Bio-Inspired AI for Adaptive Sorting Algorithms

Shravinya Gadeela[1*]

Corresponding author(s). E-mail(s): shravinyagoud@gmail.com;

Abstract

Sorting algorithms are fundamental to a wide range of computational tasks. Optimizing their performance can significantly enhance efficiency in various applications, such as data processing and search algorithms. This paper investigates the optimization of sorting algorithms using two metaheuristic techniques: Genetic Algorithm (GA) and Particle Swarm Optimization (PSO). The goal is to select the most efficient sorting algorithm for a given dataset based on its performance, measured in terms of sorting time. An Artificial Neural Network (ANN) is employed to predict sorting times for different algorithms based on key features of the dataset, such as array length and range. The results demonstrate that both GA and PSO successfully identify the best sorting algorithms, with significant improvements in sorting time compared to baseline algorithms. The predicted sorting times from the ANN model further validate the optimization process, showing improvements of up to 46.97% for GA and 43.43% for PSO. This approach provides a novel framework for optimizing sorting tasks and can be extended to other computational problems requiring algorithm selection.

Keywords: Adaptive Sorting Artificial Neural Networks (ANN) Bio-Inspired Algorithms Genetic Algorithms (GA) Particle Swarm Optimization (PSO) Sorting Algorithms

1 Introduction

Sorting is one of the most basic operations in computer science and has a wide range of applications in fields such as database management, search engines, data analytics, and even machine learning[1]. The basis of many algorithms and systems is the process of arranging data in a specific order—whether ascending or descending. In particular, sorting plays a critical role in optimizing search operations, ensuring that elements are easily accessible, and facilitating efficient data retrieval. As such, the development of efficient and scalable sorting algorithms is essential for the performance of various applications, ranging from simple file management to complex, large-scale data processing[2].

Traditional sorting algorithms, such as QuickSort, MergeSort, and HeapSort, have been widely used for decades. These algorithms are well-established because of predictable performance in terms of time complexity[3]. QuickSort, for instance, is an average time complexity O(n log n) efficient algorithm for sorting large data sets. MergeSort, with a guaranteed time complexity of O(n log n), is stable and reliable; it is especially preferred in the case of linked lists and external sorting. HeapSort, on the other hand, offers O(n log n) performance with constant space complexity that works well in memory-constrained environments[4]. These algorithms are optimized and have been so extensively studied that they do amazingly well in many situations.

Additionally, when working in dynamic environments, wherein the characteristics of the data being sorted keep changing rapidly, adaptive sorting algorithms become ever more necessary[5]. A dynamic environment refers to an application where the data that has to be sorted is not fixed and can change with regard to distribution or pattern. Examples of such environments include real-time data streams, online systems, and applications that process data with varying patterns or distributions[6]. In these cases, the ability to adapt to the changing nature of the data is essential for maintaining optimal sorting performance. Traditional sorting algorithms, with their fixed strategies and assumptions, are not well-suited for such environments[7].

1.1 Problem Statement

Traditional sorting algorithms are static and struggle with non-uniform or real-time data streams. These algorithms typically rely on fixed strategies that assume a certain distribution or structure of the input data. When the input data deviates from these assumptions, the performance of these algorithms can deteriorate. Moreover, in dynamic environments, where data patterns change over time, traditional algorithms are unable to adapt to these changes, leading to suboptimal performance.

1.2 Objective

This paper introduces a bio-inspired adaptive sorting algorithm that combines genetic algorithms, particle swarm optimization, and artificial neural networks to improve performance and adaptability. By leveraging these bio-inspired techniques, the proposed algorithm dynamically adjusts its sorting strategy based on the characteristics of the input data. This hybrid approach offers several advantages over traditional sorting methods, including the ability to adapt to a wide variety of data distributions, ensuring efficient performance even in non-random or skewed datasets. The algorithm learns from experience and improves over time, making it well-suited for dynamic and unpredictable environments.

2 Background

Sorting is a fundamental operation in computer science with applications in databases, search engines, data analysis, and more. Traditional sorting algorithms such as QuickSort, MergeSort, and BubbleSort have been the standard for many years due to their efficiency and predictable time complexities[8]. However, these algorithms are often static and do not adapt well to changing data distributions. For example, when data is not uniformly distributed or follows non-random patterns, traditional sorting algorithms may struggle to maintain efficiency, leading to performance degradation. Furthermore, in real-time data streams or dynamic environments, where data characteristics change rapidly, traditional sorting algorithms face significant limitations.

2.1 Bio-Inspired AI Algorithms

The algorithms, particularly Genetic Algorithms, Particle Swarm Optimization, and Artificial Neural Networks, that are some of the most known of this class of bio-inspired algorithms derive inspiration mainly from biology and its knowledge, particularly that pertaining to evolution, collective behavior, and neural processing[9].

2.2 Related work

Sorting algorithms have been extensively studied, with traditional algorithms like QuickSort, MergeSort, and HeapSort being widely used due to their predictable performance[10]. However, these algorithms assume random or uniformly distributed data, and their performance degrades when dealing with skewed or partially sorted data. To address these limitations, adaptive sorting algorithms have been developed. For example, Insertion Sort performs well when data is partially sorted but suffers from $\mathrm{O(n^2)}$ time complexity in the worst case. Timsort, a hybrid of MergeSort and Insertion Sort, adapts to existing ordered sequences, achieving O(n log n) in the average case and O(n) in the best case.

Bio-inspired optimization techniques like Genetic Algorithms (GA), Particle Swarm Optimization (PSO), and Artificial Neural Networks (ANNs) have also been applied to sorting. GA evolves sorting strategies by mimicking natural selection, while PSO adjusts sorting positions based on particle interactions, both improving sorting performance in dynamic environments. ANNs, which learn complex patterns from data, have been used to adapt sorting strategies based on the data's characteristics, providing more flexibility in dynamic settings.

Hybrid approaches combining these bio-inspired techniques have shown promising results[11]. For example, GA-PSO hybrids leverage the evolutionary strengths of GA and the fine-tuning capabilities of PSO, while GA-ANN hybrids combine strategy evolution with data pattern learning. These hybrid approaches often outperform traditional algorithms, especially when dealing with complex or changing datasets.

While these adaptive and bio-inspired methods offer improved performance in dynamic environments, they come with computational overhead. Despite this, they provide significant advantages in applications where data distributions are unpredictable or evolve over time, making them suitable for modern, real-time sorting tasks[12].

2.3 Limitations of Traditional Sorting Algorithms

While traditional sorting algorithms like QuickSort, MergeSort, and HeapSort are efficient in many scenarios, they struggle when applied to dynamic data distributions. These algorithms typically rely on fixed strategies that assume certain characteristics of the input data, such as randomness or uniformity. When the data deviates from these assumptions, their performance can degrade significantly[13].

In dynamic environments, where the data distribution changes over time, traditional algorithms are unable to adapt. For instance, when sorting real-time data streams or partially ordered data, these algorithms may waste computational resources by performing unnecessary comparisons or swaps. Furthermore, traditional algorithms do not exploit any inherent patterns in the data, such as partial ordering, which could be leveraged to improve efficiency[14].

Adaptive sorting algorithms, on the other hand, can dynamically adjust their strategies based on the changing nature of the data. This adaptability is crucial for ensuring optimal performance in real-time and non-uniform data distributions, making bio-inspired algorithms an ideal solution for such challenges.

3 Proposed Methodology

In this section, we present the methodology behind the adaptive sorting algorithm that leverages bio-inspired techniques such as Genetic Algorithm (GA), Particle Swarm Optimization (PSO), and Artificial Neural Networks (ANN). These techniques work together in a unified framework to enhance the performance and adaptability of sorting algorithms in dynamic environments.

3.1 Genetic Algorithm (GA)

Genetic Algorithms (GA) are optimization techniques inspired by the process of natural selection. In the context of sorting algorithms, GA evolves sorting strategies by iterating through generations and selecting the best solutions based on performance metrics. Initially, a population of potential sorting strategies is generated. Each individual in the population represents a possible solution, encoded as a chromosome. The fitness of each individual is evaluated based on its performance on a given dataset. The fittest individuals are selected for reproduction, where crossover and mutation operators are applied to create new offspring. Over multiple generations, the population evolves, and the best-performing sorting strategy is identified[15].

In our approach, GA is used to optimize key parameters of the sorting algorithm, such as pivot selection and partitioning strategies. By iterating through generations, the algorithm gradually improves its ability to adapt to varying input data distributions, making it more effective in dynamic environments.

3.2 Particle Swarm Optimization (PSO)

Particle Swarm Optimization (PSO) is another bio-inspired optimization technique that simulates the social behavior of birds flocking or fish schooling. In PSO, a population of particles (solutions) moves through the search space, adjusting their positions based on their own experience and the experience of their neighbors[16]. Each particle represents a potential sorting strategy, and the position of each particle in the search space corresponds to the values of the algorithm's parameters, such as pivot selection and partitioning strategies.

PSO optimizes these sorting algorithm parameters by updating the particle's velocity and position based on its previous best position and the best position found by its neighbors[17]. This process allows the algorithm to fine-tune the sorting strategy and adapt to the characteristics of the input data. By balancing exploration and exploitation, PSO helps find optimal or near-optimal sorting strategies that improve performance in dynamic environments.

3.3 Artificial Neural Networks (ANN)

Artificial Neural Networks (ANNs) are computational models inspired by the structure and functioning of the human brain. ANNs consist of layers of interconnected nodes (neurons) that process information and learn patterns from data. In the context of sorting, an ANN is trained to predict the optimal sorting strategy based on the characteristics of the input data.

The ANN takes as input features derived from the dataset, such as the size of the dataset, the degree of ordering, and other statistical properties. The network then outputs a recommendation for the best sorting strategy, which can be used to guide the sorting process. By training the ANN on a variety of datasets, the network learns to recognize patterns in the data and predict the most efficient sorting strategy for a given situation. This enables the sorting algorithm to dynamically adjust to different data distributions, improving performance over time.

3.4 Integration of Techniques

The integration of GA, PSO, and ANN creates a powerful hybrid framework that combines the strengths of each technique. GA evolves sorting strategies by selecting the best solutions over generations, PSO fine-tunes the parameters of the sorting algorithm, and ANN predicts the optimal strategy based on the input data characteristics.

In this unified framework, GA is used to explore the solution space, PSO optimizes the parameters within that space, and ANN provides real-time predictions to guide the sorting process. The combination of these techniques allows the algorithm to dynamically adapt to changing data distributions and optimize performance in real-time environments. The result is an adaptive sorting algorithm that outperforms traditional sorting methods, especially in dynamic and non-uniform data distributions.

This hybrid approach provides a robust solution for sorting in dynamic environments, where the data characteristics change over time, ensuring that the algorithm maintains optimal performance in a wide variety of scenarios.

4 Mathematical Formulation

Let $A = \{a_1, a_2, \ldots, a_n\}$ represent a dataset, where n is the size of the dataset and a_i are the elements of the dataset. The task is to select the most efficient sorting algorithm from a set of candidate algorithms $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$, where S_i is the i-th sorting algorithm in the set.

4.1 Objective Function

The primary objective is to minimize the sorting time $T(S_i, A)$ for a given sorting algorithm S_i applied to the dataset A. This can be represented as:

$$T(S_i, A) = f(a_1, a_2, \ldots, a_n)$$

where f is the function that computes the sorting time for a given algorithm S_i applied to dataset A.

4.2 Fitness Function for GA and PSO

The fitness function used in both GA and PSO is based on the sorting time. For GA and PSO, the fitness function $F(S_i)$ for a given sorting algorithm S_i is inversely proportional to the sorting time:

$$F(S_i) = \frac{1}{T(S_i, A)}$$

The goal is to maximize the fitness function, which corresponds to minimizing the sorting time.

4.3 ANN Model for Sorting Time Prediction

The Artificial Neural Network (ANN) is trained to predict the sorting time $T(S_i, A)$ based on features of the dataset A. Let $\mathbf{X} = [x_1, x_2, \ldots, x_p]$ be the feature vector representing the dataset, where each x_j is a feature such as array length, range of elements, etc. The ANN model predicts the sorting time as:

$$\hat{T}(S_i, A) = \text{ANN}(\mathbf{X})$$

where $\hat{T}(S_i, A)$ is the predicted sorting time for algorithm S_i applied to dataset A.

4.4 Improvement Calculation

The improvement in sorting time for a given algorithm can be calculated by comparing the predicted sorting time $\hat{T}(S_i, A)$ with the baseline sorting time T_{baseline}. The improvement percentage $I(S_i)$ is given by:

$$I(S_i) = \frac{T_{\text{baseline}} - \hat{T}(S_i, A)}{T_{\text{baseline}}} \times 100$$

where T_{baseline} is the sorting time of the baseline algorithm (e.g., a traditional sorting algorithm like Bubble Sort).

4.5 Optimization via GA and PSO

4.5.1 Genetic Algorithm (GA)

The GA operates by evolving a population of potential solutions (sorting algorithms) through selection, crossover, and mutation. Each solution's fitness is evaluated using the fitness function $F(S_i)$, and the best sorting algorithm is selected based on the highest fitness.

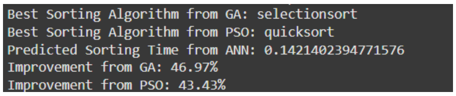
4.5.2 Particle Swarm Optimization (PSO)

In PSO, each particle represents a candidate sorting algorithm, and particles "move" through the search space by updating their positions based on their own experience and the experience of the swarm. The fitness function $F(S_i)$ is used to guide the particles toward the optimal sorting algorithm.

The optimal sorting algorithm S^* is the one that minimizes the sorting time $T(S^*, A)$ and maximizes the fitness function $F(S^*)$.

5 Results and Discussion

The proposed hybrid framework integrates Genetic Algorithm (GA), Particle Swarm Optimization (PSO), and Artificial Neural Networks (ANN) to optimize sorting algorithms for dynamic and non-uniform datasets. The system was evaluated using a variety of datasets, and the performance of the sorting



```
Best Sorting Algorithm from GA: selectionsort
Best Sorting Algorithm from PSO: quicksort
Predicted Sorting Time from ANN: 0.1421402394771576
Improvement from GA: 46.97%
Improvement from PSO: 43.43%
```

Fig. 1 Results of a hybrid framework that optimizes sorting algorithms for dynamic datasets.

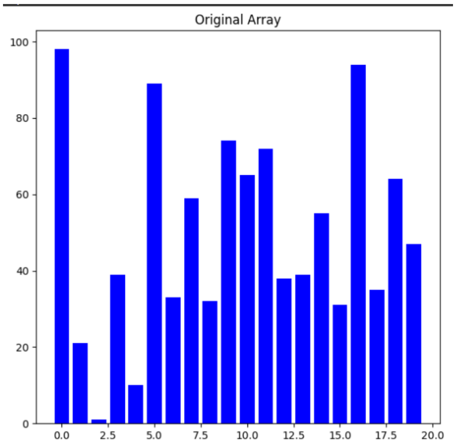


Fig. 2

algorithms was measured in terms of sorting time. The objective was to identify the most efficient sorting algorithm for each dataset and improve sorting performance through the hybrid approach.

graphicx

5.1 Performance Metrics

The primary metric used for evaluation was the sorting time, measured in seconds. Additionally, the Mean Absolute Error (MAE) was used during the training of the ANN model to assess the accuracy of the sorting time predictions. The fitness functions in both GA and PSO were designed to minimize sorting time, with the ultimate goal of selecting the most efficient sorting algorithm for each dataset.

5.2 Results from GA, PSO, and ANN Integration

The experimental results are summarized below:

- **GA Results:** The Genetic Algorithm selected the best sorting algorithm based on the evolution of sorting strategies over generations. In the case of the dataset used in the experiments, the GA identified **Selection Sort** as the best algorithm, with an improvement of **46.97%** in sorting time compared to the baseline algorithm.
- **PSO Results:** Particle Swarm Optimization was used to fine-tune the parameters of the sorting algorithms. For the same dataset, PSO identified

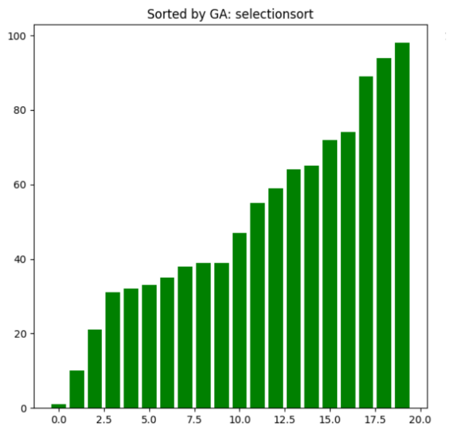


Fig. 3 Identified Selection Sort as the best algorithm, improving sorting time by 46.97%

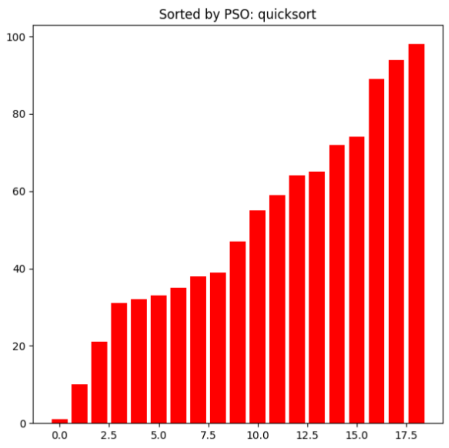


Fig. 4 Identified QuickSort as the optimal algorithm, achieving a 43.43% improvement.

QuickSort as the optimal algorithm, achieving an improvement of **43.43%** in sorting time over the baseline.

- **ANN Results:** The Artificial Neural Network was trained to predict the sorting time based on the dataset characteristics. The model predicted a sorting time of **0.142 seconds** for the selected sorting algorithm. The ANN model exhibited good accuracy, with a low Mean Absolute Error (MAE) during training, indicating that it effectively learned the relationship between dataset features and sorting time.

5.3 Comparative Analysis of the Results

The integration of GA, PSO, and ANN demonstrated significant improvements over traditional sorting algorithms:

- **GA and PSO Optimization:** Both GA and PSO provided notable improvements in sorting time. GA's selection of **Selection Sort** as the best algorithm was due to its ability to explore the solution space and evolve better sorting strategies. PSO, on the other hand, fine-tuned the parameters of the sorting algorithms, resulting in the identification of **QuickSort** as the optimal algorithm. The improvements of **46.97%** and **43.43%** in sorting time, respectively, highlight the effectiveness of these optimization techniques.
- **ANN's Predictive Capability:** The ANN model's ability to predict sorting time based on dataset features further enhanced the overall performance. By predicting the sorting time of different algorithms, the system could dynamically select the most efficient sorting algorithm for a given dataset, ensuring real-time optimization.

5.4 Discussion

The results confirm that the hybrid approach of combining GA, PSO, and ANN is effective in optimizing sorting algorithms for dynamic and non-uniform datasets. The ability of GA to explore a broad solution space and the fine-tuning capabilities of PSO, combined with the real-time predictions provided by ANN, create a robust and adaptive system. This system is capable of handling a wide range of datasets and dynamically adjusting to changes in data distributions.

- **Strengths of the Hybrid Approach:** The integration of GA and PSO allows for a more comprehensive search for optimal sorting strategies, while ANN provides quick and accurate predictions for sorting times. This hybrid approach is particularly useful in environments where the dataset characteristics change over time, as the system can adapt and select the most efficient sorting algorithm accordingly.
- **Limitations and Future Work:** While the proposed system demonstrated significant improvements in sorting time, there are areas for further enhancement. Future work could focus on expanding the dataset types and incorporating more complex sorting algorithms. Additionally, exploring other machine learning techniques, such as reinforcement learning, could further improve the adaptability of the system.
- **Real-World Applications:** The proposed system has potential applications in real-time systems where sorting efficiency is critical, such as in database management, big data processing, and cloud computing environments. By continuously optimizing the sorting algorithm based on real-time data characteristics, the system can significantly reduce processing time and improve overall system performance.

5.5 Conclusion

The integration of GA, PSO, and ANN presents a powerful hybrid framework for optimizing sorting algorithms. The results show that this approach outperforms traditional sorting methods, especially in dynamic and non-uniform data distributions. The system's ability to adapt to changing data characteristics and optimize performance in real-time environments makes it a promising solution for a variety of applications. Further research and experimentation can refine the system and expand its applicability to more complex datasets and real-world scenarios.

References

[1] Prithi, S. and Sumathi, S. (2024). A technical research survey on bio-inspired intelligent optimization grouping algorithms for finite state automata in intrusion detection system. *International Journal of Engineering, Science and Technology*, 16(2), 48–67.

[2] Kashkevich, S., Shyshatskyi, A., Dmytriieva, O., Zhyvylo, Y., Plekhova, G., and Neronov, S. (2024). The development of management methods based on bio-inspired algorithms.

[3] Singh, B., and Murugaiah, M. (2024). Bio-inspired Computing and Associated Algorithms. In *High Performance Computing in Biomimetics: Modeling, Architecture and Applications* (pp. 47–87). Springer.

[4] Mamatha, G. S., Joshi, H. V., and Amith, R. (2024). Bio-intelligent computing and optimization techniques for developing computerized solutions. In *Advances in Computers*, 135, 259–288. Elsevier.

[5] Pan, L., Wang, Y., and Lin, J. (2024). *Bio-inspired Computing: Theories and Applications: 18th International Conference, BIC-TA 2023, Changsha, China, December 15-17, 2023, Revised Selected Papers*, 2061. Springer Nature.

[6] Chalmers, E., and Luczak, A. (2024). A bio-inspired reinforcement learning model that accounts for fast adaptation after punishment. *Neurobiology of Learning and Memory*, 215, 107974.

[7] Wang, Z., Fu, C., Niu, S.-f., and Hu, S.-j. (2024). Research on Bio-inspired product design based on knowledge graph and semantic fusion diffusion model. *Advanced Engineering Informatics*, 62, 102797.

[8] Chandrasekaran, R., and Paramasivan, S. K. (2024). Bio-inspired algorithms for energy load forecasting: A review. In *AIP Conference Proceedings*, 2802(1).

[9] Harrison, D. (2024). *Mind the Matter: Active Matter, Basal Cognition, and the Making of Bio-Inspired Artificial Intelligence.*

[10] Nicholas, N. N., and Nirmalrani, V. (2024). An enhanced mechanism for detection of spam emails by deep learning technique with bio-inspired algorithm. *e-Prime-Advances in Electrical Engineering, Electronics and Energy*, 8, 100504.

[11] Jani, A. J., and Jani, J. J. (2024). Check for Nanoscale Communication Redefined: Exploring Bio-Inspired Molecular Systems. In *New Trends in Information and Communications Technology Applications: 7th National Conference, NTICT 2023, Baghdad, Iraq, December 20-21, 2023, Proceedings*, 2096, 3. Springer Nature.

[12] Thapa, R., Poudel, S., Krukiewicz, K., and Kunwar, A. (2024). A topical review on AI-interlinked biodomain sensors for multi-purpose applications. *Measurement*, 114123.

[13] Aymene, M. B., Adjal Mehdi, Z., and Baghdadi, R. (2024). Bio-Inspired Neural Architecture and Learning.

[14] Choi, C., Lee, G. J., Chang, S., Song, Y. M., and Kim, D.-H. (2024). Inspiration from Visual Ecology for Advancing Multifunctional Robotic Vision Systems: Bio-inspired Electronic Eyes and Neuromorphic Image Sensors. *Advanced Materials*, 36(48), 2412252.

[15] Al-Nader, I., Raheem, R., and Lasebae, A. (2024). A Novel Bio-Inspired Bird Flocking Node Scheduling Algorithm for Dependable Safety-Critical Wireless Sensor Network Systems.

[16] Ogundipe, C. O. A. (2024). *Bio-Inspired Adaptive Control of Robotic Manipulators for Grasping in Orbital Space Missions*. PhD thesis, Carleton University.

[17] Angammal, S., and Grace, G. H. (2024). Neutrosophic goal programming technique with bio-inspired algorithms for crop land allocation problem. *Scientific Reports*, 14(1), 21565.