



# The Conceptual Understanding of Metaheuristic Algorithms: A Brief Reviews

Widi Aribowo<sup>1\*</sup>

<sup>1</sup>Department of Electrical Engineering, Faculty of Vocational, Universitas Negeri Surabaya, Surabaya, Indonesia

## Article Info

### Article history:

Received October 10, 2025  
Revised November 10, 2025  
Accepted January 15, 2026

### Keywords:

Metaheuristic Algorithms  
Optimization  
Categories of Metaheuristics  
Population Metaheuristics  
Application of Metaheuristic

## ABSTRACT

Metaheuristic algorithms have garnered significant attention in the field of optimization due to their ability to address complex, nonlinear, and combinatorial problems where conventional exact methods are often impractical. Inspired by natural phenomena, social behaviors, and physical processes, these algorithms provide near-optimal solutions within reasonable computational time by balancing exploration and exploitation. This paper presents a comprehensive review of metaheuristic algorithms, categorizing them into single-solution-based and population-based approaches. It further discusses hybrid and adaptive variants designed to overcome limitations such as premature convergence and parameter sensitivity. The study highlights the advantages, disadvantages, and practical applications of various metaheuristics across diverse domains including engineering, logistics, artificial intelligence, energy systems, and bioinformatics offering researchers a structured guide for selecting appropriate algorithms based on problem characteristics.

*This is an open access article under the [CC BY-SA](#) license.*



## 1. INTRODUCTION

Artificial intelligence (AI) is a branch of computer science that aims to create machines or systems capable of mimicking human intelligence, such as learning, understanding language, recognizing patterns, making decisions, and solving problems. AI works through techniques such as machine learning, deep learning, natural language processing (NLP), and computer vision [1]-[6].

Metaheuristics are algorithmic approaches used to solve optimization problems, particularly complex, nonlinear, non-convex, or combinatorial ones, where exact methods (such as mathematical programming) are often impractical due to high computational requirements or the inability to find a solution in a reasonable time. Unlike exact algorithms, which guarantee optimal solutions, metaheuristics do not guarantee optimality but are designed to produce a "good enough" (near-optimal) solution within a reasonable computational time [7]-[11]. The key concept of metaheuristics lies in their ability to intelligently explore the solution space by combining global (exploration) and local (exploitation) search strategies, thereby avoiding local optimum traps and finding high-quality solutions [12][13]. Metaheuristics are general-purpose, meaning they can be applied to a wide variety of problems without the need for significant structural modifications, although their effectiveness often depends on the adjustment of parameters and strategies specific to the specific context [14]-[18].

Metaheuristics are generally divided into two main categories: population-based and single-solution-based. Population-based algorithms, such as Genetic Algorithms (GA), Particle Swarm Optimization (PSO), and Ant Colony Optimization (ACO), work by maintaining a pool of candidate solutions that evolve or interact over iterations. For example, GA mimics the process of natural selection through operators like selection, crossover, and mutation to generate increasingly better solutions. Meanwhile, PSO mimics the social behavior of birds or fish in foraging, where each "particle" updates its position based on personal and collective experience. On the other hand, single-solution-based algorithms like Simulated Annealing (SA) and Tabu Search (TS) start from a single initial solution and perform a stepwise search through its neighbors. SA mimics the cooling process of metal, allowing for the acceptance of worse solutions with a certain probability of

\*Corresponding Author

Email: [widiaribowo@unesa.ac.id](mailto:widiaribowo@unesa.ac.id)

escaping the local optimum, while TS uses short-term memory to prevent reverting to recently visited solutions [19]-[22].

A key characteristic of metaheuristics is their flexibility and adaptability. They do not rely on specific mathematical assumptions about the problem structure (such as differentiability or convexity), making them suitable for real-world problems that are often ill-structured or have many constraints. Furthermore, metaheuristics can be combined with other techniques, such as hybridization with exact methods, machine learning, or domain-specific heuristics, to improve performance [23]-[25]. This approach, known as hybrid metaheuristics, is gaining popularity in modern optimization research. For example, combining GA with local search can accelerate convergence and improve the quality of the final solution [26]-[28].

While very useful, metaheuristics have several drawbacks. First, there is no theoretical guarantee that the found solution is optimal. Second, their performance is highly sensitive to parameter settings (such as the mutation rate in GA or the initial temperature in SA), which are often determined through trial and error or automatic tuning. Third, poor algorithm design can lead to premature convergence, where the search stops too quickly on a suboptimal solution, or conversely, to excessive exploration that wastes time without significant improvement. Therefore, a thorough understanding of the algorithm's internal mechanisms and the characteristics of the problem at hand is crucial for effective application [29]-[31].

Metaheuristics and artificial intelligence (AI) are closely related because metaheuristics are often used as part of an AI approach to solve complex optimization problems. Many metaheuristic algorithms, such as Genetic Algorithms, Particle Swarm Optimization, and Ant Colony Optimization, are inspired by natural intelligence or the collective behavior of living organism and therefore fall within the AI paradigm. In practice, metaheuristics help AI find efficient solutions in large, unstructured search spaces, especially when exact methods are not feasible. Metaheuristics enhance AI's search, learning, and decision-making capabilities [32][33]. This article makes major contributions in the form of:

1. Comprehensive synthesis of the development and classification of metaheuristic algorithms, including single-solution-based, population-based approaches (evolutionary, group intelligence, social inspiration), as well as hybrid and adaptive variants.
2. Comparative analysis that maps the advantages and disadvantages of each algorithm category, including sensitivity to parameters, escalation capabilities, and robustness to local optima.
3. Mapping the practical applications of metaheuristics in various fields of science and industry, which assist researchers and practitioners in selecting the most appropriate algorithms for specific optimization challenges.
4. Identify future trends, such as integration with machine learning and development of adaptive mechanisms, that drive the use of metaheuristics as intelligent frameworks in dealing with the complexity of 21st century optimization.

This article is organized as follows. Section 2 describes the metaheuristic algorithm. Section 3 presents the results and discussion of the comparison of the metaheuristic algorithm with other methods. Section 4 contains the Conclusion

## 2. Metaheuristic Algorithms

### 2.1. The History of Metaheuristic Algorithms

The history of metaheuristic algorithms began in the mid-20th century, with the growing need to solve complex optimization problems that could not be efficiently solved using exact methods. The term "metaheuristic" itself was first introduced by Fred Glover in 1986 in the context of Tabu Search, although concepts and algorithms falling under this category had emerged much earlier [34]. The historical roots of metaheuristics can be traced back to the 1950s and 1960s, when researchers began developing rule-based approaches and intelligent search to overcome the limitations of classical optimization methods [35].

One of the most significant early milestones was the development of Simulated Annealing (SA) in the early 1980s by Scott Kirkpatrick, C. Daniel Gelatt, and Mario P. Vecchi, inspired by the physical process of annealing in metallurgy the slow cooling of molten metal to achieve a low-energy crystal structure [36]. However, the basic idea actually appeared earlier in the work of Metropolis et al. (1953), who proposed a Monte Carlo algorithm for simulating thermodynamic systems [37]. SA was one of the first metaheuristics to explicitly allow for the acceptance of worse solutions with a certain probability, thus avoiding the local optimum trap.

In the same decade, Tabu Search (TS) was developed by Fred Glover around 1986 as a memory-based approach to guide searches beyond local optima. TS introduced the concept of a tabu list—a list of temporary prohibitions against certain solutions or moves to prevent cycles and encourage exploration of a wider solution space. This approach marked a shift from blind search to one that “learns” from previous experience.

The next major development came from the world of biology and evolution. Genetic Algorithms (GAs), systematically developed by John Holland and his students at the University of Michigan in the 1970s, mimic the mechanisms of natural selection and genetics. Although the basic idea had been around in simple forms since the 1960s (for example, by John Bagley and Ingo Rechenberg), GAs became popular after the publication of Holland's book "Adaptation in Natural and Artificial Systems" (1975). GAs use operators such as selection, crossover, and mutation to evolve a population of solutions over generations, and have become the foundation for many modern evolutionary algorithms [38][39].

In the 1990s, a new wave of metaheuristics inspired by nature emerged. Ant Colony Optimization (ACO), proposed by Marco Dorigo in 1992, mimics the behavior of ant colonies in finding the shortest path to a food source through stigmergy communication (pheromone trails) [40][41]. ACO proved highly effective for combinatorial problems such as the Traveling Salesman Problem (TSP). Shortly thereafter, in the late 1990s, Particle Swarm Optimization (PSO) was developed by James Kennedy and Russell Eberhart (1995), inspired by the social behavior of flocks of birds or fish. PSO uses a simple mechanism based on particle velocity and position, guided by individual and collective experience [42][43].

Entering the 21st century, the development of metaheuristics has become increasingly rapid and diverse. Many new algorithms have emerged inspired by various natural, social, and even cultural phenomena such as the Artificial lemming algorithm [44], Starfish optimization algorithm [45], Dream Optimization Algorithm [46], Chinese Pangolin Optimizer [47], and Rabbit and Turtle Algorithm [48]. Although some of these have been criticized for lacking a strong theoretical foundation or structural similarities to previous algorithms, they reflect a creative trend in the search for more adaptive and efficient optimization approaches.

Furthermore, the modern computing era has fostered the emergence of hybrid metaheuristics, which combine two or more techniques to leverage the advantages of each. This approach often yields significantly better performance than either method alone. On the other hand, adaptive and self-adaptive metaheuristics have been developed to reduce reliance on manual parameter tuning by allowing the algorithm to dynamically adjust its strategy during the search process [49].

Historically, the evolution of metaheuristics reflects a shift from deterministic and exact approaches to stochastic, flexible, and intelligent exploration-based methods. From humble roots in physics, biology, and computer science, metaheuristics have now become a key pillar in solving real-world optimization problems in fields ranging from logistics and engineering to finance and artificial intelligence. Their development continues, with a focus on computational efficiency, scalability, and integration with cutting-edge technologies such as machine learning and cloud computing [50].

## 2.2. Categories of Metaheuristics

Metaheuristic algorithms can be categorized based on various criteria, but the most common and informative classification is based on the primary search mechanism and the working structure of the algorithm. In general, metaheuristics are divided into two broad categories: (1) Single-solution based (trajectory-based) and (2) Population-based. In addition, there are also classifications based on the source of inspiration (natural, physical, social, etc.) and the level of adaptivity (static vs. adaptive). The illustration Categories of Metaheuristics can be seen in Figure 1.

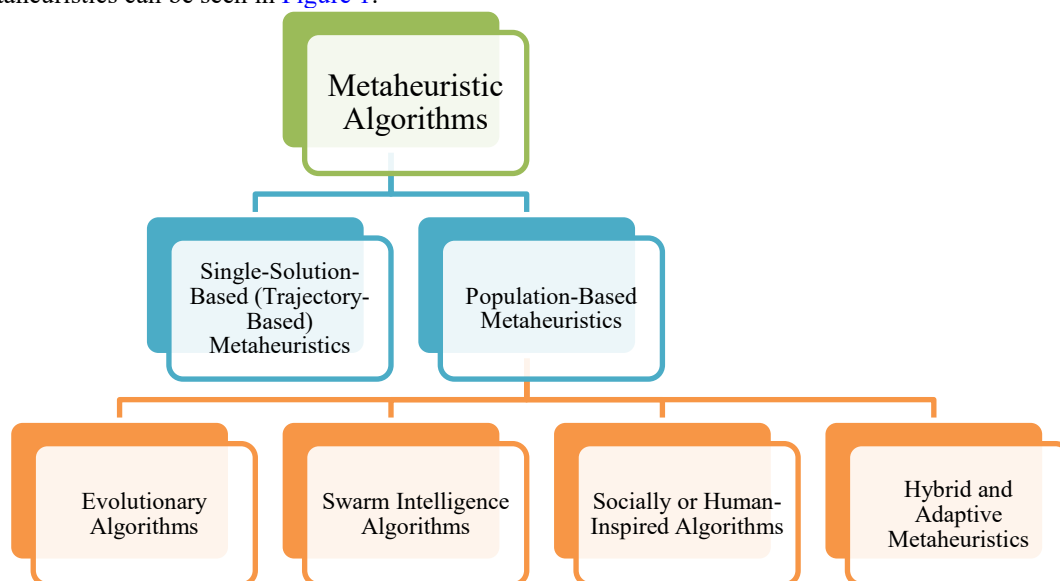


Figure 1. Categories of Metaheuristics

### 2.2.1. Single-Solution-Based (Trajectory-Based) Metaheuristics

Algorithms in this category start their search from a single initial solution and iteratively improve it by exploring its neighbors. The search process forms a "trajectory" through the solution space. Because they maintain only one solution at a time, these algorithms are generally more memory-efficient and faster per iteration, but they are prone to getting stuck in local optima without a robust exploration mechanism. Key Features:

- Focus on local exploitation with limited support for global exploration.
- Requires special strategies to avoid premature convergence.
- Often uses stochastic or memory mechanisms to escape local optima.

Example Algorithms:

- Simulated Annealing (SA): Adopts thermodynamic principles from the annealing process. Allows acceptance of worse solutions with decreasing probability over time (controlled by "temperature").
- Tabu Search (TS): Uses a memory structure (tabu list) to prohibit returning to a specific solution or move in the short term, thus encouraging diversification.
- Iterated Local Search (ILS): Combines local search with a perturbation process—after reaching a local optimum, the solution is intentionally randomized (perturbed) to initiate a new search.
- Variable Neighborhood Search (VNS): Systematically changes the structure of the search neighborhood to avoid stagnation.

### 2.2.2. Population-Based Metaheuristics

This category works with a group of candidate solutions (a population) that evolve or interact during the search process. Each individual in the population represents a potential solution, and the algorithm collectively updates the population through operations such as selection, recombination, mutation, or information exchange. This approach naturally supports global exploration due to the diversity of solutions within the population. Key Features:

- Better ability to explore the solution space due to population diversity.
- More resistant to local optima.
- Generally more computationally intensive due to maintaining multiple solutions simultaneously.

#### 2.2.2.1. Evolutionary Algorithms (EAs)

Evolutionary Algorithms (EAs) are a class of population-based metaheuristics inspired by the principles of natural biological evolution, particularly natural selection, reproduction, and genetic variation. The primary goal of EAs is to solve optimization problems—either discrete or continuous, with or without constraints—by mimicking the process of evolution: a population of candidate solutions "evolves" over generations through the mechanisms of selection, crossover, and mutation, so that the average quality of the population improves over time until it reaches a solution close to the optimal one. The primary goal of EAs is to solve optimization problems—either discrete or continuous, with or without constraints—by mimicking the process of evolution: a population of candidate solutions "evolves" over generations through selection, recombination (crossover), and mutation, so that the average quality of the population improves over time until a solution approaches the optimal one. EAs operate based on the following evolutionary analogy:

- Individual = candidate solution to an optimization problem.
- Population = a collection of individuals (usually of fixed size).
- Fitness = a measure of the quality of the solution, calculated through an objective function (the higher/lower the fitness value depends on whether the problem is maximization or minimization).
- Generation = an iteration in the evolutionary process; each generation produces a new, hopefully better population.

Despite sharing the same basic principles, there are several main variants of EAs developed for different contexts:

- Genetic Algorithm (GA)  
Developed by John Holland (1970s). Most commonly used, especially for combinatorial problems. Uses binary or symbolic representation, as well as classical crossover and mutation operators.
- Evolution Strategies (ES)

Developed in Germany (Rechenberg & Schwefel, 1960s). Focuses on continuous parameter optimization. Real vector-based representation, and Gaussian mutation is a key component. Often uses the notation ( $\mu$ ,  $\lambda$ ) or ( $\mu + \lambda$ ) for selection strategies.

- Evolutionary Programming (EP)  
Developed by Lawrence Fogel (1960s) for the evolution of finite state machines. Does not use crossover; only selection and mutation.
- Genetic Programming (GP)  
Developed by John Koza (1990s). Evolving program structures (usually in the form of trees) to solve problems such as symbolic regression, classification, or circuit design.
- Differential Evolution (DE) Developed by Storn and Price (1995). Very effective for global numerical optimization.

#### 2.2.2.2. Swarm Intelligence Algorithms

Swarm Intelligence Algorithms (SIAs) are a class of population-based metaheuristics inspired by the collective behavior of decentralized, self-organizing natural systems, such as ant colonies, bird flocks, fish schools, or bee colonies. Although individuals in these systems have limited cognitive abilities, simple local interactions among them, amplified by positive and negative feedback mechanisms, produce swarm intelligence capable of solving complex tasks such as foraging, navigation, or nest building. In computing, this principle has been adapted to solve global optimization problems. SIAs operate on three main principles:

- Decentralization: There is no central entity controlling the entire system; each agent (particle, ant, bee) makes decisions based on local information.
- Self-Organization: Global patterns emerge from local interactions between agents, without explicit planning.
- Stigmergy (in some algorithms): Indirect communication through environmental modifications (e.g., pheromone trails by ants).

Each agent in the population represents a candidate solution, and during iterations, agents update their positions based on:

- Personal experience (the best solution ever found),
- Collective experience (the best solution found by neighbors or the entire swarm),
- Stochastic motion rules that simulate exploration and exploitation.

Common Components of SIAs

- Agent population: A group of entities exploring the solution space.
- Objective function: To evaluate the quality of each agent's position (solution).
- Position update mechanism: A mathematical rule that guides the agents' movements.
- Diversification strategy: To prevent premature convergence to a local optimum (e.g., random mutation, position reset).

Some examples of algorithms inspired by the collective behavior of natural systems (insects, birds, fish, etc.) are:

- Particle Swarm Optimization (PSO): Particles move in solution space at a speed influenced by their individual best positions and the global best position.
- Ant Colony Optimization (ACO): Agents (ants) gradually build solutions using a pheromone trail that updates based on the solution's quality.
- Artificial Bee Colony (ABC): Mimics the foraging behavior of honeybees, with three types of bees: workers, observers, and scouts.
- Firefly Algorithm (FA), Cuckoo Search (CS), Bat Algorithm (BA): These fall into the more specific "nature-inspired" group, mimicking firefly light, turtledove parasitism, and bat echolocation, respectively.

#### 2.2.2.3. Socially or Human-Inspired Algorithms

Socially or Human-Inspired Algorithms are a class of metaheuristics that draw inspiration from real-life human social behavior, cultural interactions, cognitive processes, or creative activities—such as teaching, learning, collaboration, competition, music production, or even politics. Rather than mimicking nature or physics, these algorithms model human social dynamics to guide the solution-finding process in optimization. While metaphorical, this approach has proven effective because many human social activities naturally involve knowledge transfer, adaptation, healthy competition, and collaboration, which are powerful principles in the exploration and exploitation of solution spaces. Examples of Socially or Human-Inspired Algorithms. The Detail of Advantages and disadvantages of Socially or Human-Inspired Algorithms can be seen in [Table 1](#).

- Teaching-Learning-Based Optimization (TLBO)  
Inspiration: The process of teaching and learning in the classroom.
- Harmony Search (HS)  
Inspiration: The process of musical improvisation by a group of musicians.
- Imperialist Competitive Algorithm (ICA)  
Inspiration: Political dynamics and imperial expansion in world history.
- League Championship Algorithm (LCA)  
Inspiration: Sports league competitions (such as soccer).
- Socio-Cognitive Optimization (SCO):  
Inspiration: Albert Bandura's theory of social cognition.
- Cultural Algorithm (CA)  
Inspiration: The evolution of human culture.

#### General Characteristics

- Explicit knowledge transfer: Unlike nature-based algorithms, which often rely on implicit interactions (such as pheromones), social algorithms typically have explicit mechanisms for sharing information.
- Parameter-Minimum: Many algorithms, such as TLBO and HS, are designed to be free from complex parameter tuning.
- Competition-Collaboration Balance: Combines competitive pressures (to improve quality) and collaboration (to share knowledge).
- Suitable for multidimensional problems: Due to the nature of populations and global interactions.

Table 1. Advantages and disadvantages of Socially or Human-Inspired Algorithms

Aspect	Advantages	Disadvantages
Intuition and Understanding	Easy to understand because it is based on analogies to everyday human activities (teaching, playing music, competing, etc.).	Social analogies are sometimes too metaphorical and do not always reflect the unique computational mechanisms.
Algorithm Parameters	Many variants (such as TLBO, HS) are designed to be parameter-free or have only a few parameters, thus reducing the need for tuning.	Some algorithms still require parameter tuning (e.g., HMCR and PAR in HS), which can impact performance.
Ease of Implementation	The algorithm structure is generally simple and easy to code.	The lack of standardization in design makes it difficult to compare implementations across studies.
Exploration and Exploitation	Social mechanisms (such as learning from the best or competition) encourage a balance between global exploration and local exploitation.	Without additional diversification strategies, algorithms are prone to premature convergence to local optima.
Performance on Various Problems	It has proven effective on a wide range of optimization problems: engineering, finance, scheduling, and machine learning.	Performance can degrade significantly on very high-dimensional problems or those with many constraints without special modifications.
Theoretical Foundations	Some algorithms (such as TLBO) have an initial convergence analysis.	Lack of a strong theoretical foundation (e.g., convergence proofs, computational complexity) compared to classical methods like GA or PSO.
Innovation and Flexibility	It allows for the development of creative ideas through diverse socio-cultural analogies.	Many new algorithms are simply superficial modifications of existing ones (e.g., renaming "sports league" to "art competition"), without substantial improvements (over-naming).
Scalability and Efficiency	It is suitable for medium to large problems.	Computational costs can be high if not optimized, especially when the population is large or the iterations are numerous.
Hybridization	It is easy to combine with other techniques (e.g., local search, neural networks) because of its modular structure.	Poor hybridization can obscure fundamental principles and reduce interpretability.

#### 2.2.2.4. Hybrid and Adaptive Metaheuristics

Hybrid and Adaptive Metaheuristics are two advanced approaches in computational optimization developed to overcome the limitations of classical metaheuristics, such as premature convergence, slow exploration, or inability to adapt to problem characteristics. Both represent a natural evolution of metaheuristics toward more intelligent, efficient, and robust systems. The Detail of Advantages and disadvantages of Hybrid and Adaptive Metaheuristics can be seen in [Table 2](#).



Table 2. Advantages and disadvantages of Hybrid and Adaptive Metaheuristics

Aspects	Advantages of Hybrid Metaheuristics	Advantages of Adaptive Metaheuristics
Solution Quality	Produces more optimal solutions by combining the strengths of exploration (from one method) and exploitation (from another).	Dynamically improves solution quality by adjusting strategies based on current search conditions.
Convergence Speed	Accelerates convergence because complementary methods (e.g., local search) refine solutions quickly.	Avoid stagnation and accelerate convergence by switching strategies when the rate of improvement slows.
Robustness	More robust to various types of problems because the combination of methods offsets the weaknesses of each.	More robust to variations in problem characteristics (e.g., landscape shape, dimension) because it is responsive to feedback.
Handling Local Optimums	Effectively avoids local optima through diversification (from global methods) and intensification (from local methods).	Automatically increases exploration when premature convergence is detected (e.g., by increasing the mutation rate).
Design Flexibility	Can be creatively combined: metaheuristic + metaheuristic, metaheuristic + domain heuristic, or + exact method.	Flexible in adjusting parameters, operators, or strategies without changing the underlying structure of the algorithm.
Reduced Dependence on Manual Tuning	It doesn't always reduce tuning, but the right combination can make the algorithm less sensitive to a single parameter.	Significantly reduces the need for manual tuning, as parameters are adjusted during the process.
Application to Real-World Problems	Very effective for complex problems such as VRP, scheduling, engineering design, and NAS (Neural Architecture Search).	Suitable for dynamic or uncertain problems, where conditions change during optimization (e.g., real-time portfolio optimization).
Scalability	With good design (e.g., cooperative co-evolution), it can be scaled to high-dimensional problems.	Can maintain efficiency at scale by adjusting population size or search intensity.
Methodological Innovation	Encourages cross-paradigm integration (e.g., evolution + swarm + machine learning).	Encourages the development of intelligent, feedback-based mechanisms, such as fuzzy logic, reinforcement learning, or predictive models.

### 2.2.3. Application of Metaheuristic algorithms

Metaheuristic algorithms have become an important tool in solving complex optimization problems in various scientific disciplines. Because they are flexible, do not require derivatives of functions, and are able to handle large and non-linear solution spaces, metaheuristics are widely used when exact methods (such as mathematical programming) are impractical. The Application of Metaheuristic Algorithm can be seen in [Table 3](#).

Table 3. Application of Metaheuristic Algorithm

Field	Type of Problem Addressed	Used Metaheuristic Algorithm
Engineering	<ul style="list-style-type: none"> <li>Lightweight structural design</li> <li>Aerodynamics optimization</li> <li>Electronic circuit design</li> <li>Sensor placement</li> </ul>	Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Differential Evolution (DE), Harmony Search (HS), Gray Wolf Optimizer (GWO), Whale Optimization Algorithm (WOA)
Logistics & Supply Chain	<ul style="list-style-type: none"> <li>Vehicle Routing Problem (VRP)</li> <li>Production scheduling</li> <li>Inventory management</li> <li>Warehouse location (Facility Location)</li> </ul>	Ant Colony Optimization (ACO), GA, Tabu Search (TS), Simulated Annealing (SA), PSO, Iterated Local Search (ILS)
Artificial Intelligence & Machine Learning	<ul style="list-style-type: none"> <li>Feature selection</li> <li>Tuning hyperparameters</li> <li>Neural Architecture Search (NAS)</li> <li>Neural network training (Neuroevolution)</li> </ul>	PSO, GA, DE, Artificial Bee Colony (ABC), Cuckoo Search (CS), Firefly Algorithm (FA), Non-dominated Sorting Genetic Algorithm (NSGA-II)
Energy & Electric Power Systems	<ul style="list-style-type: none"> <li>Economic Load Dispatch (ELD)</li> <li>Placement of wind/solar turbines</li> <li>Optimal power flow</li> <li>Smart grid management</li> </ul>	DE, PSO, GWO, Whale Optimization Algorithm (WOA), MFO (Moth-Flame Optimizer), Hybrid PSO-GA
Finance & Economics	<ul style="list-style-type: none"> <li>Multi-purpose portfolio optimization</li> <li>Stock price predictions</li> <li>Risk management</li> <li>Algorithmic arbitrage</li> </ul>	NSGA-II, NSGA-III, PSO, GA, SA, MOEA/D

Bioinformatics & Medicine	<ul style="list-style-type: none"> <li>• Protein folding</li> <li>• DNA/RNA sequence alignment</li> <li>• Diagnosis of diseases (medical classification)</li> <li>• Medical image segmentation</li> </ul>	GA, ACO, PSO, ABC, DE
Telecommunications & Networks	<ul style="list-style-type: none"> <li>• Network routing</li> <li>• Frequency/spectrum allocation</li> <li>• Base station placement</li> <li>• Load balancing</li> </ul>	ACO, PSO, GA, FA, WOA
Agriculture & Environment	<ul style="list-style-type: none"> <li>• Irrigation optimization</li> <li>• Crop rotation</li> <li>• Calibration of climate/hydrological models</li> <li>• Natural resource management</li> </ul>	DE, GA, SA, PSO
Robotics & Autonomous Systems	<ul style="list-style-type: none"> <li>• Path planning (trajectory planning)</li> <li>• Adaptive PID control</li> <li>• Robot swarm coordination</li> <li>• SLAM optimization</li> </ul>	PSO, GA, ACO, GWO
Art, Design & Creative	<ul style="list-style-type: none"> <li>• Generative art/music</li> <li>• Fashion designs &amp; fabric patterns</li> <li>• Optimization of ad layouts</li> <li>• Game AI (NPC strategy)</li> </ul>	GP, GA, HS, PSO

### 3. DISCUSSION

The development of metaheuristic algorithms over the past few decades reflects ongoing efforts to strengthen their advantages while addressing their shortcomings. Initially, metaheuristic algorithms offered flexible solutions to complex optimization problems that cannot be solved by exact methods, thanks to their ability to handle large, non-linear and non-differentiable solution spaces without requiring gradient information. However, major drawbacks such as the absence of optimality guarantees, dependence on parameter tuning, risk of premature convergence, and variability of results due to stochastic nature are the main drivers of innovation. The advantages and disadvantages of the metaheuristic algorithm can be seen in [Table 4](#).

Table 4. Advantages and disadvantages of Metaheuristics Algorithm

Comparative Aspect	Advantages of Metaheuristics	Disadvantages of Metaheuristics
Optimality Guarantee	Does not guarantee an optimal solution	Exact methods (such as Branch-and-Bound, ILP) guarantee optimal solutions if given enough time.
Ability to Handle Complex Problems	Excellent for non-linear problems	Execution methods often fail or take a very long time on big/real problems.
Gradient Information Needs	Requires no derivative or gradient information" — suitable for black-box functions	Gradient-based methods (such as Gradient Descent, Newton) fail if the function is not differentiable or not smooth.
Flexibility & Generalization	General-purpose: can be applied to a wide range of domains without major structural modifications.	Domain-specific heuristics (e.g. 2-opt for TSP) are faster and more accurate for specific problems, but cannot be generalized.
Computation Time	Faster than exact methods on large problems, but slower than simple heuristics (such as greedy).	Slower than constructive heuristics (such as nearest neighbor) or one-time learning methods (such as neural solvers after training).
Parameter Tuning Requirement	Many metaheuristics require parameter tuning" (mutation rate	Exact methods (such as CPLEX) and some heuristics (such as greedy) do not have parameters that need to be tuned.
Scalability	It can be scaled to large problems with techniques such as decomposition or parallelization, but performance can degrade at very high dimensions (>10,000 variables).	Learning-based methods (such as GNN for TSP) can provide instant solutions after training, are highly scalable for inference.
Robustness to Noise & Uncertainty	Very robust to noisy or stochastic objective functions due to its stochastic nature.	Exact and gradient-based methods are very sensitive to noise.



Interpretability & Theoretical Analysis	Difficult to analyze mathematically; few theoretical guarantees about convergence or complexity.	Exact and gradient methods have a strong theoretical basis (convergence, complexity, optimality).
Ease of Implementation	Relatively easy to implement for new problems	Modern hybrid or adaptive metaheuristics can be very complex.
Solution Quality vs. Product Quality Time	Provides high-quality solutions in reasonable time to NP-hard problems that cannot be solved exactly.	Inconsistent: solution quality can vary between runs due to stochastic nature (except with fixed seeds).

In its development, adaptive metaheuristics emerged which dynamically adjust parameters during search to reduce dependence on manual tuning, as well as hybrid metaheuristics which combine global exploration with local exploitation to improve solution quality and speed up convergence. Additionally, integration with modern techniques has extended the scalability of metaheuristics to problems of very high dimensions and expensive objective functions. Although theoretical challenges such as the lack of formal convergence analysis still exist, the direction of development now emphasizes robustness, computational efficiency, and real-world applicability, making metaheuristics not just a heuristic tool, but an intelligent framework that continues to evolve to deal with the complexity of 21st century optimization. A comparison of the metaheuristic algorithm with other methods can be seen in Table 5.

Table 5. Comparative study of Metaheuristics Algorithm with Other Methods

Compared with	Metaheuristics Are Less Superior If	Metaheuristics Are Less Superior If
Exact Method	An optimal solution is absolutely necessary and the problem is small/medium.	An optimal solution is absolutely necessary and the problem is small/medium.
Simple Heuristics	Execution time is critical and a “good enough” solution is sufficient.	Execution time is critical and a “good enough” solution is sufficient.
ML Based Methods (e.g., GNN)	A lot of historical data is available; need ultra-fast inference after training.	A lot of historical data is available; need ultra-fast inference after training.

#### 4. CONCLUSION AND LIMITATION

Metaheuristic algorithms have become an important pillar in solving various complex, non-linear, and high-dimensional real-world optimization problems. While it does not guarantee a theoretically optimal solution, its flexibility, ability to handle black-box functions, as well as capacity to produce high-quality solutions in reasonable computing time make it the preferred choice when exact methods are not feasible. Recent developments such as hybrid and adaptive approaches have significantly improved the performance, convergence speed, and robustness of algorithms to local optimum traps. However, challenges such as lack of a strong theoretical basis, reliance on parameter tuning, and variability of results due to stochastic nature still need to be addressed. In the future, the integration of metaheuristics with cutting-edge technologies such as machine learning and cloud computing will expand their scalability and relevance in dealing with dynamic and very large-dimensional optimization problems.

#### ACKNOWLEDGEMENTS

The work in this manuscript did not receive any research funding/grant

#### REFERENCES




- [1] Y. Cao, Y. Chen, X. Fan, H. Fu, and B. Xu, “Advanced Design for High-Performance and AI Chips,” *Nano-Micro Letters*, vol. 18, no. 1, 2026. <https://doi.org/10.1007/s40820-025-01850-w>.
- [2] M. Bertl, S. Price, and D. Draheim, “Transforming legal texts into computational logic: Enhancing next generation public sector automation through explainable AI decision support,” *International Journal of Cognitive Computing in Engineering*, vol. 7, no. 1, pp. 40–57, 2026. <https://doi.org/10.1016/j.ijcce.2025.07.003>.
- [3] J. B. M. D. Nóbrega, C. L. V. Gusmão, I. C. C. Laureano, and B. M. Santiago, “ChatGPT® and Knowledge of Brazilian Dental Ethics and Legislation,” *Pesqui. Bras. Odontopediatria Clin. Integr.*, vol. 26, 2026. <https://doi.org/10.1590/pboci.2026.010>.
- [4] M. Sagredo-Gallardo, J. González Campos, C. Alfaro Contreras, and M. Elias, “Challenges and Opportunities of Artificial Intelligence in Collaborative Learning: Implications for Educational Innovation in Institutional Contexts,” *Europe Public Social Innovation Review*, vol. 11, 2026. <https://doi.org/10.31637/epsir-2026-2211>.
- [5] V. G. Pineda, A. Valencia-Arias, F. E. L. Giraldo, and E. A. Zapata-Ochoa, “Integrating artificial intelligence and quantum computing: A systematic literature review of features and applications,” *International Journal of Cognitive Computing in Engineering*, vol. 7, pp. 26–39, 2026. <https://doi.org/10.1016/j.ijcce.2025.08.002>.
- [6] W. Aribowo and H. A. Shehadeh, “A Comparative Study of Metaheuristic Optimization Algorithms in Solving Engineering Designing Problems,” *Journal of Robotic Control*, vol. 6, no. 4, pp. 1885–1898, 2025. <https://doi.org/10.18196/jrc.v6i4.26410>

- [7] A. Nourmohammadzadeh and S. Voß, "A matheuristic approach for the robust coloured travelling salesman problem with multiple depots," *European Journal of Operational Research*, vol. 328, no. 2, pp. 390–406, 2026. <https://doi.org/10.1016/j.ejor.2025.06.018>
- [8] X. Xu, "AI optimization algorithms enhance higher education management and personalized teaching through empirical analysis," *Scientific Reports*, vol. 15, no. 1, 2025. <https://doi.org/10.1038/s41598-025-94481-5>
- [9] Y. Liu, Y. Tang, and C. Hua, "Hybrid nutcracker optimization algorithm for multi-objective energy scheduling in grid-connected microgrid systems," *Journal of Computational Science*, vol. 92, 2025. <https://doi.org/10.1016/j.jocs.2025.102716>
- [10] X. Cai, W. Wang, and Y. Wang, "Multi-strategy enterprise development optimizer for numerical optimization and constrained problems," *Scientific Reports*, vol. 15, no. 1, 2025. <https://doi.org/10.1038/s41598-025-93754-3>
- [11] K. Khlie, A. Pugalenth, Z. Benmamoun, W. Aribowo, and M. Dehghani, "Sustainable Supply Chain Optimization: A Breakthrough in Swarm-based Artificial Intelligence," *Engineering, Technology & Applied Science Research*, vol. 15, no. 3, pp. 23125–23132, 2025. <https://doi.org/10.48084/etasr.10505>
- [12] T. Hamadneh *et al.*, "Motorbike Courier Optimization: A Novel Parameter-Free Metaheuristic for Solving Constrained Real-World Optimization Problems," *Int. J. Intell. Eng. Syst.*, vol. 18, no. 5, pp. 382–393, 2025. <https://doi.org/10.22266/ijies2025.0630.27>
- [13] T. Hamadneh *et al.*, "Program Manager Optimization Algorithm: A New Method for Engineering Applications," *International Journal of Intelligent Engineering and Systems*, vol. 18, no. 7, pp. 746–756, 2025. <https://doi.org/10.22266/ijies2025.0831.47>
- [14] H. Y. Jeong and B. D. Song, "Meta-learning-based adaptive operator selection for traveling salesman problem," *Applied Soft Computing*, vol. 185, 2025. <https://doi.org/10.1016/j.asoc.2025.113930>
- [15] S. Biswas *et al.*, "A Novel Hybrid Optimizer Based on Coati Optimization Algorithm and Differential Evolution for Global Optimization and Constrained Engineering Problems," *International Journal of Computational Intelligence Systems*, vol. 18, no. 1, 2025. <https://doi.org/10.1007/s44196-025-00855-y>
- [16] S. Atta, V. Basto-Fernandes, and M. Emmerich, "A Concise Review of Home Health Care Routing and Scheduling Problem," *Operations Research Perspectives*, vol. 15, 2025. <https://doi.org/10.1016/j.orp.2025.100347>
- [17] J. Guo, Z. Hu, B. Tian, and J. Wei, "Modeling and optimizing routing problems with customer satisfaction under stochastic travel times," *Transportation Research Part E: Logistics and Transportation Review*, vol. 204, 2025. <https://doi.org/10.1016/j.tre.2025.104413>
- [18] A. Susanti *et al.*, "Application of the Orangutan Optimization Algorithm for Solving Vehicle Routing Problems in Sustainable Transportation Systems," *Engineering, Technology & Applied Science Research* vol. 15, no. 3, pp. 22915–22922, 2025. <https://doi.org/10.48084/etasr.10545>
- [19] M. A. Ferradji and R. Seghir, "A novel metaheuristic global optimisation method based on grey wolf optimiser and salp swarm algorithm," *Journal of Experimental & Theoretical Artificial Intelligence*, pp. 1–37, 2025. <https://doi.org/10.1080/0952813X.2025.2515578>
- [20] M. F. Demiral, "An artificial intelligence technique: experimental analysis of population-based physarum-energy optimization algorithm," *Discover Artificial Intelligence*, vol. 5, no. 1, p. 115, 2025. <https://doi.org/10.1007/s44163-025-00367-w>
- [21] S. Yadav *et al.*, "Optimising parent selection in plant breeding: comparing metaheuristic algorithms for genotype building," *Theor. Appl. Genet.*, vol. 138, no. 9, pp. 1–20, 2025. <https://doi.org/10.1007/s00122-025-05028-1>
- [22] M. Shabani, M. Kadoch, and S. Mirjalili, "A novel metaheuristic-based approach for prediction of corrosion characteristics in offshore pipelines," *Engineering Failure Analysis*, vol. 170, p. 109231, 2025. <https://doi.org/10.1016/j.engfailanal.2024.109231>
- [23] A. H. Rabie, S. Elghamrawy, and A. E. Hassanien, "An improved Sinh Cosh optimizer for optimizing energy management system in nano-grids," *Scientific Reports*, vol. 15, no. 1, 2025. <https://doi.org/10.1038/s41598-025-16955-w>
- [24] A. Koulali, P. Radomski, P. Ziółkowski, F. Petronella, L. De Sio, and D. Mikieliewicz, "Differential evolution-optimized gold nanorods for enhanced photothermal conversion," *Scientific Reports*, vol. 15, no. 1, 2025. <https://doi.org/10.1038/s41598-025-92007-7>
- [25] L. Yuan, H. Chen, T. Chang, and G. Gong, "Optimizing performance of WPCN based on whale optimization algorithm," *Scientific Reports*, vol. 15, no. 1, 2025. <https://doi.org/10.1038/s41598-025-03636-x>
- [26] D. Kim, I. N. M. D. Chan, T. M. S. Manalastas, R. S. Concepcion II, J. R. H. Sta. Agueda, and R. K. B. Bitangcor, "Optimization of glutaraldehyde concentration in relation to swelling behavior of PVA-PEG-BTB film using hybrid genetic metaheuristic algorithm," *Scientific Reports*, vol. 15, no. 1, 2025. <https://doi.org/10.1038/s41598-025-96953-0>
- [27] A. M. Eltamaly and Z. A. Almutairi, "A novel star-nosed mole optimization algorithm applied for MPPT of PV systems," *Scientific Reports*, vol. 15, no. 1, 2025. <https://doi.org/10.1038/s41598-025-02938-4>
- [28] P. Hu and A. Ukil, "A novel auxin and wither mechanism combination optimization algorithm for maximum power point tracking of PV systems under partial shading," *Renewable Energy*, vol. 256, 2026. <https://doi.org/10.1016/j.renene.2025.123911>
- [29] Y. Raslan, M. Asiri, A. M. Maklad, and A. Fahim, "Prognosis models for nasopharyngeal carcinoma recurrences by using tabu search algorithm," *Computational Biology and Chemistry*, vol. 120, 2026. <https://doi.org/10.1016/j.compbiolchem.2025.108687>
- [30] H. Miao *et al.*, "New kiwifruit quality grading methods based on image multi-feature fusion and ResTNet model," *Expert Systems with Applications*, vol. 297, 2026. <https://doi.org/10.1016/j.eswa.2025.129507>

- [31] K. Dönmez, M. Bakır, and R. K. Cecen, "A comprehensive data-driven MCDM approach to determine the best single objective function for the aircraft sequencing and scheduling problem," *Expert Systems with Applications*, vol. 296, 2026. <https://doi.org/10.1016/j.eswa.2025.129172>.
- [32] W. Tang *et al.*, "A new method combining deep learning with meta-learning for tool-workpiece contact detection in electrochemical discharge machining," *Measurement*, vol. 257, 2026. <https://doi.org/10.1016/j.measurement.2025.118712>.
- [33] O. Bassik *et al.*, "Robust parameter estimation for rational ordinary differential equations," *Applied Mathematics and Computation*, vol. 509, 2026. <https://doi.org/10.1016/j.amc.2025.129638>.
- [34] F. Glover, "Future paths for integer programming and links to artificial intelligence," *Computers & Operations Research*, vol. 13, no. 5, pp. 533–549, 1986. [https://doi.org/10.1016/0305-0548\(86\)90048-1](https://doi.org/10.1016/0305-0548(86)90048-1)
- [35] M. Gendreau and J.-Y. Potvin, *Handbook of metaheuristics*, vol. 2. Springer, 2010.
- [36] S. Kirkpatrick, C. D. Gelatt Jr, and M. P. Vecchi, "Optimization by simulated annealing," *Science (80-. )*, vol. 220, no. 4598, pp. 671–680, 1983. <https://doi.org/10.1126/science.220.4598.671>
- [37] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *Journal of Chemical Physics*, vol. 21, no. 6, pp. 1087–1092, 1953. <https://doi.org/10.1063/1.1699114>
- [38] D. E. Goldberg, "Genetic algorithms in search, optimization, and machine learning," *Addion wesley*, vol. 1989, no. 102, p. 36, 1989.
- [39] A. E. Eiben and J. E. Smith, *Introduction to evolutionary computing*. Springer, 2015.
- [40] M. Dorigo, V. Maniezzo, and A. Colomi, "Ant system: optimization by a colony of cooperating agents," *IEEE Trans. Syst. Man, Cybern. Part B*, vol. 26, no. 1, pp. 29–41, 1996.
- [41] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 1, no. 4, pp. 28–39, 2007. <https://doi.org/10.1109/3477.484436>
- [42] J. Kennedy, "Swarm intelligence," *Handbook of nature-inspired and innovative computing: integrating classical models with emerging technologies*, Springer, 2006, pp. 187–219.
- [43] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," *MHS'95. Proceedings of the sixth international symposium on micro machine and human science*, Ieee, 1995, pp. 39–43. <https://doi.org/10.1109/MHS.1995.494215>
- [44] Y. Xiao, H. Cui, R. A. Khurma, and P. A. Castillo, "Artificial lemming algorithm: a novel bionic meta-heuristic technique for solving real-world engineering optimization problems," *Artificial Intelligence Review*, vol. 58, no. 3, 2025. <https://doi.org/10.1007/s10462-024-11023-7>.
- [45] C. Zhong, G. Li, Z. Meng, H. Li, A. R. Yildiz, and S. Mirjalili, "Starfish optimization algorithm (SFOA): a bio-inspired metaheuristic algorithm for global optimization compared with 100 optimizers," *Neural Computing and Applications*, vol. 37, no. 5, pp. 3641–3683, 2025. <https://doi.org/10.1007/s00521-024-10694-1>.
- [46] Y. Lang and Y. Gao, "Dream Optimization Algorithm (DOA): A novel metaheuristic optimization algorithm inspired by human dreams and its applications to real-world engineering problems," *Computer Methods in Applied Mechanics and Engineering*, vol. 436, 2025. <https://doi.org/10.1016/j.cma.2024.117718>.
- [47] Z. Guo, G. Liu, and F. Jiang, "Chinese Pangolin Optimizer: a novel bio-inspired metaheuristic for solving optimization problems," *The Journal of Supercomputing*, vol. 81, no. 4, 2025. <https://doi.org/10.1007/s11227-025-07004-4>.
- [48] T. Hamadneh *et al.*, "Rabbit and Turtle Algorithm: A Novel Metaheuristic for Solving Complex Engineering Optimization Problems," *International Journal of Intelligent Engineering and Systems*, vol. 18, no. 6, pp. 426–438, 2025. <https://doi.org/10.22266/ijies2025.0731.27>.
- [49] R. R. Corsini, V. Fichera, L. Longo, and G. Oriti, "A self-adaptive metaheuristic to minimize the total weighted tardiness for a single-machine scheduling problem with flexible and variable maintenance," *Journal of Industrial and Production Engineering*, vol. 42, no. 4, pp. 422–439, 2025. <https://doi.org/10.1080/21681015.2024.2429572>
- [50] J. Almeida, J. Soares, F. Lezama, S. Limmer, T. Rodemann, and Z. Vale, "A systematic review of explainability in computational intelligence for optimization," *Computer Science Review* vol. 57, p. 100764, 2025. <https://doi.org/10.1016/j.cosrev.2025.100764>.

## BIOGRAPHIES OF AUTHOR



**Widi Aribowo**    is a lecturer at the D4 Electrical Engineering Study Program, Vocational Faculty, Surabaya State University, Indonesia. The author received a Bachelor of Engineering (S.T.) degree from the Sepuluh Nopember Institute of Technology (ITS) in the field of Power Engineering, Surabaya in 2005. The author received a Master of Engineering (M.T.) degree from the Sepuluh Nopember Institute of Technology (ITS) in the field of Power Systems Engineering, Surabaya in 2009. The author also received a Doctorate (Dr.) from the Sepuluh Nopember Institute of Technology (ITS) in the field of Power Systems Engineering, Surabaya in 2009. 2025. The author researches in the field of systems and electric power control, Microgrids, and Artificial Intelligence. The author can be contacted via email: [widiaribowo@unesa.ac.id](mailto:widiaribowo@unesa.ac.id).