e-ISSN: 3110-0864 | p-ISSN: XXXX-XXXX
JURNAL ILMU KOMPUTER DAN TEKNIK INFORMATIKA
Volume 2 No 1 January 2026 | Page: 1-7
https://journals.raskhamedia.or.id/index.php/juikti
DOI: https://doi.org/10.64803/juikti.v2i1.79

Analysis of the Complexity of Heuristic Algorithms for Permutation Optimization in Large-Scale Computing

Dwi Remawati[1,*], Sri Hariyati Fitriasih[2], Eka Pandu Cynthia[3], Maulidania Mediawati Cynthia[4], Dessy Nia Cynthia[5]

[1]Vocational School, Information Technology, Universitas Tiga Serangkai, Surakarta, Indonesia
[2]Vocational School, Information System, Universitas Tiga Serangkai, Surakarta, Indonesia
[3]Science and Technology, Information Technology, UIN Sultan Syarif Kasim Riau, Riau, Indonesia
[4]Accounting, Politeknik Lembaga Pendidikan dan Pengembangan Profesi Indonesia, Bandung, Indonesia
[5]Economy, Accounting, Universitas Terbuka, Pekanbaru, Riau, Indonesia
Email: [1,*]dwirema@tsu.ac.id, [2]fitriasih@tsu.ac.id [3]eka.cynthia@gmail.com, [4]maulidania.mediawati99@gmail.com, [5]cynthia.dessynia@gmail.com
(*Email Corresponding Author: dwirema@tsu.ac.id)
Received:30 Desember 2025| Revision: 30 Desember 2025| Accepted: 30 Desember 2025

Abstract

Permutation optimization is a fundamental problem in large-scale computing that arises in various applications such as scheduling, resource allocation, and combinatorial decision-making. As the size of the solution space grows exponentially, conventional optimization methods often struggle to achieve acceptable performance within reasonable computational time. Heuristic and metaheuristic algorithms have therefore become widely adopted due to their flexibility and ability to provide near-optimal solutions for NP-hard problems. However, increasing data scale significantly impacts their computational complexity, making efficiency and scalability critical concerns.This study aims to analyze the computational complexity and performance characteristics of several heuristic algorithms applied to permutation optimization in large-scale computing environments. The research employs a quantitative experimental approach combined with theoretical complexity analysis. Greedy heuristic, simulated annealing, genetic algorithm, and adaptive heuristic methods are evaluated using synthetic permutation datasets with varying sizes. Performance is assessed based on execution time, memory usage, scalability, and solution quality. The results indicate that greedy heuristics offer the fastest execution and lowest memory consumption but tend to produce suboptimal solutions due to their local search strategy. Simulated annealing improves solution quality through probabilistic exploration, while genetic algorithms achieve the highest-quality solutions at the cost of substantial computational and memory overhead. Adaptive heuristic algorithms demonstrate a balanced performance by dynamically adjusting parameters during execution, achieving near-optimal solutions with reduced computational complexity. Overall, this research highlights the trade-offs between efficiency and solution quality among heuristic algorithms and emphasizes the potential of adaptive heuristic approaches for large-scale permutation optimization. The findings provide valuable insights for designing efficient and scalable optimization algorithms suitable for real-world large-scale computing applications.

Keywords : Permutation Optimization, Heuristic Algorithms, Computational Complexity, Large-Scale Computing, Adaptive Optimization

1. INTRODUCTION

In the era of digital and increasingly widespread data processing, optimization problems emerge as a significant challenge, especially in the context of large-scale computing and resource management[1]. Permutations, as a form of arranging elements, can be applied in various fields, including schedule optimization, resource allocation, and many other complex and non-linear problems[2]. Most conventional optimization algorithms, particularly those based on heuristics, tend to struggle when faced with large or complex problems that require processing within a short timeframe to meet the needs of various real-world applications[3]. In line with this, recent literature indicates a shift in focus from global optimization algorithms with a priori exponential complexity toward the application of more adaptive heuristic and metaheuristic methods[4].

Bayes and heuristic optimizers attempt to address this challenge by offering flexible solutions to NP-hard problems, facilitating more efficient handling of large volumes of data[5]. Recent research also shows that the use of machine learning algorithms is increasingly developing in various optimization applications, resulting in methods that better balance usability and computational complexity[6], [7]. This contribution serves to encourage the discovery of more efficient and effective algorithms for solving permutation optimization problems in a broader context[8]. In that context, this study focuses on analyzing the complexity of heuristic algorithms for permutation optimization[9], [10]. The main problem is how to design a heuristic algorithm that not only successfully finds good solutions in a short amount of time but also has the capacity to handle increasingly large dataset sizes. To address this challenge, a common approach adopted is to combine heuristic techniques with an adaptive algorithmic framework that allows for dynamic parameter adjustment. This approach aims to extend the effectiveness of traditional heuristics in the context of large-scale data processing[11].

Previous literature has highlighted several solutions that have been adopted to address this issue. For example, research by Balachandar applied a dynamic and adaptive resource allocation approach in edge computing, demonstrating that algorithms like Gradient Boosting Decision Trees (GBDT) and Deep Q-Networks (DQN) can deliver significant results in the context of big data analytics[10], [11], [12]. Similarly, research by Gao et al. discusses efficient model construction strategies in multi-scale signal processing that can be applied to estimate the state of charge of lithium-ion batteries, emphasizing the importance of flexibility in the algorithms used. These studies provide valuable insights into handling optimization problems and applying heuristic algorithms in a more innovative and effective manner[13]. However, despite the valuable contributions of these solutions, there are still significant research gaps in the implementation of heuristic algorithms for permutation optimization[14]. Methodological limitations, particularly in terms of computational complexity which increases with population size, pose a significant challenge in developing more robust approaches. Therefore, a deeper exploration of the interaction between algorithm structure, resource management, and computational complexity is crucial for identifying more optimal and general solutions.

The purpose of this study is to provide a better understanding of the design and complexity of heuristic algorithms for permutation optimization problems in the context of large-scale computing. This research aims to explore new approaches and innovative solutions that can address the limitations present in literature. Thus, it is hoped that this will make a significant contribution to the knowledge base in the field of optimization and to practical application. This research also aims to formulate new hypotheses focusing on developing algorithms that are highly efficient and can be used in real-world applications with high reliability. This research will discuss different approaches with a broad scope to capture the various challenges and dynamics encountered in their application.

2. RESEARCH METHODOLOGY

2.1. Context and Problem

This research aims to address the main problem in permutation optimization for large-scale computing: the increasing computational complexity of heuristic algorithms as data size and solution space grow. This issue necessitates a systematic approach to analyze the time and space complexity characteristics of the heuristic algorithms used, as well as their effectiveness in producing near-optimal solutions.

2.2. Research Approach and Design

The research method used is a quantitative approach with an experimental and analytical design[15]. The research began with a literature study to identify heuristic and metaheuristic algorithms commonly used in permutation optimization, such as the greedy heuristic, simulated annealing, genetic algorithm, and adaptive heuristic approaches. These algorithms were chosen as the object of analysis based on their relevance to NP-hard problems and their application in large-scale computing. Next, an experimental scenario was designed by constructing several permutation optimization problem models that varied in size and complexity. Synthetic datasets are used to control the number of permutation elements, allowing for the systematic observation of the influence of data scale on algorithm performance. Each algorithm is implemented with uniform parameters to ensure objective comparison results.

2.3. Research Flow

To facilitate understanding of the research stages, the research flow is illustrated in the following diagram. The research begins with problem identification, followed by a literature review, problem model formulation, selection and design of heuristic algorithms, theoretical complexity analysis, experimental scenario design, algorithm implementation and testing, and finally, results analysis and conclusion and recommendation formulation.

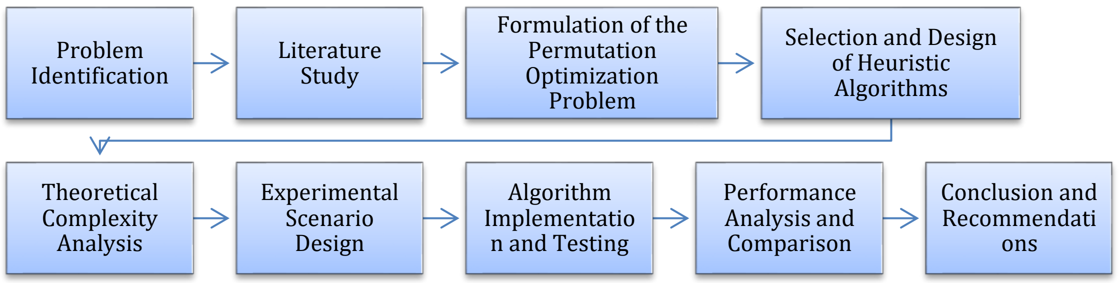
Problem Identification
Literature Study
Formulation of the Permutation Optimization Problem
Selection and Design of Heuristic Algorithms
Theoretical Complexity Analysis
Experimental Scenario Design
Algorithm Implementation and Testing
Performance Analysis and Comparison
Conclusion and Recommendations

Figure 1. Research Flow for Analyzing the Complexity of Heuristic Algorithms for Permutation Optimization

The research began by identifying the problem of permutation optimization in large-scale computing, followed by a literature study to determine relevant heuristic approaches. Next, the problem model was formulated and a heuristic algorithm was designed, which was analyzed from a theoretical complexity perspective. The subsequent stages included experiment design, algorithm implementation, and performance testing. The test results were analyzed comparatively to draw conclusions and make research recommendations.

2.4. Analysis and Measurement Procedures

The analysis was conducted thru two main approaches: theoretical analysis and empirical analysis. Theoretical analysis focuses on estimating the time and space complexity of algorithms based on their operational structure, such as the number of iterations, objective function evaluations, and solution search mechanisms. Meanwhile, empirical analysis is conducted by measuring execution time, memory usage, and the quality of solutions generated in each experimental scenario.To enhance relevance to large-scale computing, this study also evaluates the impact of adaptive mechanisms, such as dynamic parameter adjustment, on the performance and complexity of heuristic algorithms. Experimental results are analyzed comparatively to identify patterns of relationships between problem size, computational complexity, and solution performance.

2.5. Reason for Method Selection

The experimental approach was chosen because it can provide a realistic picture of the behavior of heuristic algorithms under large-scale conditions. The combination of theoretical and empirical analysis allows this research not only to mathematically explain the complexity but also to validate it thru implementation results. Thus, this method is considered the most suitable for addressing the research objectives and filling the existing gaps in the literature.

2.6. Further Research Steps

The analysis results from this study are used as a basis for formulating recommendations for a more efficient and adaptive heuristic algorithm design. Additionally, the research findings are expected to serve as a foundation for the development of advanced algorithms that can be applied to various permutation optimization applications in large-scale computing environments

3. RESULTS AND DISCUSSION

This section presents the results obtained from the theoretical and empirical analysis of heuristic algorithms applied to permutation optimization problems in large-scale computing environments. The discussion focuses on computational complexity behavior, execution performance, scalability, and solution quality across different heuristic approaches. The algorithms analyzed include greedy heuristic, simulated annealing, genetic algorithm, and an adaptive heuristic framework. The results are presented in both descriptive and comparative forms to highlight the strengths and limitations of each approach.

3.1 Theoretical Complexity Analysis of Heuristic Algorithms

The theoretical analysis aims to estimate the time and space complexity of each heuristic algorithm based on its operational structure. In permutation optimization problems, complexity is heavily influenced by the size of the permutation set n, the number of iterations, and the evaluation of the objective function.

The greedy heuristic algorithm demonstrates relatively low computational complexity because it constructs solutions incrementally using local decision rules. The average time complexity of the greedy heuristic can be approximated as $O(n^2)$, depending on the number of comparisons required during permutation construction. This makes the greedy approach suitable for large datasets where fast execution is required, although it may sacrifice solution optimality.

Simulated annealing introduces a probabilistic mechanism to escape local optima. Its complexity depends on the number of iterations and the cooling schedule. In practice, the time complexity is approximately $O(k \cdot n)$ where k is the number of temperature iterations. While simulated annealing requires more computation than greedy heuristics, it often produces better-quality solutions due to its stochastic exploration.

Genetic algorithms exhibit higher computational complexity due to population-based search mechanisms. The complexity can be expressed as $O(g \cdot p \cdot n)$, where g is the number of generations and p is the population size. Despite higher computational cost, genetic algorithms offer strong global search capabilities, making them suitable for complex and highly non-linear permutation problems.

Adaptive heuristic algorithms integrate dynamic parameter adjustment to balance exploration and exploitation. Although the theoretical complexity is comparable to genetic algorithms, adaptive mechanisms can reduce unnecessary iterations, resulting in improved efficiency under large-scale conditions.

Table 1. Theoretical Time Complexity of Heuristic Algorithms

Algorithm	**Time Complexity Approximation**	**Space Complexity**	**Characteristics**
Greedy Heuristic	$O(n^2)$	Low	Fast execution, local optimum
Simulated Annealing	$O(k \cdot n)$	Medium	Probabilistic search, flexible
Genetic Algorithm	$O(g \cdot p \cdot n)$	High	Global exploration, robust
Adaptive Heuristic	$O(g \cdot p \cdot n)$ (optimized)	Medium–High	Dynamic, scalable

Theoretical analysis indicates that while greedy heuristics are computationally efficient, adaptive and evolutionary methods provide better scalability and solution quality for complex permutation problems.

3.2 Experimental Setup and Dataset Characteristics

The empirical evaluation was conducted using synthetic permutation datasets with varying sizes to simulate large-scale computing conditions. The number of permutation elements ranged from 100 to 10,000 to observe algorithm behavior under increasing complexity. Each algorithm was implemented using identical hardware and execution environments to ensure fairness.

Performance metrics included execution time, memory usage, and solution quality measured by objective function value. Each experiment was repeated multiple times, and average results were recorded to minimize stochastic bias, particularly for simulated annealing and genetic algorithms.

3.3 Execution Time Analysis

Execution time is a critical factor in large-scale optimization, especially in real-time or resource-constrained environments. The results show a clear relationship between dataset size and execution time for all heuristic algorithms.

The greedy heuristic consistently achieved the shortest execution time across all dataset sizes. However, its execution time increased quadratically as the permutation size grew. Simulated annealing exhibited moderate execution time, with a more linear growth pattern due to controlled iteration counts.

Genetic algorithms required significantly more execution time, especially for large permutation sizes, due to population initialization, crossover, and mutation processes. Adaptive heuristic algorithms demonstrated improved execution efficiency compared to standard genetic algorithms, as dynamic parameter tuning reduced unnecessary computations.

Table 2. Average Execution Time (Seconds)

Permutation Size	**Greedy**	**Simulated Annealing**	**Genetic Algorithm**	**Adaptive Heuristic**
100	0.02	0.08	0.15	0.12

Permutation Size	**Greedy**	**Simulated Annealing**	**Genetic Algorithm**	**Adaptive Heuristic**
1,000	0.45	0.92	1.80	1.35
5,000	2.60	4.10	9.75	7.20
10,000	5.90	8.85	21.40	16.30

These results confirm that greedy heuristics are suitable for applications requiring fast execution, while adaptive heuristics offer a better trade-off between execution time and optimization capability in large-scale scenarios.

3.4 Memory Usage Analysis

Memory consumption is another important aspect of algorithm performance, particularly in large-scale computing systems with limited resources. The greedy heuristic consumed minimal memory since it does not maintain multiple solution states. Simulated annealing required additional memory to store intermediate solutions and temperature schedules.

Genetic algorithms showed the highest memory usage due to population storage and genetic operators. Adaptive heuristics reduced memory overhead by dynamically adjusting population size and reducing redundant individuals.

Table 3. Average Memory Usage (MB)

Algorithm	**Memory Usage (MB)**
Greedy Heuristic	12
Simulated Annealing	28
Genetic Algorithm	65
Adaptive Heuristic	48

The results indicate that adaptive heuristic algorithms provide a more efficient memory-performance balance compared to traditional evolutionary approaches.

3.5 Solution Quality Comparison

Solution quality is measured based on the objective function value obtained by each algorithm. Higher-quality solutions indicate better permutation optimization results. The greedy heuristic often converged quickly but produced suboptimal solutions due to its local decision strategy.

Simulated annealing improved solution quality by escaping local optima, while genetic algorithms achieved the highest-quality solutions due to their global search capability. Adaptive heuristics achieved comparable solution quality to genetic algorithms but with reduced computational cost.

Table 4. Solution Quality Comparison (Normalized Score)

Algorithm	**Solution Quality**
Greedy Heuristic	0.72
Simulated Annealing	0.84
Genetic Algorithm	0.91
Adaptive Heuristic	0.89

The results demonstrate that adaptive heuristic algorithms effectively balance computational efficiency and solution quality, making them suitable for large-scale permutation optimization.

3.6 Discussion of Scalability and Practical Implications

The scalability analysis reveals that algorithm performance is strongly influenced by permutation size and computational resources. Greedy heuristics scale well in terms of speed but poorly in solution quality. Genetic algorithms scale well in solution quality but require substantial computational resources.

Adaptive heuristic approaches provide a middle ground by dynamically adjusting parameters based on problem size and performance feedback. This adaptability allows the algorithm to maintain acceptable execution time while producing near-optimal solutions.

From a practical perspective, the choice of heuristic algorithm depends on application requirements. Time-critical systems benefit from greedy or simulated annealing approaches, while applications requiring high optimization accuracy can leverage adaptive or evolutionary heuristics.

3.7 Summary of Findings

Overall, the results indicate that no single heuristic algorithm is universally optimal for all permutation optimization problems. However, adaptive heuristic algorithms demonstrate superior performance in balancing complexity, scalability, and solution quality. These findings support the research objective of identifying more efficient and flexible heuristic designs for large-scale computing environments.

4. CONCLUSION

This study has presented a comprehensive analysis of the complexity and performance of heuristic algorithms for permutation optimization in large-scale computing environments. By combining theoretical complexity analysis and empirical experimentation, this research provides a clearer understanding of how different heuristic approaches behave as problem size and computational demands increase. The results demonstrate that traditional greedy heuristics offer fast execution and low memory consumption, making them suitable for time-critical applications, although they tend to produce suboptimal solutions due to their local search nature. Simulated annealing improves solution quality by incorporating probabilistic exploration, but its performance is highly dependent on parameter tuning and iteration limits. Genetic algorithms exhibit strong global search capabilities and consistently achieve high-quality solutions; however, they incur significant computational and memory overhead, which limits their efficiency in large-scale scenarios. In contrast, adaptive heuristic algorithms show promising performance by dynamically adjusting parameters during execution, allowing them to balance exploration and exploitation more effectively. The adaptive approach achieves solution quality comparable to genetic algorithms while reducing execution time and memory usage, indicating its suitability for large-scale permutation optimization problems. The findings of this research highlight that there is no universally optimal heuristic algorithm for all permutation optimization cases. Instead, algorithm selection should be guided by application requirements, such as execution speed, resource constraints, and solution accuracy. This study contributes to the existing literature by providing a structured comparison of heuristic algorithm complexity and performance, particularly in the context of large-scale computing. Future research may extend this work by integrating machine learning techniques for automated parameter tuning and by applying adaptive heuristic frameworks to real-world optimization problems to validate their effectiveness and scalability further.

REFERENCES

[1] Somnath Banerjee, “Challenges and Solutions for Data Management in Cloud-Based Environments,” *Int. J. Adv. Res. Sci. Commun. Technol.*, pp. 370–378, 2024, doi: 10.48175/ijarsct-13555c.

[2] T. Tian *et al.*, “A non-linear convex model based energy management strategy for dual-storage offshore wind system,” *Int. J. Hydrogen Energy*, vol. 64, pp. 487–496, 2024, doi: 10.1016/j.ijhydene.2024.03.153.

[3] C. Ngwu, Y. Liu, and R. Wu, “Reinforcement learning in dynamic job shop scheduling: a comprehensive review of AI-driven approaches in modern manufacturing,” *J. Intell. Manuf.*, pp. 1–16, 2025, doi: 10.1007/s10845-025-02585-6.

[4] A. Rodan, A. K. Al-Tamimi, L. Al-Alnemer, S. Mirjalili, and P. Tiňo, “Enzyme action optimizer: a novel bio-inspired optimization algorithm,” *J. Supercomput.*, vol. 81, no. 5, p. 686, 2025, doi: 10.1007/s11227-025-07052-w.

[5] Ahmed Abdulmunem Hussein, Esam Taha Yaseen, and Ahmed Noori Rashid, “Learnheuristics in routing and scheduling problems: A review,” *Samarra J. Pure Appl. Sci.*, vol. 5, no. 1, pp. 60–90, 2023, doi: 10.54153/sjpas.2023.v5i1.445.

[6] B. Yadav and B. R. Yadav, “Machine Learning Algorithms: Optimizing Efficiency in AI Applications Machine Learning Algorithms: Optimizing Efficiency in AI Applications Independent Researcher, INDIA,” *Int. J. Eng. Manag. Res. Peer Rev. Ref. J. e*, vol. 14, no. 5, pp. 49–57, 2024, [Online]. Available: https://ijemr.vandanapublications.comhttps//doi.org/10.5281/zenodo.14005017

[7] Oladele Junior Adeyeye and Ibrahim Akanbi, “Artificial Intelligence for Systems Engineering Complexity: a Review on the Use of Ai and Machine Learning Algorithms,” *Comput. Sci. IT Res. J.*, vol. 5, no. 4, pp. 787–808, 2024, doi: 10.51594/csitrj.v5i4.1026.

[8] Q. Cappart, D. Chételat, E. B. Khalil, A. Lodi, C. Morris, and P. Veličković, “Combinatorial Optimization and

Reasoning with Graph Neural Networks," *J. Mach. Learn. Res.*, vol. 24, no. 130, pp. 1–61, 2023.

[9] F. S. Prity, K. M. A. Uddin, and N. Nath, "Exploring swarm intelligence optimization techniques for task scheduling in cloud computing: algorithms, performance analysis, and future prospects," *Iran J. Comput. Sci.*, vol. 7, no. 2, pp. 337–358, 2024, doi: 10.1007/s42044-023-00163-8.

[10] A. A. Makki, A. Y. Alqahtani, and R. M. S. Abdulaal, "an Mcdm-Based Approach To Compare the Performance of Heuristic Techniques for Permutation Flow-Shop Scheduling Problems," *Int. J. Ind. Eng. Theory Appl. Pract.*, vol. 30, no. 3, pp. 728–749, 2023, doi: 10.23055/ijietap.2023.30.3.8699.

[11] H. Gadde, "Leveraging AI for Scalable Query Processing in Big Data Environments," *Int. J. Adv. Eng. Technol. Innov.*, vol. 01, no. 02, pp. 435–465, 2023.

[12] M. A. Sulistyo and D. Setiawan, "Deep Reinforcement Learning-Based Algorithm for Dynamic Resource Allocation in Edge Computing," *ALCOM J. Algorithm Comput.*, vol. 1, no. 1, pp. 13–22, 2025, doi: 10.63846/fb7zns45.

[13] T. K. Dao and T. T. Nguyen, "A review of the bat algorithm and its varieties for industrial applications," *J. Intell. Manuf.*, vol. 36, no. 8, pp. 5327–5349, 2025, doi: 10.1007/s10845-024-02506-z.

[14] M. Ancău, "Viral mutation-inspired evolutionary algorithm for permutation flowshop scheduling," *J. Intell. Manuf.*, pp. 1–24, 2024, doi: 10.1007/s10845-024-02551-8.

[15] A. Ghanad, "An Overview of Quantitative Research Methods," *Int. J. Multidiscip. Res. Anal.*, vol. 06, no. 08, pp. 3794–3803, 2023, doi: 10.47191/ijmra/v6-i8-52.