Building of Informatics, Technology and Science (BITS)
Volume 5, No 4, March 2024 Page: 659−671
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v5i4.4622

Algorithmic Advancements in Heuristic Search for Enhanced Sudoku Puzzle Solving Across Difficulty Levels

Moch Deny Pratama*, Rifqi Abdillah, Darlis Herumurti, Shintami Chusnul Hidayati

Faculty of Intelligent Electrical and Informatics Technology, Department of Informatics, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia
Email: [1,*]mr.denypr@gmail.com, [2]rifqi.bring@gmail.com, [3]darlis@if.its.ac.id, [4]shintami@its.ac.id
Correspondence Author Email: mr.denypr@gmail.com
Submitted: **27/11/2023**; Accepted: **30/03/2024**; Published**: 30/03/2024**

Abstract−Computer technology, particularly artificial intelligence, has found diverse applications in the rapidly evolving era of the industrial revolution, notably in gaming, delving into artificial intelligence and explicitly applying game-solving techniques to Sudoku puzzles. Sudoku, a popular game requiring logical precision, serves as an ideal platform for exploring algorithms such as depth-first search, breadth-first search, and heuristic search. This research identifies memory-intensive demands in breadth-first search and the potential issue of infinite traversal in depth-first search. To address these challenges, the study proposes implementing the heuristic search algorithm, which prioritizes promising paths based on estimations of proximity to the goal state made by a heuristic function. The primary objective is to enhance Sudoku puzzle-solving by comparing the performance of the heuristic search algorithm with traditional breadth-first and depth-first search methods, with a particular focus on improving efficiency and reducing memory usage, including time and steps. The results indicate that the heuristic search algorithm outperforms traditional methods, demonstrating faster completion times and reduced memory requirements, thereby contributing to the advancement of Sudoku-solving algorithms. The study evaluates their performance across different difficulty levels, utilizing data from sudoku.com and extremesudoku.info. Notably, the heuristic search algorithm emerges as a superior method, outperforming other algorithms in terms of completion steps and time efficiency. The implementation and analysis involved three types of Sudoku puzzle-solving methods, revealing that the heuristic search algorithm significantly outperforms other algorithms, optimizing its performance in solving Sudoku puzzles. The average time required to complete Sudoku puzzles from data sourced from Sudoku.com was 0.02, 0.05, and 0.61 seconds for each level, respectively. In contrast, according to extremesudoku.info, it took 0.31 seconds for the highest difficulty level. Furthermore, the average total steps needed on sudoku.com ranged from 43 to 1201 steps for each level, spanning from easy to hard. On extremesudoku.info, 509 steps were required for the highest difficulty level. These results affirm the reliability of heuristic search, consistently demonstrating encouraging outcomes and outperforming other algorithms across diverse conditions. This strategic selection facilitates a comprehensive analysis of Sudoku problem-solving algorithms, allowing for the exploration of algorithmic performance and providing a comprehensive range of Sudoku puzzles, thereby ensuring the study's robustness and validity.

Keywords: Artificial Intelligence Applications; Sudoku Puzzle Solving; Heuristic Search Algorithm; Algorithm Performance Evaluation; Game-solving Techniques

1. INTRODUCTION

In the current era of the Fourth Industrial Revolution, characterized by significant technological advancements, sophisticated technologies find widespread application across various domains, with gaming being a prominent area of focus. Computers play a crucial role in driving innovation in information technology during this period [1]. The utilization of computer technology transcends multiple sectors, including industry, tourism, business, health, education, psychology, and gaming, among others [2]. Within the realm of computer science, Artificial Intelligence (AI) emerges as a key discipline, delving into the exploration of computational systems capable of emulating human cognitive processes to solve complex problems based on predefined commands and instructions [3]. Artificial intelligence finds practical implementation in diverse contexts, notably in the realm of gaming through systems like Game Solver. This technology harnesses the power of AI algorithms to autonomously solve various types of games, leveraging computer technology to achieve optimal outcomes. Sudoku, a popular puzzle game, serves as a notable example of how artificial intelligence is employed in gaming scenarios [2].

In recent times, there has been a notable surge in the popularity of games designed to enhance cognitive abilities and logical precision. Among these, Sudoku emerges as a prominent example, offering players a puzzle-based gaming experience characterized by strategic placement of numbers within designated rows and columns [4]. Sudoku represents a combinatorial puzzle, typically presented in a 9x9 grid format, subdivided into 3x3 subgrids. Variations of the game may feature alternative grid sizes, such as 6x6 grids with 2x3 subgrids. The primary objective remains consistent: to fill the grid with numbers from 1 to 9 without repetition in any row, column, or subgrid [5]. Sudoku puzzles enjoy widespread popularity across Arab and European countries, captivating players with their intricate challenges and logical gameplay mechanics. Each Sudoku puzzle consists of a 9x9 grid, comprising a total of 81 cells. The game typically commences by displaying a selection of random numbers from 1 to 9, providing initial clues to the player. Subsequently, players must strategically deduce and input the remaining numbers in the empty cells, ensuring that each row, column, and 3x3 block contains a complete set of numbers 1 to 9 without any repetitions, thereby avoiding redundancy [6]. In the context of solving Sudoku puzzles, the implementation of artificial intelligence has emerged as an effective strategy, leveraging a diverse range of algorithmic approaches to achieve optimal outcomes. By harnessing sophisticated AI algorithms, Sudoku solvers can adaptively analyze puzzle configurations and employ problem-solving techniques to efficiently identify and fill empty cells. This utilization,

Building of Informatics, Technology and Science (BITS)
Volume 5, No 4, March 2024 Page: 659−671
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v5i4.4622

significantly enhances the adaptability and problem-solving capabilities of gaming systems, particularly in puzzles like Sudoku.

Several algorithms have been developed to tackle Sudoku puzzle games, among which Breadth-First Search (BFS), Depth-First Search (DFS), and Heuristic Search stand out as prominent examples [7]. When applied to puzzle games, the breadth-first search algorithm facilitates the search for the best solution regarding completion steps and the ability to address various conditions based on the puzzle's requirements. Implementing the Breadth-First Search algorithm in a puzzle game offers a faster solution without requiring intense deliberation. This algorithm solves problems and provides a step-by-step solution for optimizing game-solving performance, even in processing time. In previous related research, the Breadth-First Search (BFS) algorithm has been utilized to address these problems. The application of the BFS algorithm in solving puzzle games offers users the advantage of obtaining optimal solutions in terms of completion steps and the ability to address diverse conditions inherent to the puzzle. However, it is important to note that this approach may necessitate an extended time frame due to the potentially large number of steps required to explore all possible solutions [8]. In the Depth-First Search algorithm, each iteration involves selecting the cell with the most miniature set of values and then choosing the value with the first numeric sequence available for the selected cells. This method maximizes the probability of making correct guesses through deep search and is referred to as the minimum remaining value heuristic [9]. Iterative deep search has long been considered the best algorithm for many games. Depth-first search is more straightforward for completing games because it does not require exploring a large tree before searching for the next depth [10]. In previous research conducted by [11], the implementation of backtracking with Depth-First Search is proposed to facilitate the shortest route back to the start position. This approach is necessitated by constraints such as limited energy and processing resources inherent in the problem-solving environment. Research conducted by [12] demonstrates that heuristic algorithms achieve good performance efficiency in eight type of puzzle. These algorithms are compared with others such as breadth-first search, depth-first search, and iterative deepening search. The results indicate that the heuristic algorithm can solve the problem more efficiently.

The principal limitation of the breadth-first search lies in its demanding memory requirements [13]. To execute BFS, each level of the search tree must be stored in memory to generate subsequent levels. As a result, the memory usage increases proportionally with the number of stored nodes, leading to a substantial space complexity. This reliance on memory makes BFS impractical for large-scale problems, as it quickly exhausts available memory resources on typical computers, often within a matter of minutes. On the other hand, the Depth-First Search algorithm faces a different limitation: the potential to get stuck in an infinite loop, persistently traversing the left-most path of the search tree indefinitely [14]. This indication occurs when DFS encounters a tree with branching factors that lead to infinite paths. To address this issue, a common solution involves setting a cutoff depth for the search, thereby restricting the exploration to a finite depth level. However, determining the ideal depth cutoff presents a challenge, as this value is rarely known in advance and may vary depending on the specific problem instance. These limitations underscore the importance of considering algorithmic trade-offs and selecting the most appropriate search strategy based on the characteristics of the problem at hand. Additionally, researchers continue to explore optimization techniques and alternative algorithms to mitigate these limitations and improve the efficiency and effectiveness of search algorithms in solving complex problems.

In the realm of problem-solving algorithms, two fundamental approaches are breadth-first search (BFS) and depth-first search (DFS), both falling under the category of uninformed search algorithms. These algorithms operate by systematically exploring the search space without incorporating any additional information about the problem structure. Consequently, they prioritize exploration based solely on the current state without considering the potential optimality of the chosen path [12]. In contrast, heuristic search algorithms, a subtype of informed search, introduce the concept of heuristic functions to guide the search process. These functions provide estimates of how close a given state is to the goal state, enabling the algorithm to make informed decisions about which paths to explore. By incorporating heuristic information, informed search algorithms can navigate the search space more intelligently, avoiding unnecessary exploration and prioritizing paths that are likely to lead to the goal state. In the context of solving Sudoku puzzles, the application of heuristic search algorithms holds promise for enhancing performance and efficiency compared to traditional uninformed search methods [15]. By leveraging heuristic functions to evaluate the proximity of each state to the goal state, heuristic search algorithms can make more informed decisions about which cells to fill next, potentially reducing the search space and accelerating the puzzle-solving process. The proposed research aims to investigate the efficacy of implementing the heuristic search algorithm [9]. By conducting a comprehensive analysis of performance and completion time across various difficulty levels, the study seeks to demonstrate the superiority of heuristic search in optimizing puzzle-solving efficiency.

The urgency to conduct research in this area comes from the increasing demand for efficient problem-solving algorithms, especially in gaming applications. As technology advances rapidly, there is a need to explore innovative methods to improve puzzle-solving abilities, such as those used in Sudoku. To address this need, the proposed research utilizes heuristic search algorithms to optimize Sudoku puzzle-solving efficiency. By leveraging heuristic functions to guide the search process, these algorithms offer a promising approach to overcoming the limitations. A significant research gap exists in the efficacy of heuristic search algorithms compared to traditional search methods in the context of Sudoku puzzle-solving. While previous studies have demonstrated the effectiveness of heuristic algorithms in various problem-solving tasks, their specific application to Sudoku puzzles remains relatively unexplored. The

Building of Informatics, Technology and Science (BITS)
Volume 5, No 4, March 2024 Page: 659−671
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v5i4.4622

proposed research aims to bridge this gap by conducting a comprehensive analysis of performance and completion time across different difficulty levels of Sudoku puzzles. The research methodology involves utilizing two distinct datasets, each containing Sudoku puzzles of varying difficulty levels, to ensure the robustness and validity of the findings. Overall, the research endeavors to contribute to the advancement of Sudoku-solving algorithms by showcasing the benefits of heuristic search over uninformed search methods. Through empirical analysis and evaluation, the study aims to provide valuable insights into the effectiveness of heuristic search in improving puzzle-solving performance across different difficulty levels.

2. RESEARCH METHODOLOGY

2.1 Data Collection

Figure 1., shows the research methodology employed in this study follows a structured process flow, encompassing several distinct stages to ensure the systematic investigation and analysis of the research problem. The first step is collecting data, involves acquiring the necessary Sudoku puzzle datasets from reputable sources, such as sudoku.com and extremesudoku.info. These datasets encompass puzzles of varying difficulty levels, providing a diverse range of challenges for analysis. The data collection process explores the origins of the dataset essential to this study, with a primary focus on the intricacies of the Sudoku puzzle game. Within this dataset, numerical values ranging from 1 to 9 are artfully distributed across the cells of a 9x9 Sudoku grid, totaling 81 cells. A thorough examination becomes imperative as it delves into the intricate relationships between numbers within rows, columns, and smaller 3x3 sub-grids. This scrutiny ensures strict adherence to Sudoku rules, preventing the occurrence of repeated values or redundancy. The Sudoku problems are methodically curated from two distinct data platforms, each presenting diverse difficulty levels. The data obtained from extremesudoku.info [16] categorizes difficulty into five levels: "Evil," "Excessive," "Egregious," "Excruciating," and the highest tier labeled "Extreme", as shown in Figure 2. In contrast, the sudoku.com [17] data classifies difficulty into three tiers: "easy," "medium," and "hard", as shown in Figure 3. This strategic selection of diverse datasets, each featuring varying difficulty levels, enriches the depth of the study. It facilitates a comprehensive analysis of Sudoku problem-solving algorithms, allowing for exploring algorithmic performance across various challenges. The second step is selected algorithmic methods, including breadth-first search, depth-first search, and heuristic search, are implemented using appropriate programming languages, such as Python. This stage involves coding the algorithms to enable their application in solving Sudoku puzzles. The algorithms are then applied to solve the Sudoku puzzles obtained from the datasets. This experimentation phase involves systematically solving each puzzle using the implemented algorithms while recording relevant performance metrics, such as completion time and number of steps. Subsequently, the collected data from the experimentation phase are analyzed and evaluated to assess the performance of each algorithm. This involves comparing the efficiency and effectiveness of the algorithms across different difficulty levels and identifying patterns or trends in their performance. The last step is results of the performance evaluation are interpreted to draw meaningful conclusions regarding the efficacy of the heuristic search algorithm compared to breadth-first search and depth-first search. Any notable findings or insights are highlighted and discussed in detail. The findings are discussed in the context of the research objectives and hypotheses. The strengths, limitations, and implications of the study are addressed, and recommendations for future research are proposed.

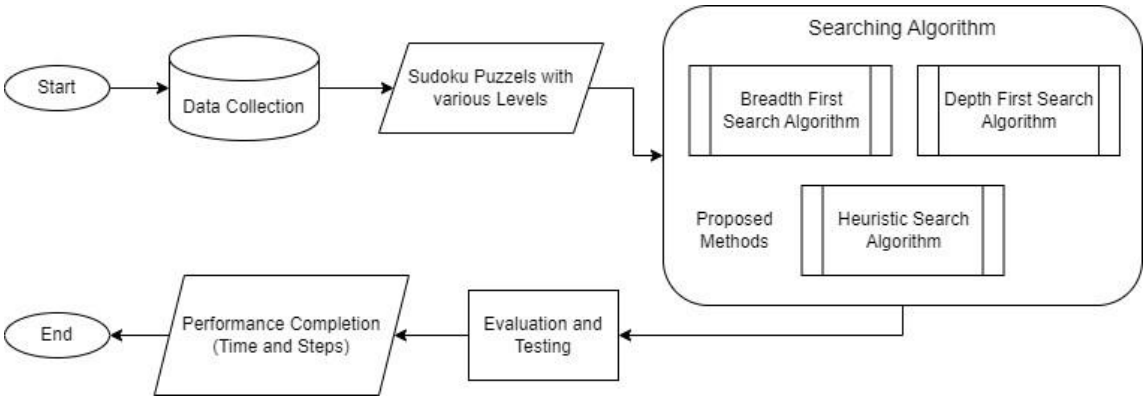

Figure 1. Research Methods Stage

Sudoku stands out as one of the most popular puzzle games. In the standard challenge, a 9×9 grid is divided into 3×3 cells, ensuring that each row, column, and subgrid contains all the numbers from 1 to 9 [18]. Claimed as one of the most exciting and addictive puzzles for people of all ages, Sudoku relies on number positioning and logic. The puzzles in the Sudoku game typically start unsolved or partially solved [19]. Traditional types of puzzles are frequently digitized, with many of these games transitioning to digital platforms without altering their core mechanics. Notable examples include Sudoku and crossword puzzles. Digitization enhances the accessibility of traditional puzzle games,

Building of Informatics, Technology and Science (BITS)
Volume 5, No 4, March 2024 Page: 659−671
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v5i4.4622

allowing them to be enjoyed anytime and anywhere using a mobile phone or computer electronic device [3]. Based on the following illustration regarding the Sudoku Puzzle, described in Figure 4., it is a sudoku puzzle problem, and Figure 5. is the solution to a sudoku problem in Figure 4. Based on the following illustration regarding the Sudoku Puzzle, described in Figure 3., it is a sudoku puzzle problem, and Figure 5., is the solution to a sudoku problem. Figures 4 and 5 visually represent a Sudoku puzzle problem and its corresponding solution, respectively, providing clear examples of the game's format and the goal of solving it.

Figure 2., illustrates the graphical representation of the difficulty levels categorized by extremesudoku.info, offering a visual understanding of the different tiers of Sudoku puzzles available in the dataset.

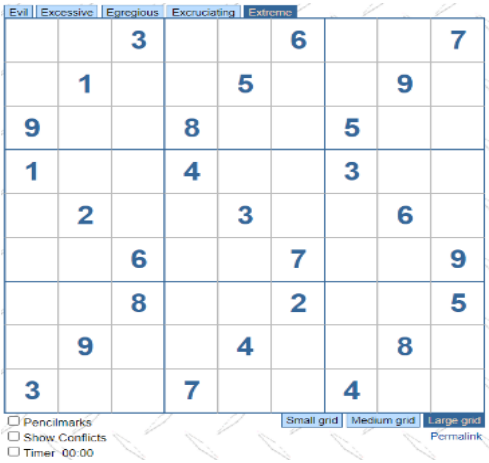

Figure 2. Sudoku Puzzle Level extremesudoku.info [16]

Figure 3., illustrates the classification of Sudoku puzzle difficulty levels as categorized by sudoku.com, helps in understanding the different levels of challenges present in the dataset. This strategic selection of diverse datasets, each featuring varying difficulty levels, enriches the depth of the study. It facilitates a comprehensive analysis of Sudoku problem-solving algorithms, allowing for the exploration of algorithmic performance across various challenges.

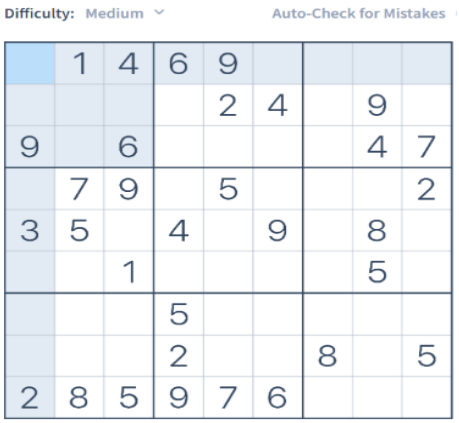

Figure 3. Sudoku Puzzle Level sudoku.com [17]

Figure 4., illustrates a 9x9 Sudoku puzzle, portraying a square grid divided into nine 3x3 subgrids. The puzzle contains a selection of numbers strategically placed within its corners, serving as initial clues provided to the player at the beginning of the puzzle-solving process. These initial numbers provide a foundation for the player to start deducing the correct placement of the remaining numbers. The configuration of the puzzle imparts a puzzling appearance, challenging the player to engage in logical reasoning and strategic thinking to complete it. The objective of the Sudoku puzzle is to fill in the remaining squares with numbers ranging from 1 to 9. However, the placement of each number must adhere to specific rules: each row, column, and 3x3 subgrid must accommodate all digits from 1 to 9 without repetition. While Figure 3 represents an incomplete puzzle with 24 squares still unoccupied, the provided clues are substantial for deducing the correct numbers and progressing towards solving the puzzle. The player must carefully analyze the existing numbers and consider the constraints imposed by the Sudoku rules to determine the correct placement for each remaining digit. As the player progresses through the puzzle, logical deductions and strategic reasoning are employed to fill in the remaining squares, gradually revealing the complete solution. Figure 3 represents a snapshot of the Sudoku puzzle-solving process, highlighting the challenge and satisfaction of deducing the correct numbers to ultimately solve the puzzle.

Building of Informatics, Technology and Science (BITS)
Volume 5, No 4, March 2024 Page: 659−671
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v5i4.4622

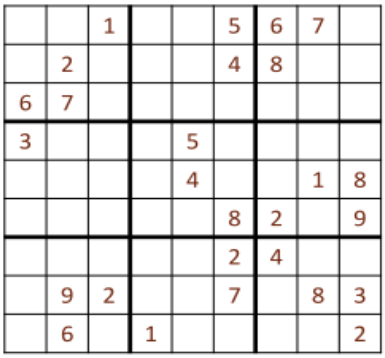

Figure 4. Example of Board in a Sudoku Puzzle Game [18]

Figure 5., illustrates a partially completed 9x9 Sudoku puzzle solution, presenting a grid divided into nine 3x3 subgrids. Each subgrid contains some numbers already filled in, while other cells remain blank, indicating numbers yet to be determined. The goal of the Sudoku puzzle is to fill in the empty cells with numbers from 1 to 9, ensuring that each row, column, and 3x3 subgrid contains all digits exactly once. In this illustration, certain numbers are already placed in specific cells, providing clues for completing the puzzle. However, there are still vacant cells that require the solver's attention to determine the correct numbers to fill them. The placement of each number must adhere to the Sudoku rules, which dictate that no row, column, or subgrid can contain duplicate numbers. The puzzle solver must use logic and deduction to determine the correct numbers for each empty cell, considering the numbers already present in the grid and ensuring that no conflicts arise. As the puzzle progresses, filling in one cell correctly may provide clues that help determine the numbers for neighboring cells, gradually leading to the completion of the entire puzzle. Overall, Figure 5., represents a typical stage in solving a Sudoku puzzle, where some progress has been made, but further analysis and decision-making are necessary to reach the final solution.

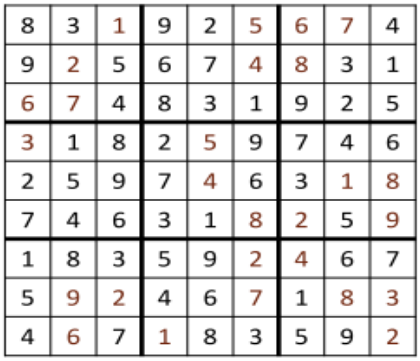

Figure 5. Example of a Solution in a Sudoku Puzzle Game [18]

2.2 Approach of Searching Algorithms

Breadth-first Search (BFS) is an algorithm for searching a tree data structure to find a node that satisfies a given property. This algorithm operates by exploring all nodes at a level, and when no nodes are found, the exploration proceeds to the next level [20]. A simplified version is employed when the breadth-first search algorithm is applied to solving a Sudoku puzzle. In this algorithmic approach, the leftmost and topmost nodes are root nodes, guiding the initial exploration. In contrast, the rightmost and bottommost nodes are designated finish nodes. Furthermore, an additional consideration is given to the diagonal plane, treated as a distinct level that warrants exploration before proceeding to subsequent levels. This strategic structuring of the BFS algorithm enhances its efficiency in systematically traversing the Sudoku puzzle, contributing to an optimized and effective solving process.

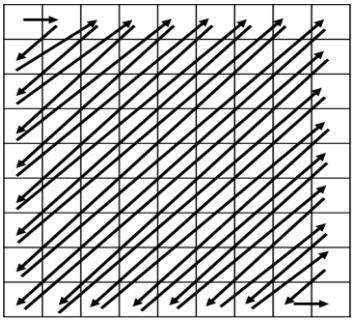

Figure 6. Illustrate Breadth-First Search to Solve Sudoku Puzzle

Building of Informatics, Technology and Science (BITS)
Volume 5, No 4, March 2024 Page: 659−671
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v5i4.4622

This approach uses breadth-first search algorithm method to explore sideways based on the existing layers. The search is carried out sideways to find empty Sudoku values based on existing values (hints). This algorithm is a graph traversal algorithm that explores a graph level by level. It starts at the root (selecting an arbitrary node as the root in the case of a graph). It explores the neighbour nodes before moving to the next level of neighbours. It calculates the performance of the breadth-first search method at each different level by observing the number of steps and the most optimal completion time. An illustration of the breadth-first search for the search algorithm is shown in Figure 6.

Depth-First Search (DFS) is a search algorithm for traversing or searching tree or graph data structures. Deep-first search employs the backtracking concept, meaning the algorithm starts exploring nodes by moving forward as far as possible, if a dead end is encountered, the search retraces to the root of the last branch traversed by the previous path [11]. The deep-first search algorithm is a versatile tool with applications in various domains. Its utility extends to tasks such as topological sorting, scheduling problem resolution, graph cycle detection, and tackling puzzles with a unique solution, exemplified by mazes or Sudoku puzzles. Additionally, deep-first search finds network analysis relevant, particularly in examining whether a graph is bipartite. When specifically applied to the challenge of solving Sudoku puzzles, the deep-first search method undergoes a streamlined adaptation known as straightforward deep-first search. This approach is characterized by its simplicity, opting for a lack of constraint propagation techniques. Notably, the order of cells defining the search tree in this method remains fixed, contributing to the method's clarity and efficiency in navigating the intricacies of Sudoku puzzles.

This approach uses the deep-first search algorithm to explore the Sudoku completion technique in depth. The search is carried out with a downward or deep search direction to solve a Sudoku puzzle by searching for a Sudoku solution. This graph traversal algorithm explores a graph by going as deep as possible along each branch before backtracking. It starts at the root node (or an arbitrary node) and explores as far as possible along each branch before backtracking. Calculate the deep-first search method's performance at each level, such as easy, medium, hard, extreme, and evil, by observing the number of steps and the most optimal completion time. An illustration of the depth-first search for the search algorithm is shown in Figure 7.

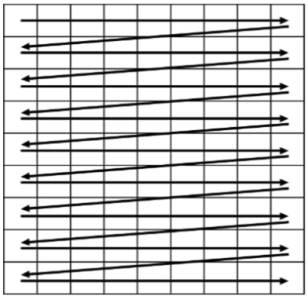

Figure 7. Illustrate Depth First Search to Solve Sudoku Puzzle [21]

2.3 Approach of Proposed Heuritic Searching Algorithm

Heuristics Search is a technique that enhances efficiency in the search process but at the expense of completeness. Heuristic functions evaluate individual problem states, determining their potential to contribute to the desired solution [22]. The heuristic search method for solving a Sudoku puzzle is divided into three parts. The first step involves candidate reduction, achieved by identifying cells with the potential to be filled and eliminating cells with no possibilities. The second step is to identify uniqueness in each cell, following a simple rule. If a candidate cell has only one unique possibility in a column, that value is assigned to the cell. The third step involves finding hidden pairs within each column. The two-unit cells belonging to a pair are then assigned to one of the members, allowing the elimination of other candidates for these two cells [23]. Heuristic search is a part of informed search algorithms used to find the best solution to a problem through a search algorithm [24]. The heuristic function and its associated values are employed to strive for the optimal solution to the problem, targeting the goal by expanding the node closest to it. In such cases, heuristics are critical in guiding the search in potentially excellent directions, thereby reducing search effort [25]. Heuristic methods like genetic algorithms (GA), particle swarm optimization (PSO), and others are frequently employed to optimize objective functions. These methods have demonstrated their efficacy in addressing various engineering problems, showcasing their versatility and proficiency across diverse domains [26]. Their application's common objective is to minimize the number of moves required to achieve a specified goal. To enhance this optimization process, a widely adopted cost-to-go heuristic is the Manhattan distance, which is ideal when navigating environments without barriers. This heuristic serves as a guiding metric, contributing to the efficiency of the methods by providing a valuable measure for evaluating the progress toward the goal [27].

This approach uses heuristic search employing the greedy search algorithm, which navigates the Sudoku puzzle landscape by leveraging distance heuristic principles. The algorithm strategically selects the most promising paths by prioritizing efficiency and immediate gains, aiming to identify optimal solutions swiftly. Operating with a cost-effective direction, it meticulously examines the Sudoku puzzle, discerning vacant positions by leveraging existing clues. This algorithm is a heuristic search strategy that makes locally optimal choices at each problem stage to find a

Building of Informatics, Technology and Science (BITS)
Volume 5, No 4, March 2024 Page: 659−671
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v5i4.4622

global optimum. It operates by selecting the best option at each decision point without considering the overall consequences. In other words, it prioritizes immediate gains and does not backtrack to reconsider previous choices. The algorithm is handy when a simple, quick solution is acceptable, and it often works well for optimization problems where finding a perfect solution is impractical or time-consuming. The performance evaluation of the heuristic method encompasses multiple difficulty levels, such as easy, medium, hard, extreme, evil, and more. This assessment includes a comprehensive analysis of the required steps and the optimal completion time, providing insights into the algorithm's effectiveness across diverse Sudoku challenges.

2.4 Evaluation of Algorithmic Performance

In this performance evaluation section, we study how effective algorithmic methods perform in solving Sudoku puzzles, leveraging the Python programming language for assessment. We utilize two different types of Sudoku puzzles and apply them at different levels of difficulty, including easy, medium, hard, extreme challenges, and others. Through this rigorous evaluation process, we carefully compared and analyzed the number of steps taken and the time required to successfully solve each Sudoku puzzle using this algorithm. By closely examining these metrics at different levels of puzzle complexity, we gain valuable insights into the efficiency and effectiveness of the algorithm, explaining its performance characteristics and capabilities in various situations.

3. RESULT AND DISCUSSION

In this section, an experiment is conducted using breadth-first search, depth-first search, and heuristic methods to solve Sudoku puzzles. The experimental data were obtained from sudoku.com and extremesudoku.info. sudoku.com presents three difficulty levels used in this paper: 'easy,' 'medium,' and 'hard.' On extremesudoku.info, a website providing Sudoku puzzles, five difficulty levels are available, ranging from the easiest, labeled 'Excessive' to the most difficult, labeled 'Evil.' The test involves using two puzzles from each level, resulting in eight levels ranging from very easy to complex. Parameters for comparison include the number of steps required to complete each level and the time taken in the puzzle-solving process. The puzzles consist of 9x9 matrices with hints from the Sudoku puzzle, where zero values denote empty boxes. Figure 8., provides an illustrative example of a Sudoku puzzle, demonstrating how the grid is initially set up with some numbers filled in as hints. It represents a typical 9x9 Sudoku grid where players need to fill in the empty cells with numbers from 1 to 9 while adhering to specific rules.

```
Testing on invalid 9x9 grid...
Problem:
[6, 0, 0, 0, 0, 0, 0, 0, 3]
[0, 0, 7, 1, 0, 8, 9, 0, 0]
[0, 1, 0, 0, 3, 0, 0, 8, 0]
[0, 6, 0, 0, 5, 0, 0, 3, 0]
[0, 0, 2, 6, 0, 9, 8, 0, 0]
[0, 7, 0, 0, 8, 0, 0, 2, 0]
[0, 5, 0, 0, 7, 0, 0, 1, 0]
[0, 0, 1, 5, 0, 2, 3, 0, 0]
[2, 0, 0, 0, 0, 0, 0, 0, 4]
```

Figure 8. Example of implementation Sudoku Puzzle

Figure 9., illustrates the results of applying the breadth-first search (BFS) algorithm to solve a Sudoku puzzle are presented. It displays the completed Sudoku grid achieved through the BFS method, along with details such as the total time taken to solve the puzzle (1.38 seconds) and the total number of steps explored (7741).

```
Solving with BFS...
Found solution
[6, 9, 8, 7, 2, 5, 1, 4, 3]
[3, 2, 7, 1, 4, 8, 9, 6, 5]
[5, 1, 4, 9, 3, 6, 7, 8, 2]
[8, 6, 9, 2, 5, 7, 4, 3, 1]
[4, 3, 2, 6, 1, 9, 8, 5, 7]
[1, 7, 5, 4, 8, 3, 6, 2, 9]
[9, 5, 3, 8, 7, 4, 2, 1, 6]
[7, 4, 1, 5, 6, 2, 3, 9, 8]
[2, 8, 6, 3, 9, 1, 5, 7, 4]
Elapsed time: 1.3784444332122803 seconds
Number of step: 7741
```

Figure 9. The sudoku puzzle that was solved with the breadth-first search algorithm

Building of Informatics, Technology and Science (BITS)
Volume 5, No 4, March 2024 Page: 659−671
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v5i4.4622

Based on the experimental results in solving the Sudoku puzzle in Figure 8., the results of using the breadth-first search algorithm and depth-first search algorithm methods are shown in Figure 9., the results of the BFS method display a total time needed to complete a Sudoku puzzle of 1.38 seconds and a total of 7741 steps explored. Meanwhile, in Figure 10., the outcomes of utilizing the depth-first search (DFS) algorithm to solve a Sudoku puzzle. It exhibits the solved Sudoku grid obtained through the DFS approach, accompanied by information regarding the total time required for solving the puzzle (0.02 seconds) and the total number of steps explored (114).

```
Solving with DFS...
Found solution
[6, 9, 8, 7, 2, 5, 1, 4, 3]
[3, 2, 7, 1, 4, 8, 9, 6, 5]
[5, 1, 4, 9, 3, 6, 7, 8, 2]
[8, 6, 9, 2, 5, 7, 4, 3, 1]
[4, 3, 2, 6, 1, 9, 8, 5, 7]
[1, 7, 5, 4, 8, 3, 6, 2, 9]
[9, 5, 3, 8, 7, 4, 2, 1, 6]
[7, 4, 1, 5, 6, 2, 3, 9, 8]
[2, 8, 6, 3, 9, 1, 5, 7, 4]
Elapsed time: 0.016798973083496094 seconds
Number of step: 114
```

Figure 10. The sudoku puzzle that was solved with the depth-first search algorithm

3.1 Evaluation of the Breadth-First Search Algorithm Approach

Evaluation and comparative analysis of the performance of the breadth-first search algorithm method in solving Sudoku puzzles. This analysis involved two types of Sudoku problems at each level, sourced from sudoku.com and extremesudoku.info respectively. Table 1., presents a detailed examination of the performance outcomes for two Sudoku puzzles, Puzzle 1 and Puzzle 2, across varying difficulty levels: Easy, Medium, and Hard, from Sudoku.com. The metrics include the time taken for completion in seconds and the number of steps involved in the solving process. In Puzzle 1, the easy level was solved in 0.068 seconds with 379 steps, the medium level took 2.095 seconds with 13,657 steps, and the hard level required 8.353 seconds with 55,243 steps. In Puzzle 2, the Easy level exhibited a completion time of 0.019 seconds with 119 steps, the medium level took 0.579 seconds with 3,199 steps, and the Hard level showed a completion time of 2.459 seconds with 14,881 steps. This comprehensive analysis unveils the algorithmic efficiency across distinct difficulty levels and puzzles, shedding light on the intricate balance between time efficiency and the number of steps taken.

Table 1. Results from Sudoku.com with BFS Algorithm

	Puzzle 1		**Puzzle 2**	
	Time (s)	**Step**	**Time (s)**	**Step**
Easy	0.068172	379	0.018902	119
Medium	2.095683	13657	0.578852	3199
Hard	8.353435	55243	2.459471	14881

Table 2., presents a comprehensive analysis of the algorithmic performance in solving Sudoku puzzles across different difficulty levels, denoted as Levels 1 to 5, for Puzzle 1 and Puzzle 2 from extremesudoku.info. For Puzzle 1, Level 1 requires a solving time of 0.494 seconds and 2062 steps, with subsequent levels showcasing incremental solving times and step counts. The complexity increases notably in Level 5, where the puzzle demands 9.182 seconds and 62890 steps. Puzzle 2, however, displays a variation in solving times and steps, with Level 1 needing 0.14347 seconds and 1018 steps. While Level 2 and Level 3 follow a similar pattern, the difficulty peaks in Level 5, necessitating 4.576 seconds and 37120 steps. This analysis underscores the escalating computational demands associated with higher difficulty levels, providing valuable insights into the algorithmic efficiency and performance variations across distinct Sudoku.

Table 2. Results from Extremesudoku.info with BFS Algorithm

	Puzzle 1		**Puzzle 2**	
	Time (s)	**Step**	**Time (s)**	**Step**
Lv. 1	0.494374	2062	0.14347	1018
Lv. 2	1.033092	4199	0.461971	3405
Lv. 3	1.105831	3353	0.488081	2140
Lv. 4	2.002056	12186	0.957622	7571
Lv. 5	9.182086	62890	4.576726	37120

Building of Informatics, Technology and Science (BITS)
Volume 5, No 4, March 2024 Page: 659−671
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v5i4.4622

3.2 Evaluation of the Depth-First Search Algorithm Approach

Evaluation and comparative analysis of the performance of the depth-first search method in solving Sudoku puzzles. This analysis involved two types of Sudoku problems at each level, sourced from sudoku.com and extremesudoku.info respectively. Table 3., presents the performance metrics for two Sudoku puzzles, labeled Puzzle 1 and Puzzle 2, across different difficulty levels, including Easy, Medium, and Hard, from Sudoku.com. The time to complete each puzzle is measured in seconds, while the number of steps involved in the solving process is also documented. In Puzzle 1, the easy level required 0.056 seconds with 373 steps, the medium level took 1.488 seconds with 11,857 steps, and the hard level consumed 2.456 seconds with 19,646 steps. On the other hand, Puzzle 2 exhibited slightly different results, with Easy, Medium, and Hard levels taking 0.025 seconds (101 steps), 0.034 seconds (270 steps), and 0.480 seconds (3,468 steps), respectively. This comparative analysis provides insights into the efficiency of the solving algorithms across varying puzzle complexities, considering both time and step metrics.

Table 3. Results from Sudoku.com with DFS Algorithm

	Puzzle 1		Puzzle 2	
	Time (s)	**Step**	**Time (s)**	**Step**
Easy	0.056563	373	0.025617	101
Medium	1.488234	11857	0.034118	270
Hard	2.456386	19646	0.480075	3468

Table 4., presents the performance metrics of two different Sudoku puzzles, Puzzle 1 and Puzzle 2, across various difficulty levels Lv. 1 to 5 from extremesudoku.info. Puzzle 1's solving time and step count exhibit notable fluctuations across levels. Level 1 demands 2.603 seconds and 14992 steps, with a significant decrease in time but an increase in steps observed in Level 2. Surprisingly, Level 3 shows a meager solving time of 0.024 seconds and 102 steps. The complexity varies again in Level 4 before reaching its peak in Level 5, requiring 17.656 seconds and 115666 steps. Puzzle 2, on the other hand, displays a distinct pattern with a rapid decrease in solving time and steps from Level 1 to Level 3, followed by a subtle increase in Level 4 and a substantial surge in Level 5. This analysis underscores the diverse challenges posed by different Sudoku puzzles at varying difficulty levels, reflecting algorithmic adaptability and efficiency across scenarios.

Table 4. Results from Extremesudoku.info with DFS Algorithm

	Puzzle 1		Puzzle 2	
	Time (s)	**Step**	**Time (s)**	**Step**
Lv. 1	2.60369	14992	0.024895	167
Lv. 2	0.381973	2393	0.227792	1591
Lv. 3	0.024002	102	0.024004	84
Lv. 4	1.378444	7741	0.016799	114
Lv. 5	17.65697	115666	9.457493	79534

3.3 Evaluation of the Heuristic Search Algorithm Approach

Evaluation and comparative analysis of the performance of the proposed heuristic search method in solving Sudoku puzzles. This analysis involved two types of Sudoku problems at each level, sourced from sudoku.com and extremesudoku.info respectively. Table 5., presents the performance metrics of two Sudoku puzzles, labeled Puzzle 1 and Puzzle 2, about solving time and the number of steps involved across different difficulty levels, Easy, Medium, and Hard, from Sudoku.com. For Puzzle 1, the Easy level demonstrates a completion time of 0.02 seconds with 43 steps, while the medium level exhibits 0.09 seconds and 153 steps. Notably, the Hard level in Puzzle 1 records a more significant solving time of 1.07 seconds and a higher step count of 2182. Puzzle 2, on the other hand, showcases consistent completion times of 0.02 seconds for both Easy and Medium levels, each requiring 43 and 51 steps, respectively. In contrast, the Hard level in Puzzle 2 shows a slight increase in solving time to 0.14 seconds but maintains a relatively low step count of 219. This analysis provides a nuanced understanding of the algorithmic performance, between solving time and the complexity of Sudoku puzzles at various difficulty levels.

Table 5. Results from Sudoku.com with Heuristic Search Algorithm

	Puzzle 1		Puzzle 2	
	Time (s)	**Step**	**Time (s)**	**Step**
Easy	0.02	43	0.02	43
Medium	0.09	153	0.02	51
Hard	1.07	2182	0.14	219

Table 6., presents a comparative analysis of solving time and step count for Puzzle 1 and Puzzle 2 across different difficulty levels, identified as Lv. 1 to Lv. 5 extremesudoku.info. Puzzle 1's solving time gradually increases from 0.14 seconds in Lv. 1 to 0.23 seconds in Lv. 5, accompanied by an ascent in step count from 100 to 408.

Building of Informatics, Technology and Science (BITS)
Volume 5, No 4, March 2024 Page: 659−671
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v5i4.4622

Conversely, Puzzle 2 exhibits more variability, with a notable spike in solving time and step count in Lv. 2, reaching 0.27 seconds and 207 steps, respectively. Lv. 3 shows a moderate increase in solving time to 0.19 seconds, with a decrease in step count to 86. Lv. 4 witnesses a decline in solving time (0.06 seconds) and step count (128). Lv. 5 of Puzzle 2 records a substantial rise in solving time to 0.40 seconds and a notable increase in step count to 610. This comprehensive analysis offers insights into the algorithm's performance at different difficulty levels, highlighting its adaptability and efficiency.

Table 6. Results from Extremesudoku.info with Heuristic Search Algorithm

	Puzzle 1		Puzzle 2	
	Time (s)	**Step**	**Time (s)**	**Step**
Lv. 1	0.14	100	0.10	43
Lv. 2	0.05	260	0.27	207
Lv. 3	0.19	86	0.06	447
Lv. 4	0.06	374	0.02	128
Lv. 5	0.23	408	0.40	610

This comprehensive analysis underscores the escalating computational demands associated with higher difficulty levels, offering valuable insights into algorithmic efficiency and performance variations across distinct Sudoku puzzles.

3.4 Comparative Performance Evaluation of Algorithm Methods

The comparative visualization performance of the algorithm methods from two sources of Sudoku puzzles, sudoku.com and extremesudoku.info respectively. Figure 11., displays a comparison graph illustrating the time efficiency of three algorithms: breadth-first search, depth-first search, and a heuristic algorithm. The graph showcases the average time required by each algorithm to finish tasks across three distinct difficulty levels: easy, medium, and hard, based on sudoku.com data. Notably, the heuristic algorithm emerges as the swiftest performer across all difficulty levels. This superiority can be attributed to heuristic algorithms leveraging domain knowledge, enabling them to navigate and find solutions faster. Following closely, depth-first search ranks as the second fastest algorithm, trailed by breadth-first search in terms of completion time.

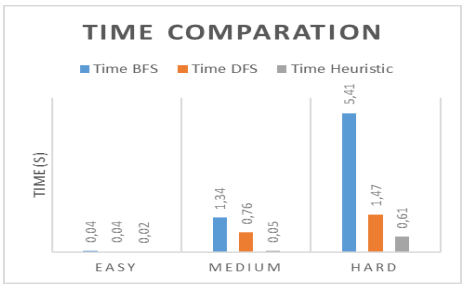

Figure 11. Comparison of Time performance between BFS, DFS, Heuristic Search Methods from sudoku.com

Figure 12., presents a graphical representation comparing the number of steps required to complete tasks of varying difficulty levels, categorized as easy, medium, and hard, based on data obtained from sudoku.com. The y-axis of the graph indicates the number of steps, while the x-axis represents the difficulty level of the tasks. As depicted in the graph, there is a clear upward trend in the number of steps required as the difficulty level of the tasks increases. For instance, tasks categorized as easy require a relatively low number of steps, while tasks classified as hard demand a significantly higher number of steps for completion. This trend highlights the inherent complexity associated with more challenging tasks, necessitating increased effort and problem-solving strategies to achieve successful outcomes. Overall, the graph provides a visual representation of how task difficulty impacts the number of steps required for completion, emphasizing the importance of effective planning and execution strategies in tackling more intricate tasks.

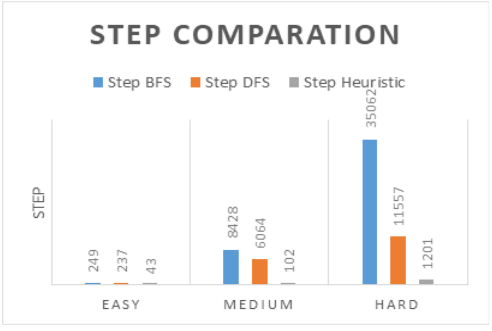

Figure 12. Comparison of Steps Performance Between BFS, DFS and Heuristic Search Methods from sudoku.com

Building of Informatics, Technology and Science (BITS)
Volume 5, No 4, March 2024 Page: 659−671
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v5i4.4622

Figure 13., displays a graph that compares the time complexity of three algorithms: breadth-first search (BFS), depth-first search (DFS), and a heuristic algorithm based on extremesudoku.info data. Time complexity measures the amount of time an algorithm requires to complete a task as the size of the input data increases. The graph illustrates that the heuristic algorithm exhibits the most favorable time complexity, followed by DFS and BFS. The heuristic algorithm is the fastest when handling tasks with more extensive input data. Overall, the heuristic algorithm stands out with the most efficient time complexity among the three algorithms. Its optimal performance makes it particularly suitable for tasks involving substantial input data sizes, such as determining the shortest path on a large map or solving complex combinatorial optimization problems to find optimal solutions.

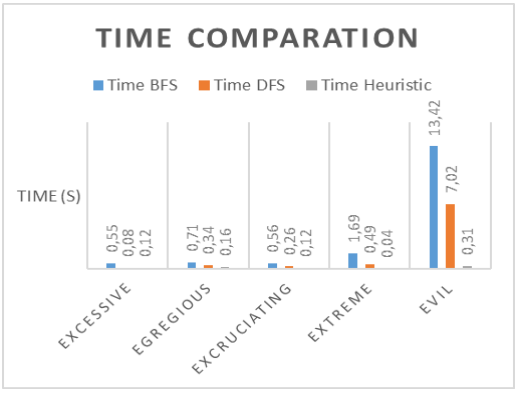

Figure 13. Comparison of Time performance between BFS, DFS, Heuristic Search Methods from extremesudoku.info

Figure 14., displays a graph that compares the performance of three distinct algorithms, BFS, DFS, and a heuristic algorithm, across various tasks. The y-axis denotes the performance metric, encompassing factors like the number of steps required to complete each task. Meanwhile, the x-axis categorizes tasks based on their difficulty levels. Notably, the heuristic algorithm consistently outperforms BFS and DFS across all tasks. This superiority is attributed to heuristic algorithms leveraging domain knowledge to guide their search, facilitating quicker solution discovery. The graph conclusively indicates that the heuristic algorithm is the most effective choice for tasks spanning different difficulty levels. This algorithm shines mainly when performance is crucial and relevant domain knowledge is available. Examples of such applications include employing the heuristic algorithm to find the shortest path in a road network or optimizing job scheduling for efficiency.

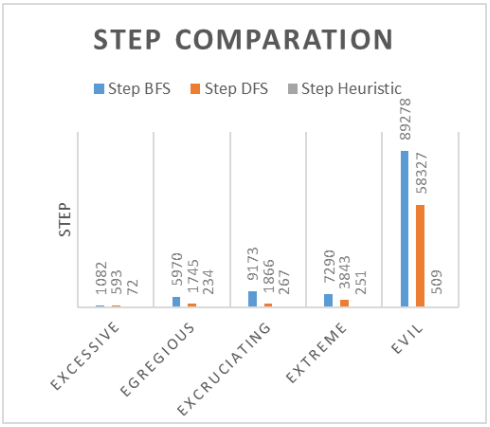

Figure 14. Comparison of Steps Performance Between BFS, DFS, Heuristic Search Methods from extremesudoku.info

Based on the results of the comparative analysis of the performance evaluation of the algorithmic approaches, two dataset sources were used to test the Sudoku puzzle: sudoku.com and extremesudoku.info. Utilizing the algorithmic methods, it is evident that the algorithm's optimization using the heuristic search method yields superior results in terms of the time and total steps required to solve the Sudoku puzzle, as illustrated in Figure. 11 and 13, where the heuristic search algorithm demonstrates the best results with the shortest time values required to complete the Sudoku puzzle. The time needed to complete the Sudoku puzzle on sudoku.com ranges from 0.02 to 0.61 seconds across different difficulty levels, while on extremesudoku.info, it takes 0.31 seconds at the highest difficulty level. This algorithm's superior performance is attributed to its inclusion in the Informed Search algorithm category. Regarding the number of steps required to solve the Sudoku puzzle, as shown in Figure. 12 and 14, the heuristic search algorithm also yields the best results with the fewest steps. The total steps required on sudoku.com range from 43 to 1201 across different difficulty levels, whereas extremesudoku.info requires 509 at its highest level. These results

Building of Informatics, Technology and Science (BITS)
Volume 5, No 4, March 2024 Page: 659−671
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v5i4.4622

underscore the optimization capabilities of the Heuristic Search Algorithm, which leverages information to outperform other algorithms. This research aimed to enhance Sudoku puzzle-solving efficiency and reduce memory usage through the implementation of the heuristic search algorithm, comparing its performance with traditional breadth-first and depth-first search methods.

4. CONCLUSION

This research is focused on enhancing algorithm performance using heuristic search to finish Sudoku puzzles of varying difficulty levels. The study also compared with the Depth-First Search and Breadth-First Search methods. Memory-intensive demands in breadth-first search and the potential issue of infinite traversal in depth-first search were identified, prompting the proposal of the heuristic search algorithm as a solution. The study conducted a comparative analysis across different difficulty levels, utilizing data from sudoku.com and extremesudoku.info to evaluate algorithmic performance. The application of search algorithms intended for Sudoku puzzles has proven highly effective, completing tasks in significantly less time than manual solutions. Qualitative findings highlight the superiority of the informed search method over uninformed search in Sudoku puzzles. This superiority is attributed to the structured algorithm and clear search objectives of the informed search method, resulting in faster and more efficient outcomes. The depth-first search approach demonstrates greater effectiveness than breadth-first search within the uninformed search method. The Heuristic Search Algorithm, falling under Informed Search, stands out in terms of optimization compared to breadth-first search and depth-first search, showcasing superior results in completion steps and time requirements. The Heuristic Search Algorithm attains the most minor completion steps, with 43 steps on sudoku.com and 86 on extremesudoku.info. Moreover, the smallest completion time remains consistent at 0.02 seconds for both sources. Results consistently showed that the heuristic search algorithm outperformed traditional methods, demonstrating faster completion times and reduced memory requirements. The strategic selection of Sudoku puzzles ensured the robustness and validity of the study's conclusions, highlighting the efficacy of the heuristic search algorithm in optimizing Sudoku puzzle-solving. These results affirm the reliability of heuristic search, consistently delivering favorable outcomes and outperforming other algorithms across diverse conditions. Overall, this research contributes to the advancement of Sudoku-solving algorithms and underscores the potential of heuristic search in problem-solving scenarios. For future research, the study recommends exploring alternative data sources with optimization algorithms [7] and testing additional methods, like the backtracking method, to further enhance completion speed included time or steps [4][11].

REFERENCES

[1] A. Jungherr and D. B. Schlarb, “The extended reach of game engine companies: How companies like epic games and Unity technologies provide platforms for extended reality applications and the metaverse,” *Soc. Media+ Soc.*, vol. 8, no. 2, p. 20563051221107640, 2022.
[2] Herimanto, P. Sitorus, and E. M. Zamzami, “An Implementation of Backtracking Algorithm for Solving A Sudoku-Puzzle Based on Android,” *J. Phys. Conf. Ser.*, vol. 1566, no. 1, 2020, doi: 10.1088/1742-6596/1566/1/012038.
[3] B. de Kegel and M. Haahr, “Procedural puzzle generation: A survey,” *IEEE Trans. Games*, vol. 12, no. 1, pp. 21–40, 2020, doi: 10.1109/TG.2019.2917792.
[4] 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,” vol. 5, no. 1, pp. 29–35, 2023.
[5] A. Narayanaswamy and P. Shrivastava, “Image Detection and Digit Recognition to solve Sudoku as a Constraint Satisfaction Problem,” 2019.
[6] R. H. Chae and A. C. Regan, “An analysis of Harmony Search for solving Sudoku puzzles,” *Soft Comput. Lett.*, vol. 3, p. 100017, 2021.
[7] C. Wang *et al.*, “A Novel Evolutionary Algorithm with Column and Sub-Block Local Search for Sudoku Puzzles,” *IEEE Trans. Games*, vol. PP, pp. 1–11, 2023, doi: 10.1109/TG.2023.3236490.
[8] R. Rahim, R. Dijaya, M. T. Multazam, A. D. Gs, and D. Sudrajat, “Puzzle game solving with breadth first search algorithm,” *J. Phys. Conf. Ser.*, vol. 1402, no. 6, 2019, doi: 10.1088/1742-6596/1402/6/066040.
[9] H. Lloyd and M. Amos, “Solving Sudoku with Ant Colony Optimization,” *IEEE Trans. Games*, vol. 12, no. 3, pp. 302–311, 2020, doi: 10.1109/TG.2019.2942773.
[10] T. Cazenave, “Monte Carlo Game Solver,” 2020, doi: 10.1007/978-3-030-89453-5_5.
[11] T. H. Lim and P. L. Ng, “Evaluating recursive backtracking depth-first search algorithm in unknown search space for self-learning path finding robot,” in *Artificial Intelligence for Communications and Networks: Second EAI International Conference, AICON 2020, Virtual Event, December 19-20, 2020, Proceedings 2*, Springer, 2021, pp. 531–543.
[12] R. Jain and M. Patel, “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,” *IJSDR2302112 Int. J. Sci. Dev. Res.*, 2023.
[13] H. Wen and W. Zhang, “Improving Parallelism of Breadth First Search (BFS) Algorithm for Accelerated Performance on GPUs,” in *2019 IEEE High Performance Extreme Computing Conference (HPEC)*, IEEE, 2019, pp. 1–7.
[14] J. Li, “Efficient and Effective Techniques for Large-Scale Multi-Agent Path Finding.” University of Southern California, 2022.
[15] A. Shafique, F. Sohail, and A. I. Qadri, “Comparative Analysis of Search Algorithms in AI,” no. October, 2022, doi: 10.13140/RG.2.2.29282.61123.

Building of Informatics, Technology and Science (BITS)
Volume 5, No 4, March 2024 Page: 659−671
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v5i4.4622

[16] "Sudoku Data Source (extremesudoku.info)." [Online]. Available: https://extremesudoku.info/
[17] "Sudoku Data Source (sudoku.com)." [Online]. Available: https://sudoku.com/
[18] T. Sasaki, D. Miyahara, T. Mizuki, and H. Sone, "Efficient card-based zero-knowledge proof for Sudoku," *Theor. Comput. Sci.*, vol. 839, pp. 135–142, 2020, doi: 10.1016/j.tcs.2020.05.036.
[19] K. Khanchandani, A. Pandey, S. Khan, M. Patil, and R. Rajan, "Automated Sudoku Analyser," *Asian J. Converg. Technol.*, vol. V, no. I, pp. 1–7, 2019.
[20] J. Luo, "Exploring Robot Morphology Spaces through Breadth-First Search and Random Query," *arXiv Prepr. arXiv2309.14387*, 2023.
[21] A. B. Matos, "The most difficult Sudoku puzzles are quickly solved by a straightforward depth-first search algorithm," 2016, [Online]. Available: https://www.dcc.fc.up.pt/~acm/sudoku.pdf
[22] D. Chen, Y. Bai, W. Zhao, S. Ament, J. M. Gregoire, and C. P. Gomes, "Deep reasoning networks for unsupervised pattern de-mixing with constraint reasoning," *37th Int. Conf. Mach. Learn. ICML 2020*, vol. PartF16814, pp. 1477–1486, 2020.
[23] Y. Björnsson, S. Helgason, and A. Pálsson, "Searching for Explainable Solutions in Sudoku," *IEEE Conf. Games*, 2021.
[24] Y. Wu and H. Nakayama, "Graph-Based Heuristic Search for Module Selection Procedure in Neural Module Network," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 12624 LNCS, pp. 560–575, 2021, doi: 10.1007/978-3-030-69535-4_34.
[25] V. Maniezzo, T. Stützle, and S. Voß, *Matheuristics*. Springer, 2021.
[26] M. Elbes, S. Alzubi, T. Kanan, A. Al-Fuqaha, and B. Hawashin, "A survey on particle swarm optimization with emphasis on engineering and network applications," *Evol. Intell.*, vol. 12, pp. 113–129, 2019.
[27] B. Cserna, W. J. Doyle, J. S. Ramsdell, and W. Ruml, "Avoiding dead ends in real-time heuristic search," *32nd AAAI Conf. Artif. Intell. AAAI 2018*, pp. 1306–1313, 2018.