically A* with Manhattan and Euclidean heuristics, outperform the UCS algorithm in terms of both space and time complexity when solving the 8-puzzle problem. A* with Manhattan and Euclidean heuristics effectively reduced the number of nodes generated and expanded by over 99.1% and 98.7% respectively on average. The Manhattan heuristic showed a slightly higher average reduction compared to the Euclidean heuristic. The research also highlighted the significant impact of the effective branching factor on time complexity, with an average reduction of over 21% for the Manhattan heuristic. Furthermore, this study concluded that

*Correspondence: gamini@sjp.ac.lk

minimizing the effective branching factor is crucial for improving the time complexity of search algorithms.

Overall, this study provides valuable insights into the influence of heuristic accuracy on the performance of search algorithms for solving the 8-puzzle problem, benefiting various fields such as artificial intelligence, game theory, and operations research.

References

Bhasin, H., Singla, N., 2012. Genetic based Algorithm for N – Puzzle Problem. International Journal of Computer Applications. 51(22).

Fathi, A., Youssef, B., Azmy, B., Takla, J., Fawzy K., El-Behaidy, W., 2019. AI Conceptual Gaming for Game Developers. iKNiTO JS Journal Management System. 1, 26-32. doi:10.21608/fcihib.2019.107522.

Hidayat, W., Susanti, F., Wijaya, D.R., 2021. A Comparative Study of Informed and Uninformed Search Algorithm to Solve Eight-Puzzle Problem. Journal of Computer Science. 17(11), 1147-1156. doi: 10.3844/jcssp.2021.1147.1156.

Igwe, K., Pillay, N., Rae, C., 2013. Solving the 8-Puzzle Problem Using Genetic Programming, SAICSIT '13: Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference, pp 64-67.

Iordan, E., 2016. A Comparative Study of Three Heuristic Functions used to Solve the 8-Puzzle. British Journal of Mathematics & Computer Science. 16, 1-18. doi:10.9734/BJMCS/2016/24467.

Iordan, E., 2021. Analytical Study of the A* Heuristic Search Algorithm used to Solve Efficiently a Puzzle Game. Advanced Aspects of Engineering Research. 9, 19-28. doi:10.9734/bpi/aaer/v9/7492D.

Ishan, 2022. How to check if an instance of 8 puzzle is solvable. www.geeksforgeeks.org/check-instance-8-puzzle-solvable/. Accessed 15 November 2023.

Jain, R., Patel, M., 2023. Investigating the Impact of Different Search Strategies (Breadth First, Depth First, A*, Best First, Iterative Deepening, Hill Climbing) on 8-Puzzle Problem Solving - A Case Study. International Journal of Scientific Development and Research (IJSDR). 8, 633-641.

Johnson, W., Story, W.E., 1879. Notes on the 15 Puzzle. American Journal of Mathematics. 2(4), 397-404.

Kallem, S.R., 2012. Artificial Intelligence Algorithms. IOSR Journal of Computer Engineering (IOSRJCE). 6, 01-08.

Lina, T.N., Rumetna, M.S., 2021. Comparison Analysis of Breadth First Search and Depth Limited Search Algorithms in Sudoku Game. Bulletin of Computer Science and Electrical Engineering. 2(2), 74-83. doi:10.25008/bcsee.v2i2.1146.

Menon, V., Amali, G.B., Gopichand, G., Santhi, H., Gayatri, P., 2018. Performance Analysis of Various Uninformed and Informed Search Strategies on 8 Puzzle Problems - A Case Study. World Wide Journal of Multidisciplinary Research and Development. 4(12), 96 -99.

Mishra, A.K., Siddalingaswamy, P.C., 2017. Analysis of Tree based Search Techniques for Solving 8-Puzzle Problem, 2017 Innovations in Power and Advanced Computing Technologies (i-PACT), Vellore, India, pp 1-5.

Pathak, M.J., Patel, R.L., Rami, S.P., 2018. Comparative Analysis of Search Algorithms. International Journal of Computer Applications. 179(50), 40-43.

Piltaver, R., Luštrek, M., Gams, M., 2011. The Pathology of Heuristic Search in the 8-puzzle. Journal of Experimental & Theoretical Artificial Intelligence, 24(1), 65–94. https://doi.org/10.1080/0952813X.2010.545997.

Rahim, R., Abdullah, D., Simarmata, J., Pranolo, A., Ahmar, A.S., Hidayat, R., Napitupulu, D., Nurdiyanto, H., Febriadi B., Zamzami, Z., 2006. Block Architecture Problem with Depth First Search Solution and its Application. In Journal of Physics: Conference Series. doi.org/10.1088/1742-6596/954/1/012006.

*Correspondence: gamini@sjp.ac.lk

Rahim, R., Kurniasih, N., Hasibuan, A., Andriany, L., Najmurrokhman, A., Supriyanto, S., Wardayani, W., Hidayat, R., Lubis, D.S.W., Napitupulu, D., Nugraha, A.T., Siburian, H.K., Abdullah, D., 2018. Congklak, A Traditional Game Solution Approach with Breadth First Search, In MATEC Web of Conferences, 197. doi.org/ 10.1051/ matecconf/ 201819703007.

Reinefeld, 1993. Complete Solution of the Eight-Puzzle and the Benefit of Node Ordering in IDA*, International Joint Conference on Artificial Intelligence, pp 248-253.

Russell, S.J., Norvig, P., 2022. Artificial Intelligence: A Modern Approach, fourth ed, Pearson.

*Correspondence: gamini@sjp.ac.lk