Received: 2 November 2025 | Revised: 28 July 2026 | Accepted: 28 August 2026 | Published online: 12 November 2026

REVIEW

Journal of Data Science and Intelligent Systems
2026, Vol. 00(00) 1–25
DOI: 10.47852/bonviewJDSIS62028069

BON VIEW PUBLISHING

Advances in Metaheuristics and Learnheuristics (2019–2026): A Comprehensive Review, Taxonomy, and Future Directions

Tansel Dokeroglu[1] **and Deniz Canturk**[1,*]

[1] *Software Engineering Department, TED University, Türkiye*

Abstract: This review examines more than 150 recent metaheuristics introduced between 2019 and 2026, emphasizing their significant impact and performance in solving complex optimization problems. In the past three decades, more than 500 metaheuristic algorithms have been proposed, with no sign of deceleration, resulting in a vast and growing landscape that makes it increasingly difficult to select, compare, and evaluate the most suitable methods for specific challenges. Our analysis considers several important aspects, including citation trends, application areas, the availability of source code, parameter configuration, the originality of proposed mechanisms and operators, and the ability of these methods to address common challenges such as stagnation and premature convergence. We also examine the practical applications and performance of the 15 most influential metaheuristic algorithms proposed over the past six years. Particular attention is given to learnheuristics, an emerging research area that combines machine learning with metaheuristic optimization to address dynamic and data-driven problems. To provide a clearer view of this growing field, we introduce a unified taxonomy covering both metaheuristics and learnheuristics. The taxonomy classifies these methods according to the level of learning integration, problem type, learning approach, and intended purpose. We believe that this classification can help researchers better understand the current state of the field and identify potential opportunities for future research.

Keywords: metaheuristic, learnheuristics, optimization, taxonomy

1. Introduction

Metaheuristics are general-purpose optimization approaches that guide the search for high-quality solutions without relying on problem-specific assumptions. Most of these methods are inspired by natural, biological, or physical phenomena and seek to achieve a suitable balance between exploration and exploitation of the search space. Although they cannot guarantee global optimality, metaheuristics have proved to be effective for many difficult optimization problems, particularly those that are large-scale, nonlinear, or NP-hard, where exact methods may become computationally impractical.

The field of metaheuristic optimization has grown rapidly over the past decade, with a large number of new algorithms being proposed and an increasing interest in combining metaheuristic search with machine learning (ML) techniques [1–3]. This rapid growth, however, has made it increasingly difficult to assess which newly introduced methods have had a meaningful impact on the field. To address this issue, we conducted a comprehensive review of metaheuristics proposed during the last six years, with particular emphasis on their influence, practical relevance, and methodological contributions. The reviewed algorithms were examined from several perspectives, including citation impact, the range of problem types addressed, availability of source code, parameter tuning requirements, originality of the underlying mechanisms and operators, and their ability to mitigate well-known challenges such as stagnation and premature convergence.

Among the 158 algorithms examined, 15 were identified as particularly influential based on the criteria considered in this study. These algorithms represent a diverse set of approaches and introduce different mechanisms for improving the search process and overcoming persistent limitations of metaheuristic optimization. Rather than focusing solely on reported performance, our review examines the similarities and differences among these methods, their main strengths and limitations, and the types of problems to which they have been applied. We also discuss their practical use in real-world applications and consider the extent to which the proposed mechanisms contribute to their effectiveness. Through this analysis, we aim to provide a clearer picture of recent developments in metaheuristic optimization and to identify research directions that may be of particular interest to the community.

Learnheuristics are hybrid optimization methods that incorporate ML techniques into metaheuristic algorithms to improve the way complex optimization problems are solved [4]. In

***Corresponding author:** Deniz Canturk, Software Engineering Department, TED University, Türkiye. Email: deniz.canturk@tedu.edu.tr

© The Author(s) 2026. Published by BON VIEW PUBLISHING PTE. LTD. This is an open access article under the CC BY License (https://creativecommons.org/licenses/by/4.0/).

conventional metaheuristics, the search process is generally controlled by predefined rules, operators, and parameter settings. Learnheuristics, in contrast, allow parts of this process to be informed by data and updated as the optimization progresses. This provides an opportunity to make the search more responsive to the characteristics of the problem and to changes that may occur during optimization.

The role of ML within a learnheuristic can vary considerably depending on the problem and the design of the algorithm. For instance, a learning model may be used to identify promising regions of the search space, select or adapt algorithmic parameters, predict the quality of candidate solutions, or approximate constraints and objective functions. In dynamic combinatorial optimization, this data-driven component can be particularly useful when problem conditions change over time. Changes in demand, resource availability, or other input parameters may require the optimization process to adjust its search strategy continuously rather than relying on a fixed configuration.

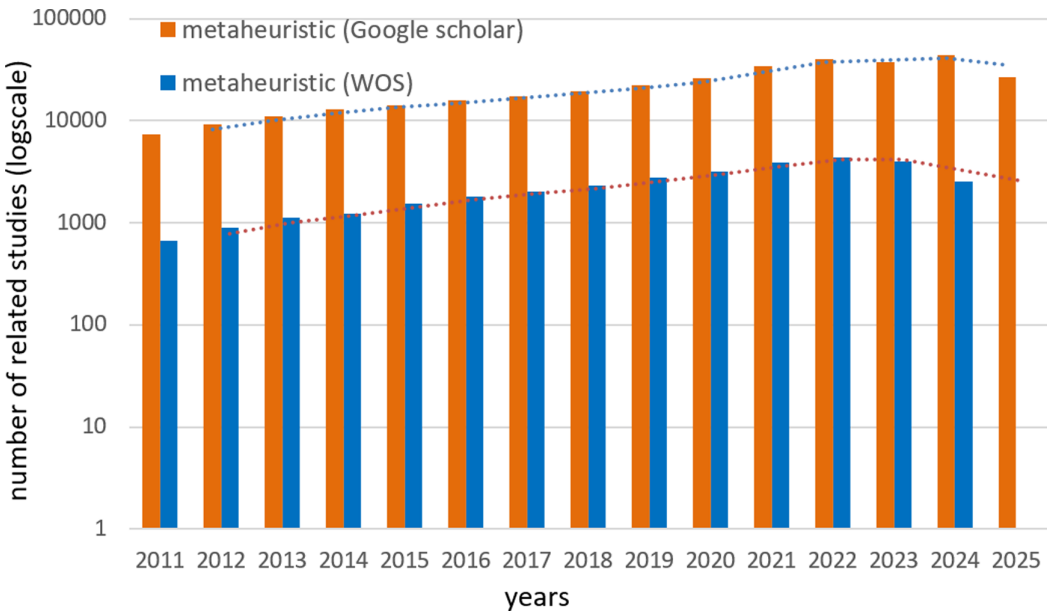
The growing interest in learnheuristics is largely driven by the nature of contemporary optimization problems. In many practical settings, the available information is incomplete, uncertain, or continuously changing, which limits the effectiveness of optimization methods designed for static environments. By combining the adaptive capabilities of ML with the search mechanisms of metaheuristics, learnheuristics provide a possible framework for dealing with such settings. Their potential has attracted attention in a variety of application areas, including logistics, financial decision-making, scheduling, and intelligent systems. Nevertheless, the effectiveness of a learnheuristic depends strongly on how learning is integrated into the optimization process and on the availability and quality of the data used for learning.

An important distinction in this context concerns the timing of the learning process. In offline learning, the ML component is trained before the optimization begins, typically using historical or previously collected data. This strategy can work well when the problem characteristics remain relatively stable. However, its effectiveness may decrease when the underlying data distribution, constraints, or problem parameters change over time. A model trained on historical data may then encounter situations that were not represented in the training set and may consequently provide less reliable guidance during optimization.

Online learning offers an alternative by allowing the learning component to update its knowledge as new information becomes available during the optimization process. This can be particularly useful for dynamic optimization problems, where the conditions of the problem may change continuously. Between these two approaches, hybrid learning strategies provide another option by combining an initial offline training phase with subsequent online adaptation. Such a combination can reduce the computational burden associated with learning from scratch while still allowing the model to respond to changes in the problem environment. These different learning strategies highlight an important aspect of learnheuristic design: the effectiveness of the approach depends not only on the choice of the ML model or metaheuristic but also on when and how learning is incorporated into the optimization process.

Figure 1 shows the number of publications retrieved from Google Scholar and Web of Science (WoS) using the keyword "metaheuristic" over the 15-year period from 2011 to 2026. The results indicate a sustained and growing interest in metaheuristic optimization, reflected in the increasing volume of research published over the period considered. The continued emergence of new algorithms and algorithmic variants suggests that research in this area remains highly active. This trend may be attributed, in part, to the broad range of optimization problems addressed by metaheuristics and their applicability across different domains. As new and increasingly complex optimization challenges arise, researchers continue to investigate alternative search strategies and develop new variants to improve solution quality, computational efficiency, and adaptability. The observed publication trend therefore indicates that metaheuristic optimization remains an active research area, with continued methodological development and practical applications expected in the coming years. Moving forward, advancements are anticipated to further enhance research

Figure 1
Results of literature queries conducted on Google Scholar and Web of Science (WoS), showing the distribution of publications related to metaheuristic and learnheuristic optimization algorithms over time



outcomes, fostering improvements in both efficiency and effectiveness for complex problem-solving. Table 1 presents the most cited and influential metaheuristic algorithms selected for this review paper, showcasing those with substantial impact in the field. Table A1 introduces a second group of notable new metaheuristics. Beyond these, numerous additional articles on emerging metaheuristic methods were published within the same period but could not be included in our study.

Figure 2 is the Word cloud generated from the most frequently occurring terms in the selected metaheuristic optimization literature, based on publications collected from major academic databases over the considered time period.

A new taxonomy for integrating metaheuristics and learning heuristics—also known as learnheuristics—can be structured around five key dimensions (see Figure 3). First, the role of learning in the algorithm can be classified as offline learning, where ML models are trained before optimization begins; online learning, where models adapt dynamically during the optimization process; or hybrid learning, which combines both approaches. Second, the level of integration between ML and the metaheuristic framework can vary. At the low level, ML may assist in parameter tuning or local search. Mid-level integration involves influencing selection, crossover, or solution evaluation. High-level integration allows ML to guide overall strategies, such as dynamic decision-making or problem reformulation.

The third dimension of the taxonomy concerns the characteristics of the optimization problem, including its temporal behavior (static or dynamic), objective structure (single-objective or multi-objective), and degree of complexity (e.g., deterministic or stochastic). The fourth dimension focuses on the type of learning method incorporated into the optimization process, such as supervised learning, unsupervised learning, reinforcement learning, and evolutionary learning. Finally, the taxonomy considers the role of the ML component within the optimization framework. Depending on the specific method, ML may be used for prediction, such as estimating objective or fitness values; classification; control, including adaptive parameter adjustment; or data generation, such as modeling or simulating changing problem inputs. Together, these dimensions provide a structured view of the different ways in which ML can be incorporated into metaheuristic optimization.

The proposed taxonomy is multidimensional rather than mutually exclusive. A learnheuristic may therefore belong to more than one category across different dimensions, depending on how learning is incorporated and the characteristics of the optimization problem. This flexibility is important because the integration of ML and metaheuristics can take many forms, and imposing strict boundaries between categories may not adequately reflect the diversity of existing approaches.

The taxonomy should also be viewed as a conceptual framework rather than a rigid hierarchical classification. Although Figure 3 is presented in a structure resembling an organizational chart, its primary purpose is to illustrate the main dimensions along which learnheuristics can differ, including the level of learning integration, the learning method employed, the role of ML, and the characteristics of the optimization problem. These dimensions may overlap in practice, and a single optimization method can simultaneously incorporate multiple learning strategies or serve more than one purpose. The proposed framework is therefore intended to provide a flexible basis for organizing and comparing learnheuristics rather than establishing strict classification boundaries.

To the best of our knowledge, existing surveys have generally examined metaheuristics from broader perspectives or focused on specific algorithm families, while a detailed analysis of the most influential recently proposed algorithms remains limited. In this study, we address this gap by examining 15 recently introduced metaheuristics in greater depth and evaluating them from multiple perspectives, including their research impact, methodological contributions, practical applications, and ability to address common limitations of metaheuristic optimization. This analysis provides a more focused view of recent developments in the field and offers insights into the strengths, limitations, and future potential of these emerging approaches.

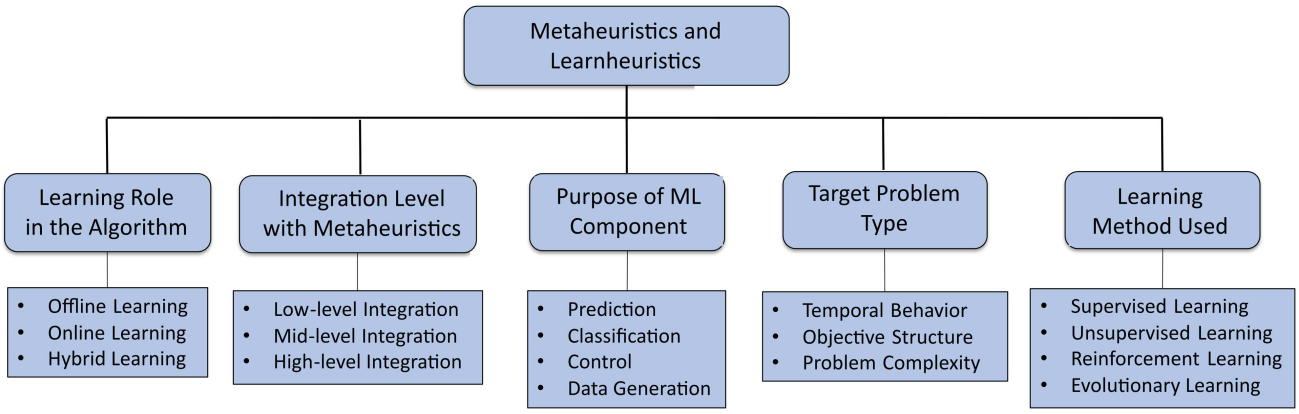
Table 1
List of widely recognized and frequently cited metaheuristic algorithms included in this study between 2019 and 2026

Metaheuristic	Year	# citations
Harris Hawks Optimization (HHO) [5]	2019	22500
Butterfly Optimization Algorithm (BOA) [6]	2019	11500
Gradient-Based Optimizer (GBO) [7]	2020	9480
Slime Mold Algorithm (SMO) [8]	2020	6680
Marine Predators Algorithm [9]	2020	6230
Equilibrium Optimizer [10]	2020	6140
Aquila Optimizer [11]	2021	4770
Seagull Optimization [12]	2019	4060
Manta Ray Foraging Optimization [13]	2020	3650
Chimp Optimization Algorithm [14]	2020	3590
Honey Badger Algorithm [15]	2022	3120
Archimedes Optimization Algorithm [16]	2021	2950
Squirrel Search Algorithm [17]	2019	2770
Tunicate Swarm Algorithm [18]	2020	2670
Henry Gas Solubility Optimization [19]	2019	2460

Figure 2
The Word cloud generated from the most frequently occurring terms in the selected metaheuristic optimization literature, based on publications collected from major academic databases over the considered time period



Figure 3
Proposed taxonomy of metaheuristic and learnheuristic optimization algorithms, illustrating their classification based on underlying search mechanisms and the integration of learning strategies



The contributions of our review can be listed as follows:

- A new taxonomy is proposed for metaheuristics and learnheuristics.
- Identification of 15 influential metaheuristic algorithms introduced between 2019 and 2026, based on criteria such as citation count, problem diversity, code availability, ease of parameter tuning, and resistance to optimization issues.
- Detailed analysis of selected algorithms, examining unique mechanisms, strengths, and limitations, to guide researchers and practitioners in selecting suitable algorithms for diverse optimization challenges.
- Evaluation of algorithm accessibility, parameter setting, binary encoding, and ease of implementation to encourage broader usability and adoption in academic and industrial contexts.
- Listing the state-of-the-art applications of learnheuristic and metaheuristic algorithms between 2019 and 2026 in various domains.
- Synthesis of emerging trends within the metaheuristic field, including new mechanisms, hybrid models, and similar strategies, provides insights into ongoing research directions and future exploration areas.

Section 2 presents an overview of surveys and reviews on learnheuristic and metaheuristic algorithms conducted over the last six years (2019–2026). Section 3 summarizes the 15 most influential metaheuristics selected from the same period, detailing their mathematical formulations and briefly explaining a few related studies. Section 4 highlights recent applications of metaheuristic algorithms, with a primary focus on their usage from 2019 to 2026. Section 5 discusses current challenges associated with recent metaheuristic algorithms. The final section offers concluding remarks and suggests directions for future research.

2. Previous Surveys

In recent years, the rapid development of metaheuristic optimization has led to a substantial increase in the number of survey and review studies published in the field. These studies,

however, vary considerably in terms of their scope, methodological approach, and areas of application. Some focus on specific algorithm families or optimization problems, whereas others provide broader discussions of recent developments and emerging trends. In this section, we review the existing survey literature from a structured and analytical perspective, with particular attention to the main topics covered, application domains addressed, and research directions identified. This analysis also allows us to highlight the limitations of previous reviews and identify areas that remain insufficiently explored [20, 21].

2.1. Categorization of existing surveys

The reviewed studies can be broadly grouped into three main categories based on their primary focus:

1) Methodological surveys: These studies focus on the development, comparison, and evaluation of metaheuristic algorithms. For example, Hussain et al. [22] analyzed 1222 publications, identifying research trends, algorithmic developments, and open challenges. Similarly, Halim et al. [23] examined simulation-based metaheuristics, emphasizing their advantages over deterministic approaches and the need for robust evaluation metrics. Wong and Ming [24] provided an overview of evolutionary algorithms, discussing benchmarking issues and real-world applications. In addition, Agushaka and Ezugwu [25] investigated initialization strategies in population-based algorithms, highlighting the importance of population diversity and size. Das et al. [26] comprehensively reviewed recent metaheuristics, proposing a structured taxonomy, analyzing their performance and limitations, and highlighting open challenges and future research directions. Tomar et al. [27] provided a comprehensive overview of these algorithms by comparing their key concepts, advantages, limitations, challenges, and practical applications to help researchers select appropriate methods for different optimization problems.

   Further methodological contributions include Ezugwu et al. [28], who proposed a taxonomy for classical and modern metaheuristics supported by a bibliometric analysis, and Hussain et al. [29], who analyzed swarm-based algorithms with a focus on exploration–exploitation balance. These studies provide valuable insights into algorithmic structures and performance characteristics, although they often rely on benchmark-based evaluations.

2) Application-oriented and domain-specific surveys: A second group of studies emphasizes the application of metaheuristics in specific problem domains. For instance, Pellerin et al. [30] reviewed hybrid metaheuristics for solving the resource-constrained project scheduling problem, analyzing their performance and design principles. Mohammed et al. [31] conducted a detailed meta-analysis of the whale optimization algorithm (WOA) and its variants, demonstrating improvements through hybridization.

   In addition, Abualigah et al. [32] evaluated metaheuristic optimization algorithms in engineering applications, providing comparative analyses and identifying future research directions. Similarly, Dokeroglu et al. [33] focused on feature selection problems, examining recent algorithms and their limitations. These studies highlight the practical relevance of metaheuristics, although they are often limited to specific domains.

3) Hybrid and learning-based surveys: More recent studies explore the integration of ML techniques with metaheuristics, leading to the emergence of learnheuristics. Talbi [4] addressed this integration by proposing a taxonomy based on search components and optimization objectives, while identifying key research challenges. Juan et al. [34] surveyed hybrid optimization approaches, including learnheuristics, demonstrating their effectiveness in solving large-scale NP-hard problems in domains such as logistics and healthcare.

   The concept of learnheuristics was initially introduced by Calvet et al. [35], who proposed combining ML with metaheuristics to handle dynamic combinatorial optimization problems. These approaches enhance adaptability by incorporating data-driven mechanisms into the optimization process.

2.2. Trends in recent research

One of the most notable developments in recent metaheuristic research is the growing use of ML techniques within optimization frameworks. The integration of learning mechanisms has encouraged the development of more adaptive and data-driven approaches, particularly for problems where the search environment or available information changes over time. Within this context, learnheuristics have attracted increasing attention by using information gathered during the optimization process to modify search strategies, guide the selection of promising solutions, or adjust algorithmic parameters.

At the same time, the number of newly proposed metaheuristic algorithms has continued to increase, with many methods drawing inspiration from natural, biological, or physical phenomena. Although this continuous development broadens the range of available optimization approaches, it has also led to growing discussion about the novelty and distinctiveness of newly introduced algorithms. In many cases, different methods rely on similar search principles or employ closely related mechanisms despite being presented under different names or metaphors. This raises questions about the actual contribution of some newly proposed algorithms and emphasizes the need for more careful evaluation of their methodological novelty, empirical performance, and differences from existing approaches.

2.3. Application domains

Metaheuristic and learnheuristic approaches have been widely applied across various domains. In engineering, they are used for design optimization and control systems. In ML, they support tasks such as feature selection and hyperparameter tuning. In operations research and logistics, they are applied to scheduling, routing, and resource allocation problems.

Furthermore, in dynamic and uncertain environments, learnheuristics provide adaptive solutions by integrating learning mechanisms into the optimization process. This makes them particularly suitable for real-world applications involving time-dependent or data-driven scenarios.

2.4. Limitations and research gaps

Despite the extensive body of literature, several limitations remain. First, many studies rely heavily on benchmark functions, with limited validation on real-world problems. Second, the absence of standardized evaluation frameworks makes it difficult to compare algorithms consistently.

Additionally, dynamic optimization problems are still relatively underexplored compared to static scenarios. Although learnheuristics offer promising solutions, their integration with metaheuristics is still evolving. Finally, the growing number of newly proposed algorithms highlights the need for more critical analysis and consolidation within the field.

3. The Most Influential Metaheuristics (Between 2019 and 2026)

In this section, we review recent developments in metaheuristic optimization, focusing on algorithms introduced between 2019 and 2026 and their applications in a range of state-of-the-art optimization problems. The aim is to provide a structured overview of these developments while examining the main ideas and mechanisms underlying the proposed methods.

A direct and fully standardized comparison of all reviewed algorithms is difficult because the studies considered in the literature often differ in problem definitions, datasets, experimental settings, evaluation metrics, and reporting practices. For this reason, the review does not rely solely on numerical performance comparisons. Instead, we complement the descriptive analysis with a cross-algorithm examination of commonly reported characteristics, including the balance between exploration and exploitation, sensitivity to parameter settings, and convergence behavior. In addition, Table 2 presents a more systematic analysis of algorithmic similarities, focusing on the underlying search operators and population dynamics rather than broad conceptual or metaphor-based analogies. This perspective provides a more meaningful basis for examining relationships among recently proposed metaheuristic algorithms.

3.1. Harris hawk optimization

The HHO algorithm is a powerful metaheuristic inspired by the cooperative hunting strategies of Harris Hawks, known for their group-based tactics and sudden attacks on prey [5]. The HHO algorithm is particularly effective in solving complex optimization problems characterized by nonlinearity, high dimensionality, and multiple local optima. HHO simulates hawks scouting for prey and executing coordinated surprise pounces, enabling both exploration (broad search) and exploitation (local search) of the solution space. See Figure 4 for the steps of the HHO.

Table 2
Comparison of similarities and mechanisms between metaheuristic algorithms introduced between 2019 and 2026

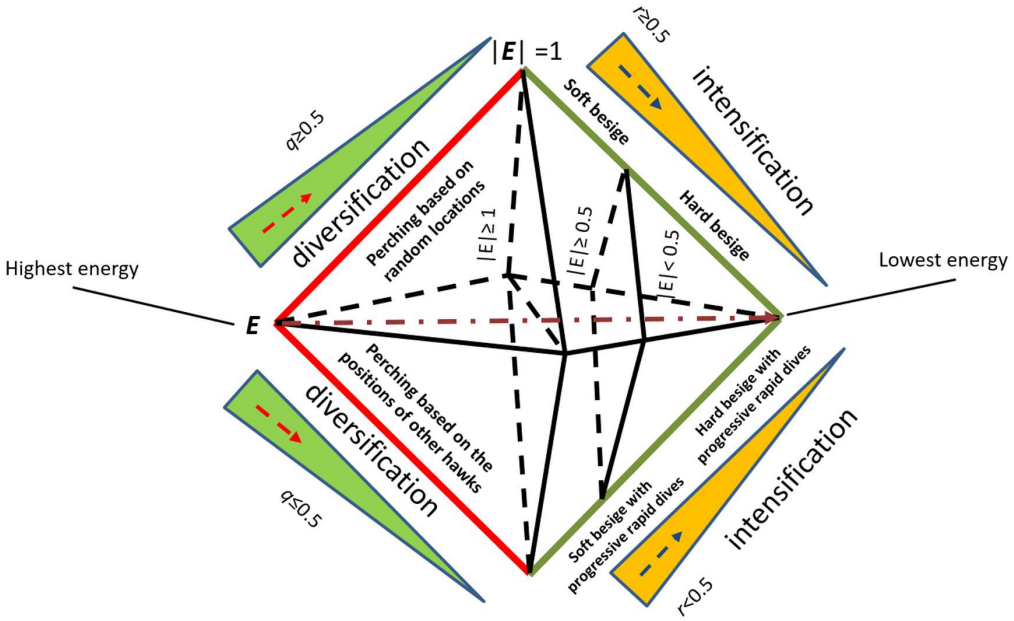
Algorithm	Similar to	Population interaction	Exploration mechanism	Exploitation mechanism	Key operator similarity
Harris Hawks Optimization	Grey Wolf Optimizer	Leader-based hierarchy	Random perching	Soft/hard besiege	Best-agent position update
Butterfly Optimization Algorithm	Bacterial Foraging Optimization	Fragrance-based indirect	Global attraction	Local random walk	Fitness-weighted movement
Gradient-Based Optimizer	Differential Evolution	Vector-based	Gradient perturbation	Local escaping operator	Differential-like update
Slime Mold Algorithm	Harmony Search	Weighted population	Oscillation search	Adaptive contraction	Weight-driven update
Marine Predators Algorithm	Grey Wolf Optimizer	Population-wide	Lévy flight	Elite phase switching	Multi-phase update
Equilibrium Optimizer	Particle Swarm Optimization	Global-best driven	Equilibrium pool	Attraction to equilibrium	PSO-like update (no velocity)
Aquila Optimizer	Harris Hawks Optimization	Best-solution driven	Wide-area soaring	Targeted dive	Adaptive step-size control
Seagull Optimization Algorithm	Particle Swarm Optimization	Social interaction	Migration	Attack phase	Velocity-like movement
Manta Ray Foraging Optimization	Whale Optimization Algorithm	Population coordination	Spiral motion	Somersault foraging	Trigonometric update
Chimp Optimization Algorithm	Cuckoo Search	Role-based	Random exploration	Best-agent attraction	Coefficient-based update
Squirrel Search Algorithm	Particle Swarm Optimization	Individual + global	Random movement	Local exploitation	Mean/best-guided search
Henry Gas Solubility Optimization	Cuckoo Search	Implicit interaction	Gaussian perturbation	Best-solution attraction	Probabilistic update

(*Continued*)

Table 2
(*Continued*)

Algorithm	Similar to	Population interaction	Exploration mechanism	Exploitation mechanism	Key operator similarity
Archimedes Optimization Algorithm	Genetic Algorithm	Evolutionary population	Density variation	Buoyancy adjustment	Physics-based update
Tunicate Swarm Algorithm	Particle Swarm Optimization	Swarm-based	Random drift	Best-agent attraction	Social-force modeling
Honey Badger Algorithm	Bat Algorithm	Local/global switch	Random digging	Aggressive exploitation	Distance-based update

Figure 4
The steps of the HHO metaheuristic according to the energy level (E), *q*, and *r* values



Initialize a population of N hawks, represented by:

$$X_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,d}), \quad i = 1, 2, \ldots, N, \tag{1}$$

where d is the dimension of the search space. The positions of the hawks are initialized randomly within the problem boundaries. During the exploration phase, Hawks randomly search for prey based on their current position and a reference leader (best solution found so far):

$$X_i(t+1) = X_{\text{rand}}(t) - r_1 |X_{\text{rand}}(t) - 2r_2 X_i(t)|, \tag{2}$$

where X_{rand} is a randomly chosen hawk and r_1 and r_2 are random numbers uniformly distributed in $[0, 1]$.

The transition from exploration to exploitation depends on the prey's behavior and escape energy E:

$$E = 2E_0\left(1 - \frac{t}{T}\right), \tag{3}$$

where E_0 is a random number in $[-1, 1]$, t is the current iteration, and T is the maximum number of iterations.

If $|E| \geq 0.5$, the hawks perform the soft besiege (exploitation phase):

$$X_i(t+1) = \Delta X(t) - E|JX_{\text{best}}(t) - X_i(t)|, \tag{4}$$

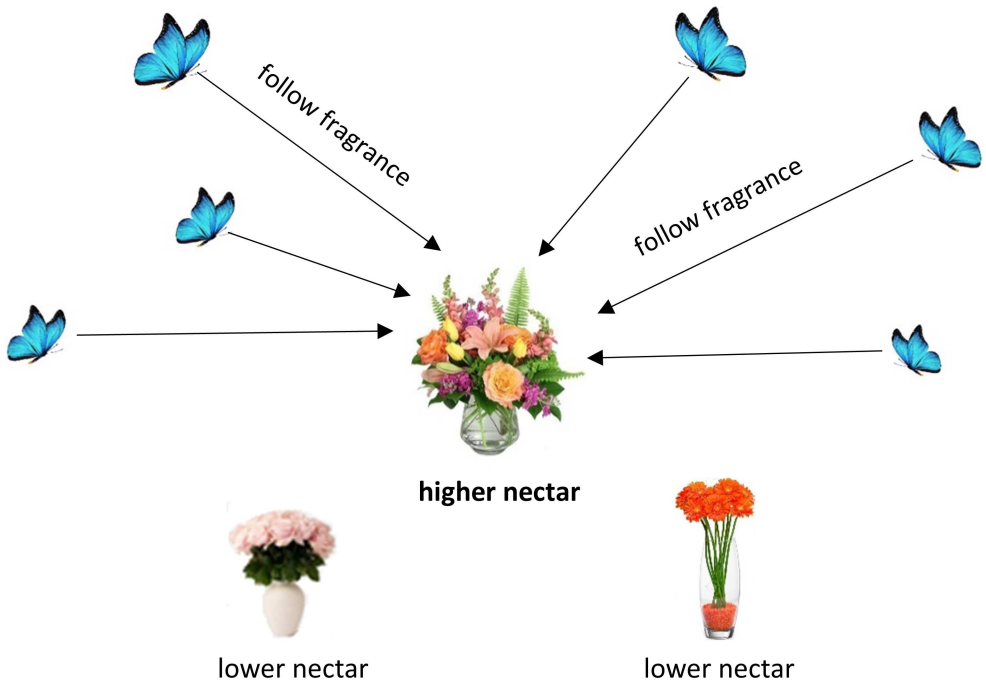
where $\Delta X(t)$ is the difference between the best and current solutions and J is a random jump strength coefficient.

Kamboj et al. [36] proposed the Hybrid Harris Hawks-Sine Cosine Algorithm (hHHO-SCA), which enhances HHO by incorporating the exploration behavior of the Sine–Cosine Algorithm. Designed for complex and highly constrained engineering problems, hHHO-SCA consistently outperformed SCA, HHO, and other optimizers such as Ant Lion Optimization (ALO), Moth Flame Optimization (MFO), and Grey Wolf Optimizer (GWO), demonstrating strong effectiveness in multidisciplinary design tasks.

3.2. Butterfly optimization algorithm

The Butterfly Optimization Algorithm (BOA) is a metaheuristic based on the foraging and mating behavior of butterflies [6]. It simulates the way butterflies use their sense of smell to find food or mates, balancing exploration and exploitation in the search space. BOA has been successfully applied to various optimization problems, demonstrating competitive performance in finding optimal solutions (see Figure 5 for the movement behavior of butterflies).

Figure 5
The movement behavior of butterflies



Butterflies perceive the quality of a solution (fitness) via sensory perception modeled as fragrance. The fragrance is defined as:

$$f_i = c \cdot I_i^a \tag{5}$$

where f_i is the fragrance of butterfly i, c is a constant, I_i is the fitness of the solution (smell intensity), and a is a sensory modality parameter, controlling the degree of perception. The movement of butterflies is controlled by both global and local search strategies, depending on the fragrance perceived. The global search allows butterflies to move toward the best solution in the population:

$$X_i(t+1) = X_i(t) + r \cdot f_i \cdot (X_{\text{best}} - X_i(t)) \tag{6}$$

where X_{best} is the best solution found so far, r is a random number in $[0, 1]$, and f_i is the fragrance of butterfly i. For local search, the movement is determined by the fragrance of nearby butterflies:

$$X_i(t+1) = X_i(t) + r \cdot f_i \cdot (X_j(t) - X_k(t)) \tag{7}$$

where $X_j(t)$ and $X_k(t)$ are two randomly selected butterflies. To switch between global and local search, a random switching probability p is introduced:

$$p = \text{rand}(0, 1) \tag{8}$$

If p is less than a threshold p_0, a global search is performed; otherwise, a local search is executed. This mechanism ensures a balance between exploration and exploitation. The sensory modality parameter a is adapted over iterations to fine-tune the algorithm:

$$a(t) = a_{\min} + (a_{\max} - a_{\min}) \cdot \frac{t}{T} \tag{9}$$

where $a_{\min}$ and $a_{\max}$ define the range for the sensory modality, t is the current iteration, and T is the total number of iterations.

Makhadmeh et al. [6] presented a comprehensive overview of BOA, including its inspiration, mathematical model, key adaptations, and main application areas. The study classifies BOA variants as original, modified, or hybrid and analyzes its strengths and limitations in solving optimization problems.

3.3. Gradient-based optimizer

A gradient-based optimizer (GBO) combines gradient- and population-based methods; the search direction is specified by Newton's method to explore the search domain utilizing a set of vectors and two main operators (i.e., gradient search rule and local escaping operators (LEOs)). Minimization of the objective function is considered in the optimization problems [7, 37]. GBOs operate by calculating gradients—essentially the slope or rate of change of the function with respect to the model parameters—and then updating these parameters in the direction that reduces the objective function, aiming for an optimal or near-optimal solution. One of the most widely used GBOs is Stochastic Gradient Descent, which updates parameters based on the gradient calculated for a single or mini-batch of samples.

Premkumar et al. [38] introduced a multi-objective GBO (MOGBO) for solving multi-objective truss-bar design problems. MOGBO employs a gradient-based approach with operators like the LEO and gradient search rule, using non-dominated sorting and crowding distance mechanisms to achieve Pareto optimal solutions. Performance tests on various benchmark problems show MOGBO outperforms other algorithms in accuracy, runtime, and metrics like hyper-volume and diversity, proving its effectiveness in complex multi-objective optimization tasks. Jiang et al. [39] proposed eight variants of the binary GBO utilizing S-shaped and V-shaped transfer functions to convert the search space to a discrete format. The performance of these binary GBO algorithms is evaluated on 18 UCI datasets and 10 high-dimensional datasets, comparing them against other feature

selection methods. Results indicate that one binary GBO variant outperforms other algorithms, demonstrating superior overall performance in various metrics.

3.4. Slime mold algorithm

The Slime Mold Algorithm (SMA) is inspired by the foraging behavior and morphological changes of the slime mold **Physarum polycephalum**, rather than attempting to model its complete life cycle [8, 40]. **Physarum polycephalum** is a eukaryotic organism that typically thrives in cool and humid environments. Its plasmodium stage is the most active phase of its life cycle and is the primary focus of the biological model underlying SMA. During this stage, the slime mold searches for and surrounds food sources while releasing enzymes that facilitate digestion. As it moves toward available nutrients, its leading edge expands into a fan-like structure supported by an interconnected network of veins through which cytoplasmic flow is maintained, as illustrated in Figure 6. This flexible network structure enables the organism to explore and connect multiple food sources simultaneously, forming efficient pathways between them. The foraging morphology and behavior of the slime mold that inspire the SMA are presented in Figure 6.

The following formulas have been proposed to simulate the contraction mode:

$$\vec{X}(t+1) = \begin{cases} \vec{X}_b(t) + \overrightarrow{vb} \cdot \left(\vec{W} \cdot \left(\vec{X}_A(t) - \vec{X}_B(t)\right)\right), & r < p \\ \vec{vc} \cdot \overrightarrow{X(t)}, & r \geq p \end{cases} \tag{10}$$

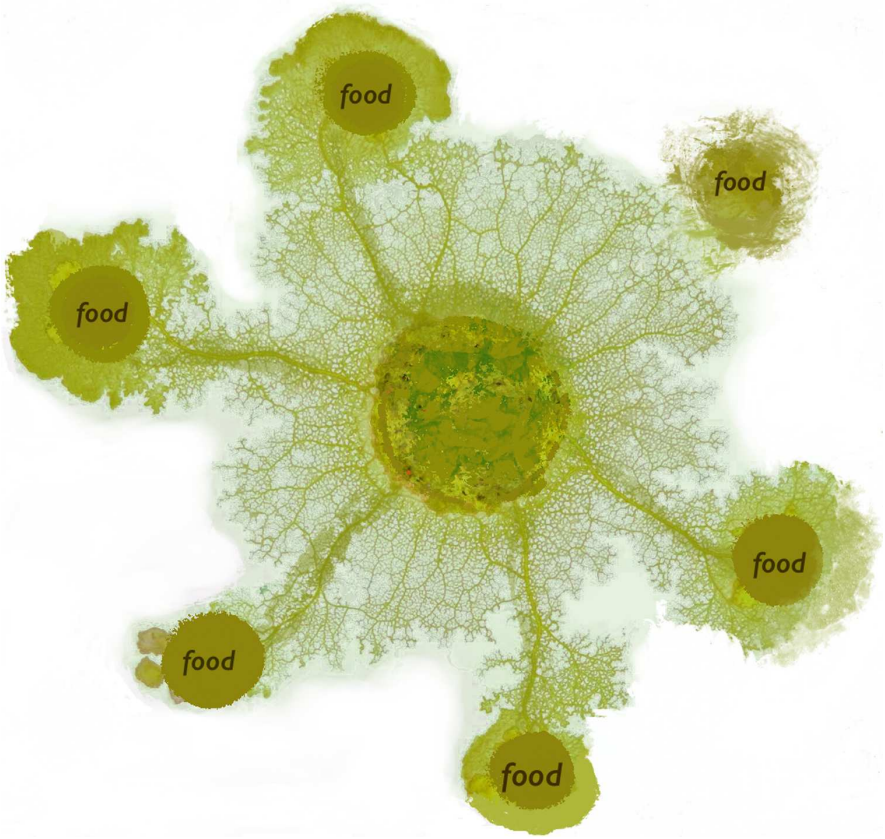
where $\overrightarrow{vb}$ is a parameter that ranges from $[-a, a]$ and $\vec{vc}$ decreases linearly from one to zero. The variable t denotes the current iteration, $\overrightarrow{X_b}$ indicates the location of the individual with the highest odor concentration detected, $\vec{X}$ represents the position of the slime mold, and $\overrightarrow{X_A}$ and $\overrightarrow{X_B}$ are two individuals randomly selected from the slime mold population. Additionally, $\vec{W}$ signifies the weight of the slime mold, which is formulated as follows:

$$\vec{W}(\text{SmellIndex}(i)) = \begin{cases} 1 + r \cdot \log\left(\frac{bF - S(i)}{bF - wF} + 1\right), & \text{condition} \\ 1 - r \cdot \log\left(\frac{bF - S(i)}{bF - wF} + 1\right), & \text{otherwise} \end{cases} \tag{11}$$

where "condition" indicates that $S(i)$ ranks in the top half of the population, r represents a random value within the interval $[0, 1]$, bF signifies the best fitness value achieved during the current iteration, wF represents the worst fitness value obtained thus far in the iterative process, and SmellIndex refers to the sequence of fitness values sorted in ascending order for the minimum value problem.

Chen et al. [40] studied and analyzed key research related to the development of the SMA. A total of 98 SMA-related studies were retrieved, selected, and identified from the WoS database. The review focuses on two main aspects: advanced versions of the SMA and its application domains. Premkumar et al. [41] presented a Multi-objective SMA (MOSMA) for tackling multi-objective optimization challenges in industrial settings, based on the oscillatory behaviors of slime mold in laboratory experiments. MOSMA integrates the core principles of SMA with elitist

Figure 6
Foraging morphology of slime mold



non-dominated sorting and a crowding distance operator to ensure broad coverage of Pareto optimal solutions.

3.5. Marine predators algorithm

The Marine Predators Algorithm (MPA) simulates the behavior of marine predators foraging in the ocean [9]. The algorithm primarily relies on different movement phases that represent various predation strategies based on the interaction between predators and prey. See Figure 7 for the phases of the MPA.

In the initial exploration phase, the algorithm uses random movements inspired by Lévy flights. The position of each predator X_i at iteration $t + 1$ is updated as follows:

$$\mathbf{X}_i^{t+1} = \mathbf{X}_i^t + r \cdot \text{Lévy}(\lambda) \cdot (\mathbf{X}_i^t - \mathbf{X}_{\text{best}}^t),$$

where r is a random number in the range $[0, 1]$, Lévy(λ) represents a Lévy flight with scaling parameter λ, and $\mathbf{X}_{\text{best}}^t$ is the position of the best solution found so far.

Su and Liu [42] proposed a novel marine predator whale optimization algorithm (MPWOA) that integrates predator–prey behaviors from WOA and MPA. The algorithm employs Sinusoidal, Brownian, and Levy motion models to balance exploration and exploitation, supported by ranking-based updates, chaotic maps, and adaptive inertia weights to avoid local optima and accelerate convergence. Experimental results on CEC 2017/2019 benchmarks and real-world problems demonstrate that MPWOA achieves faster convergence and higher-quality solutions compared to state-of-the-art algorithms. Ramezani et al. [43] proposed an enhanced MPA variant that integrates opposition-based learning (OBL), chaotic mapping, self-adaptive population techniques, and an adaptive phase-switching mechanism for improved exploration and exploitation.

3.6. Equilibrium optimizer

The Equilibrium Optimizer (EO) is inspired by the mass balance principles used to describe dynamic systems that evolve toward an equilibrium state [10, 44]. The algorithm models the optimization process as a gradual movement toward equilibrium, where candidate solutions are iteratively updated to balance exploration of the search space with exploitation of promising regions. This process is controlled through concentration-based updating mechanisms that determine how the solutions move during the search.

EO maintains an equilibrium pool composed of several candidate solutions that serve as reference points for updating the population. By considering multiple equilibrium candidates and employing adaptive control mechanisms, the algorithm guides individuals toward potentially promising regions while retaining sufficient diversity in the population. The position of each individual is updated with respect to the selected equilibrium candidates according to the following equation:

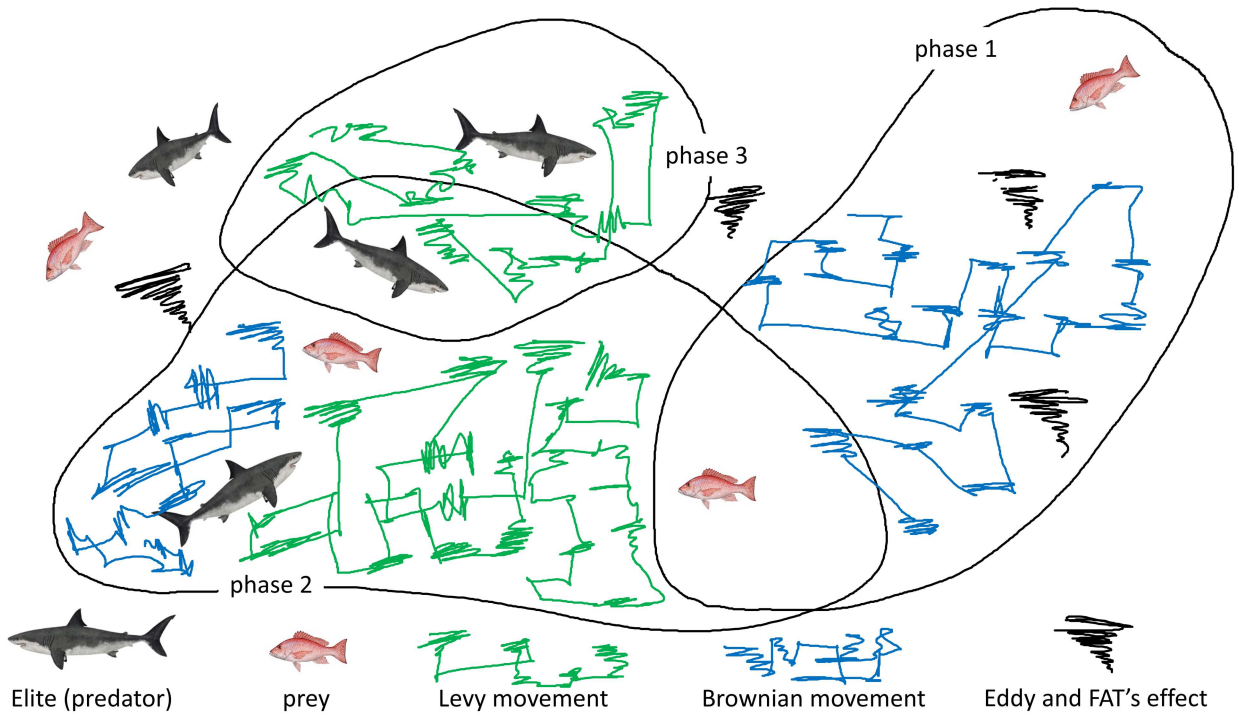
$$\begin{aligned} X_i(t+1) = X_i(t) + \lambda \cdot (X_{\text{eq}}(t) - X_i(t)) \\ + F \cdot (X_{\text{eq}}(t) - X_{\text{rand}}(t)) \end{aligned} \tag{12}$$

where $X_{\text{eq}}(t)$ is the position of the equilibrium candidate at iteration t, λ is the random control parameter for exploration, F is the control parameter for exploitation, and $X_{\text{rand}}(t)$ is a random solution to introduce diversity. The parameters λ and F are updated dynamically over time to balance exploration and exploitation:

$$\lambda = 1 - \frac{t}{T} \tag{13}$$

$$F = \text{rand}\,(0, 1) \tag{14}$$

Figure 7
Three phases of the marine predators metaheuristic



where t is the current iteration and T is the maximum number of iterations. The equilibrium candidates in the pool are updated to reflect the best solutions found so far. This ensures that individuals are attracted toward high-quality solutions while maintaining diversity:

$$X_{\text{eq}}(t+1) = X_{\text{best}}(t) + \beta \cdot (X_{\text{best}}(t) - X_{\text{mean}}(t)) \quad (15)$$

where $X_{\text{best}}(t)$ is the best solution at iteration t, $X_{\text{mean}}(t)$ is the mean solution across the population, and β is a constant controlling the influence of the best solution. As iterations proceed, the control parameters λ and F help the algorithm converge toward the equilibrium by reducing random fluctuations and encouraging exploitation.

Wang et al. [45] proposed an improved EO using a neural network to enrich photovoltaic (PV) cell data, enhancing optimization efficiency. Tested on three diode models, it outperforms other algorithms, achieving lower error rates and improving both precision and reliability, making it highly effective for PV cell parameter estimation.

3.7. Aquila optimizer

The Aquila Optimizer (AO) is inspired by the hunting behavior of Aquila, a bird of prey [46]. AO mimics Aquila's powerful and efficient hunting strategies, combining exploration and exploitation to search for global optima. The algorithm consists of different movement strategies, which are applied dynamically to balance exploration of the search space and exploitation of promising regions (see Figure 8 for its soar and vertical dive behavior).

AO dynamically switches between different movement strategies depending on the stage of the hunt:

$$X_i(t+1) = X_{\text{best}}(t) + \alpha \cdot F(X_{\text{best}}(t) - X_i(t)) \quad (16)$$

During the hunting phase, the distance between the Aquila and the prey is calculated, influencing its strategy:

$$D = |C \cdot X_{\text{best}}(t) - X_i(t)| \quad (17)$$

where D is the distance between Aquila i and the prey and C is a coefficient representing the influence of the prey's position. Aquilas perform a dive to capture the prey with more precision, expressed as:

$$X_i(t+1) = X_{\text{best}}(t) + D \cdot e^{b \cdot l} \cdot \sin(2\pi l) \quad (18)$$

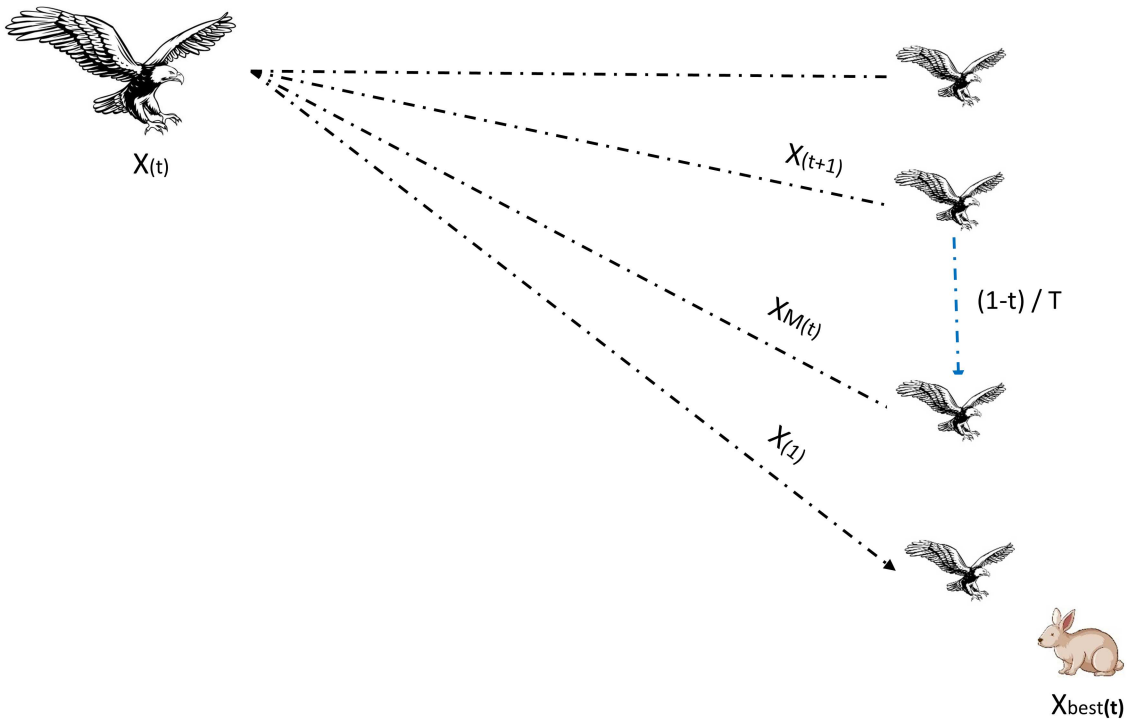
where b controls the width of the dive, l is a random variable controlling the angle of the attack, and e is the base of the natural logarithm, indicating the sharpness of the dive. AO uses adaptive parameters to adjust the search dynamically.

Al-qaness et al. [47] addressed the shortcomings of the Adaptive Neuro-Fuzzy Inference System (ANFIS) model in oil production estimation by optimizing its parameters with a modified AO and OBL. The proposed model outperforms several modified ANFIS models and time-series forecasting methods using real-world datasets and performance metrics like root mean square error and mean absolute error. Mahajan et al. [48] introduced a hybrid optimization method combining AO and Arithmetic Optimization Algorithm to enhance convergence and solution quality.

3.8. Seagull optimization

Dhiman et al. [12] proposed the Seagull Optimization Algorithm (SOA), a bio-inspired metaheuristic that models the migration and attacking behaviors of seagulls. The algorithm uses these behaviors to regulate the exploration and exploitation of the search space. Its performance was evaluated against nine well-known metaheuristic algorithms using 44 benchmark test

Figure 8
Aquila's high soar and vertical dive behavior



functions, with particular attention given to computational complexity and convergence characteristics. In addition, SOA was tested on seven constrained industrial optimization problems to assess its performance in practical settings. The reported results showed that SOA achieved competitive performance across the benchmark problems and was effective in handling large-scale and computationally demanding constrained optimization tasks.

The migration behavior modeled in SOA is governed by three main conditions. The first is collision avoidance, which helps prevent search agents from moving too close to one another. To achieve this, an additional variable, A, is incorporated into the position-updating process of each search agent:

$$\overrightarrow{C_s} = A \times \overrightarrow{P_s}(x) \tag{19}$$

where $\overrightarrow{C_s}$ denotes the position of the search agent that does not collide with other search agents, $\overrightarrow{P_s}$ indicates the current position of the search agent, x refers to the current iteration, and A represents the movement behavior of the search agent within the specified search space. (ii) Movement toward the best neighbor's direction: After avoiding collisions with their neighbors, the search agents proceed in the direction of the best neighboring agent:

$$\overrightarrow{M_s} = B \times (\overrightarrow{P_{bs}}(x) - \overrightarrow{P_s}(x)) \tag{20}$$

where $\overrightarrow{M_s}$ indicates the position of the search agent $\overrightarrow{P_s}$ in relation to the best-fit search agent $\overrightarrow{P_{bs}}$ (i.e., the fittest seagull). The behavior of B is randomized, which helps maintain an appropriate balance between exploration and exploitation. (iii) Stay close to the best search agent: Finally, the search agent can adjust its position in relation to the best search agent:

$$\overrightarrow{D_s} = |\overrightarrow{C_s} + \overrightarrow{M_s}| \tag{21}$$

where $\overrightarrow{D_s}$ denotes the distance between the search agent and the best-fit search agent (i.e., the best seagull with the lowest fitness value).

Panagant et al. [49] developed a surrogate-assisted SOA for structural shape optimization, targeting minimum mass under stress constraints. Using FEA with a Kriging surrogate, the method achieved competitive performance comparable to WOA and Squirrel Search Algorithm (SSA), showing strong suitability for industrial component design.

3.9. Manta ray foraging optimization

The Manta Ray Foraging Optimization (MRFO) algorithm is inspired by the foraging behavior of manta rays, focusing on both exploration and exploitation strategies through specific movements [13]. See Figure 9 for the somersault foraging behavior of three mantas in two dimensions.

In the exploration phase, manta rays use a spiral movement to search for prey, modeled by the following update equation:

$$\mathbf{X}_i^{t+1} = \mathbf{X}_i^t + A \cdot \left(\cos(\theta) \cdot (\mathbf{X}_i^t - \mathbf{X}_{\text{best}}^t) + \sin(\theta) \cdot (\mathbf{X}_i^t - \mathbf{X}_{\text{mean}}^t)\right),$$

where A is a scaling factor, θ is a random angle in the range $[0, 2\pi]$, $\mathbf{X}_{\text{best}}^t$ is the position of the best solution found so far, and $\mathbf{X}_{\text{mean}}^t$ is the mean position of the population. In the exploitation phase, manta rays move toward the best solution by adjusting their positions using the following equation:

$$\mathbf{X}_i^{t+1} = \mathbf{X}_{\text{best}}^t + B \cdot (\mathbf{X}_i^t - \mathbf{X}_{\text{best}}^t),$$

where B is a random factor influencing the movement toward the best solution.

Additionally, the manta ray may use a local search for the best solution:

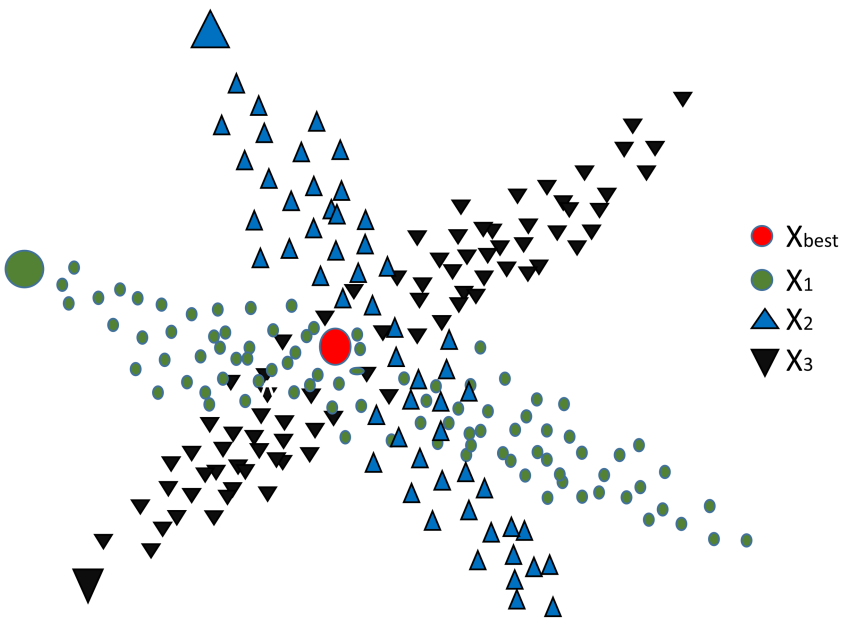
$$\mathbf{X}_i^{t+1} = \mathbf{X}_i^t + F \cdot (\mathbf{X}_{\text{best}}^t - \mathbf{X}_i^t),$$

where F is a random factor that governs the intensity of the search.

Manta rays search for prey by adapting their movement toward the position of the prey. This is represented by:

$$\mathbf{X}_i^{t+1} = \mathbf{X}_i^t + C \cdot (\mathbf{X}_{\text{prey}}^t - \mathbf{X}_i^t),$$

Figure 9
Somersault foraging behavior of three mantas in 2D



where C is a scaling factor and $\mathbf{X}^t_{\text{prey}}$ is the position of the prey.

Tang et al. [50] proposed m-MRFO, which enhances MRFO using an elite search pool, adaptive parameter control, and distribution estimation. Tested on 23 benchmarks and CEC2017, it delivers clear performance gains and proves effective for engineering design tasks. Houssein et al. [51] introduced MRFO-OBL, integrating OBL to boost diversity and avoid local optima. Applied to multilevel thresholding for COVID-19 CT image segmentation, MRFO-OBL outperforms six metaheuristics—including standard MRFO—achieving higher segmentation quality, consistency, and robustness.

3.10. Chimp optimization algorithm

The Chimp Optimization Algorithm (ChOA) is inspired by the intelligent hunting and social cooperation behaviors of chimpanzees [14]. The algorithm models chimpanzee hunting strategies to balance exploration and exploitation during the optimization process. ChOA incorporates four main roles in chimpanzee groups: attackers, drivers, barriers, and chasers, each contributing to different search behaviors.

$$X_i(t+1) = X_{\text{best}}(t) - A \cdot D \tag{22}$$

where $X_{\text{best}}(t)$ is the position of the best chimpanzee at iteration t, A is a coefficient that controls the direction and step size, and D represents the distance between chimpanzee i and the prey (best solution). The coefficient A is calculated as follows:

$$A = 2 \cdot a \cdot r - a \tag{23}$$

where a decreases linearly from 2 to 0 over the course of iterations, balancing exploration and exploitation, and r is a random number between 0 and 1. The distance between the chimpanzee and prey is calculated as:

$$D = |C \cdot X_{\text{best}}(t) - X_i(t)| \tag{24}$$

where C is another coefficient that controls the exploration phase and is calculated as:

$$C = 2 \cdot r \tag{25}$$

Khishe et al. [52] proposed a weighted ChOA to improve convergence and avoid local optima using a position-weighted equation, achieving better exploration–exploitation balance and superior performance on benchmark and real-world problems. Jia et al. [53] introduced an Enhanced ChOA that incorporates polynomial mutation, Spearman's rank correlation, and a beetle antennae operator to boost accuracy and exploration, outperforming the original ChOA and several algorithms across diverse benchmark and engineering tests.

3.11. Honey badger algorithm

The Honey Badger Algorithm (HBA) is inspired BY the foraging behavior and fearless nature of honey badgers [15]. It is designed to solve complex optimization problems by utilizing a population of agents that explore the search space, balancing exploration and exploitation. The algorithm incorporates strategies such as local search, random movement, and hierarchical structure, allowing it to escape local optima and efficiently converge toward global solutions. HBA has been applied in various fields, including engineering, finance, and ML, demonstrating strong performance compared to other optimization algorithms. Figure 10 shows the inverse square law technique used by honey badgers.

In the exploration phase, honey badgers perform random search movements to explore the search space. The position update equation is as follows:

$$\mathbf{X}_i^{t+1} = \mathbf{X}_i^t + \alpha \cdot \text{rand} \cdot (\mathbf{X}_i^t - \mathbf{X}_{\text{best}}^t),$$

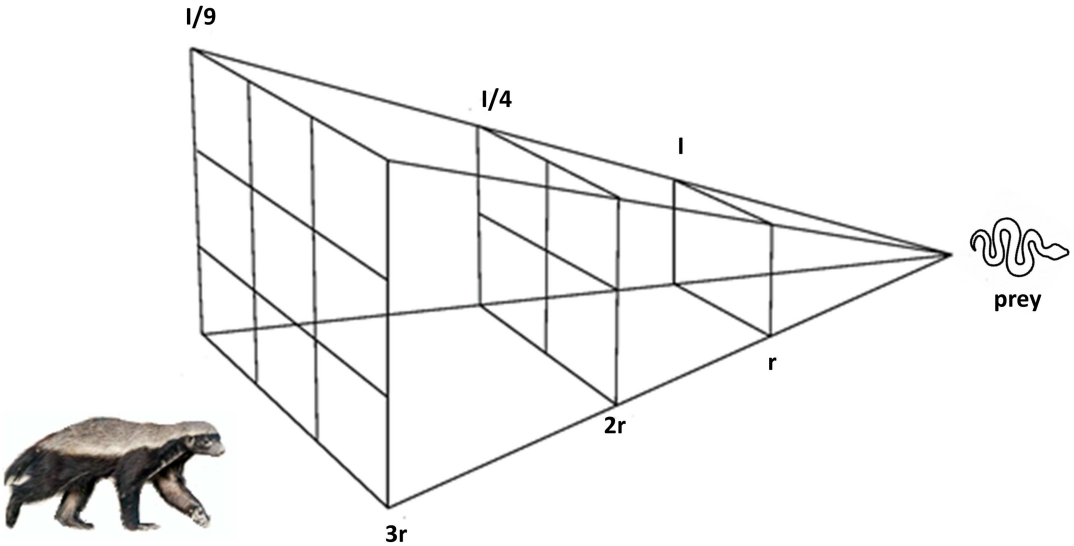
where α is a scaling factor, rand is a random number between $[0, 1]$, and $\mathbf{X}_{\text{best}}^t$ is the best solution found so far.

In the exploitation phase, honey badgers exploit the best solution by updating their positions using a local search mechanism:

$$\mathbf{X}_i^{t+1} = \mathbf{X}_{\text{best}}^t + \beta \cdot (\mathbf{X}_i^t - \mathbf{X}_{\text{best}}^t),$$

where β is a random factor determining the movement toward the best solution.

Figure 10
The inverse square law technique



Alternatively, if the honey badger is near a food source, it performs a more aggressive search:

$$\mathbf{X}_i^{t+1} = \mathbf{X}_i^t + \gamma \cdot (\mathbf{X}_{\text{best}}^t - \mathbf{X}_i^t),$$

where γ is a factor controlling the intensity of the aggressive search.

Duzenli et al. [54] improved the HBA for PV parameter estimation by proposing two variants: one using a Gauss/Mouse chaotic map to enhance exploration–exploitation balance and another integrating OBL for more efficient search. Tested on CEC2017/CEC2019 benchmarks, both variants achieved strong performance across single-diode, double-diode, and multiple PV models. Xu et al. [55] introduced symbiotic mechanism-based HBA (SHBA), which, when evaluated on CEC2017, CEC2022, and several engineering problems, showed faster convergence, higher accuracy, and overall competitive results compared to nine advanced algorithms.

3.12. Squirrel search algorithm

The SSA mimics the foraging behavior of flying squirrels, which involves both exploration and exploitation [17]. The algorithm is characterized by random search movements and local search around discovered food sources. Figure 11 shows the fly behavior of squirrels between trees.

In the exploration phase, squirrels search for food sources by moving randomly across the space. This is modeled as:

$$\mathbf{X}_i^{t+1} = \mathbf{X}_i^t + \alpha \cdot \text{rand} \cdot (\mathbf{X}_i^t - \mathbf{X}_{\text{best}}^t),$$

where α is a scaling factor, rand is a random number between $[0, 1]$, and $\mathbf{X}_{\text{best}}^t$ is the position of the best squirrel found so far.

In the exploitation phase, squirrels exploit the discovered food sources by performing a local search around the best food source found so far. The exploitation movement is given by:

$$\mathbf{X}_i^{t+1} = \mathbf{X}_{\text{best}}^t + \beta \cdot (\mathbf{X}_i^t - \mathbf{X}_{\text{best}}^t),$$

where β is a random factor that determines the intensity of the search around the best solution.

Alternatively, a squirrel may perform a random walk around the current solution:

$$\mathbf{X}_i^{t+1} = \mathbf{X}_i^t + \gamma \cdot \text{rand} \cdot (\mathbf{X}_i^t - \mathbf{X}_{\text{mean}}^t),$$

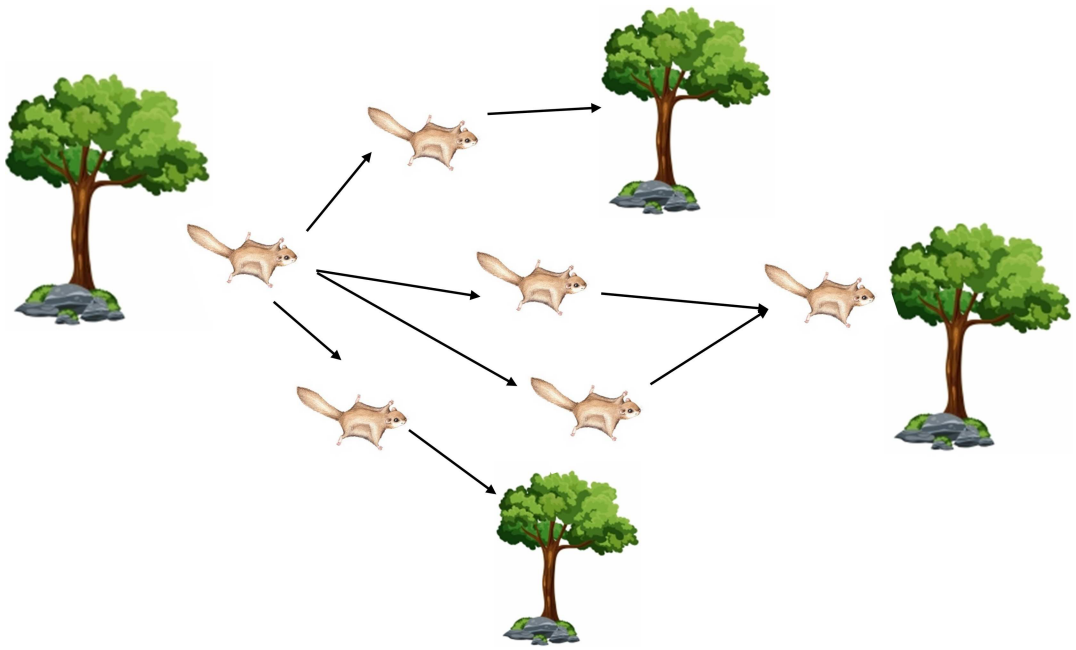
where γ is a scaling factor and $\mathbf{X}_{\text{mean}}^t$ is the mean position of the population.

Sakthivel et al. [56] proposed a multi-objective SSA for economic and environmental power dispatch, using Pareto dominance and an elitist repository with crowding distance to maintain diverse optimal solutions. Tested on three complex systems, it achieves superior cost–emission trade-offs compared to other heuristics. Kalyan et al. [57] applied SSA to optimize a fractional-order fuzzy proportional-integral-derivative (PID) controller for a dual-area hydro–thermal–gas system, accounting for real-world nonlinearities. Combined with static synchronous series compensator (SSSC) and superconducting magnetic energy storage (SMES) devices, the controller outperforms classical PID, traditional fuzzy PID, and recent methods in performance, robustness, and accuracy.

3.13. Archimedes optimization algorithm

The Archimedes Optimization Algorithm (AOA) is a metaheuristic optimization algorithm inspired by Archimedes' principle, specifically buoyancy and density principles [16]. The algorithm mimics the buoyant force acting on an object submerged in a fluid, which balances exploration (search for new solutions) and exploitation (refining existing solutions) by

Figure 11
The fly behavior of squirrels between trees



adjusting the density and volume of the solutions over time. The AOA algorithm generates initial candidate solutions randomly. The key component of the AOA is the buoyant force, which is calculated using the principle of buoyancy:

$$F_b = \rho \cdot V \cdot g \tag{26}$$

where F_b is the buoyant force, ρ is the density of the fluid (related to the solution's quality), V is the volume of the object (candidate solution), and g is the gravitational acceleration constant. The positions of the candidate solutions are updated based on the calculated buoyant force and density:

$$X_i(t+1) = X_i(t) + F_b \cdot \left(1 - \frac{\rho}{\rho_{\max}}\right) \tag{27}$$

where $X_i(t+1)$ is the new position of solution i at time $t+1$, F_b is the buoyant force acting on the solution, and $\rho_{\max}$ is the maximum allowable density, controlling the search's exploration and exploitation balance.

Yıldız et al. [58] examined the application of AOA in minimizing product development costs. It focuses on optimizing vehicle structures using size, shape, and topology optimization, demonstrating political optimization algorithm (POA) superior search capability and computational efficiency. Akdag [59] proposed an Improved AOA (IAOA) to solve the Optimal Power Flow problem. IAOA enhances population diversity and balances exploitation and exploration to prevent premature convergence. Tested on IEEE and South Marmara systems, the obtained simulation results and comparisons with different techniques show that the IAOA provides robustness in minimizing fuel emissions.

3.14. Tunicate swarm algorithm

The Tunicate Swarm Algorithm (TSA) is inspired by the collective behavior of tunicates, specifically their social interactions and foraging strategies [18]. Tunicates, also known as sea squirts, exhibit unique behaviors that allow them to effectively find food and adapt to their environment. The TSA captures these behaviors to create an efficient search mechanism for solving complex optimization problems. Tunicates possess the ability to locate food sources in the ocean, yet there is no information available about the food source within the specified search area. To avoid conflicts between search agents that are other tunicates in the swarm, a vector $\vec{A}$ is utilized to calculate the new positions of the search agents as shown in Formula 28.

$$\vec{A} = \vec{G} + \vec{M} \tag{28}$$

$\vec{G}$ represents the gravitational force, and $\vec{M}$ signifies the social forces acting between the search agents. The calculation for vector $\vec{M}$ is expressed as:

$$\vec{M} = \lfloor P_{min} + c_1 \cdot (P_{max} - P_{min}) \rfloor \tag{29}$$

P_{min} and P_{max} denote the initial and subordinate speeds that facilitate social interaction. In [18], P_{min} and P_{max} are set to 1 and 4, respectively. After conflicts have been resolved, the search agents then move toward the direction of their best neighbor. The calculation is represented by:

$$\overrightarrow{PD} = \left| \overrightarrow{FS} - r_{and} \cdot \vec{P}_p(x) \right| \tag{30}$$

where $\overrightarrow{PD}$ is the distance between the food source and the search agent (the tunicate), x indicates the current iteration, $\overrightarrow{FS}$ is the position of the food source (the optimum), $\vec{P}_p(x)$ signifies the position of the tunicate, and r_{and} is a random number within a specified range. Then, the search agent positions itself in relation to the best search agent (food source). The formula for this positioning is:

$$\vec{P}_p(x') = \begin{cases} \vec{F}S + \vec{A} \cdot \vec{P}_D, & \text{if rand} \geq 0.5 \\ \vec{F}S - \vec{A} \cdot \vec{P}_D, & \text{if rand} < 0.5 \end{cases} \tag{31}$$

where $\vec{P}_p(x')$ is the updated position of the tunicate for the position of the food source $\overrightarrow{FS}$. To mathematically simulate the swarm behavior of tunicates, the first two optimal solutions are recorded, enabling the update of the positions of other search agents based on the locations of the best agents. The following formula describes this swarm behavior:

$$\vec{P}_p(x+1) = \frac{\vec{P}_p(x) + \vec{P}_p(x+1)}{2 + c_1} \tag{32}$$

Search agents update their positions as $\vec{P}_p(x')$, according to the best agents. The final position will be randomly located within a cylindrical or cone-shaped area defined by the position of the tunicate.

Houssein et al. [51] introduced a TSA variant enhanced with an LEO to prevent stagnation and improve convergence and local search, showing clear performance gains on the CEC'2017 benchmark suite. Rizk-Allah et al. [60] proposed the Enhanced TSA (ETSA), which strengthens exploration and exploitation, performs robustly across 20 benchmark functions, and generally runs faster than competing methods. ETSA also demonstrated practical effectiveness in solving the Economic Dispatch Problem.

3.15. Henry gas solubility optimization

Henry Gas Solubility Optimization (HGSO) is a metaheuristic inspired by the behavior of gas molecules in a liquid solution, based on Henry's Law [61]. The algorithm simulates how gas particles interact and converge toward an optimal solution in the solution space. In HGSO, each molecule adjusts its position based on its concentration and the solubility coefficient k_H (Henry's constant), which impacts the movement intensity of each molecule.

The concentration C_i for each molecule is updated to reflect its solution quality:

$$C_i^{(t+1)} = C_i^{(t)} + \alpha \left(C_{\text{best}} - C_i^{(t)} \right)$$

where C_{best} is the concentration of the best solution found so far, and α is a learning factor that controls the influence of C_{best} on C_i.

The solubility of each molecule is affected by Henry's coefficient, which is updated using an evaporation function:

$$k_H^{(t+1)} = k_H^{(t)} \cdot e^{-\beta \cdot t}$$

where β is a decay parameter and t is the current iteration. This coefficient modulates the search intensity, reducing as iterations progress to encourage convergence.

Each molecule's new position is determined by the concentration and solubility effects:

$$\mathbf{x}_i^{(t+1)} = \mathbf{x}_i^{(t)} + k_H^{(t)} \cdot (C_{\text{best}} - C_i^{(t)}) \cdot \text{randn}(\mathbf{x}_i)$$

where randn($\mathbf{x}_i$) is a Gaussian-distributed random number applied to introduce controlled stochastic behavior.

The algorithm stops when a maximum number of iterations T is reached, or the change in the best solution is less than a threshold ϵ:

$$\|\mathbf{x}_{\text{best}}^{(t+1)} - \mathbf{x}_{\text{best}}^{(t)}\| < \epsilon$$

Yıldız et al. [62] proposed the chaotic HGSO (CHGSO), which integrates chaotic maps to improve convergence in complex engineering optimization. Tested on several constrained design problems, CHGSO proved robust and effective, consistently achieving high-quality solutions across mechanical and manufacturing applications.

Other recent metaheuristics can be seen in Table A1 (Appendix A).

4. State-of-the-Art Applications of Metaheuristics and Learnheuristics

This section outlines recent applications of metaheuristics and learnheuristics from 2019 to 2026 across domains such as engineering, healthcare, energy, telecommunications, and urban planning. In engineering, they optimize structural designs, energy systems, and control mechanisms. In ML, metaheuristics support hyperparameter tuning and feature selection, improving model performance.

1) Engineering and energy systems: Metaheuristics are extensively used for structural optimization, energy systems, and control applications. For example, Dhiman [63] introduced the Emperor Penguin Optimizer-based emperor penguin and salp swarm algorithm (ESA), while Xu and Yin [64] proposed the Black Widow Optimization algorithm, both demonstrating strong performance on benchmark and engineering problems. In the energy domain, Güven and Samy [65] optimized hybrid off-grid systems using a hybrid Firefly–Genetic Algorithm, and Minai and Malik [66] reviewed applications in solar PV systems, energy storage, and wind farm design. Similarly, Das et al. [67] addressed techno-economic optimization of hybrid energy systems.
2) Machine learning and data-driven optimization: Metaheuristics play a critical role in ML tasks such as hyperparameter tuning, feature selection, and neural network optimization. Ray et al. [68] reviewed feature selection techniques, while Akay et al. [69] analyzed deep neural network architectures and their integration with metaheuristics. Nematzadeh et al. [70] demonstrated that Grey Wolf Optimization (GWO) can outperform traditional grid search in hyperparameter tuning. Furthermore, Talbi [4] highlighted the synergy between ML and metaheuristics, emphasizing emerging hybrid approaches.
3) Logistics, transportation, and scheduling: Metaheuristics are widely used in routing, scheduling, and resource allocation problems. Classical algorithms such as Tabu Search and Simulated Annealing are frequently applied in these domains. Pereira et al. [71] developed an improved algorithm for the cold chain vehicle routing problem, while Sadeghi-Moghaddam et al. [72] addressed the fixed charge transportation problem using WOA. Additionally, Juntama et al. [73] proposed a reinforcement learning-based hyperheuristic that significantly reduced airspace complexity.
4) Smart systems, internet of things (IoT), and networks: In smart systems and IoT environments, metaheuristics are used for energy optimization, traffic prediction, and network design. Iwendi et al. [74] optimized IoT energy consumption using a hybrid WOA–Simulated Annealing (SA) approach. Alizadeh et al. [75] combined deep learning models with Firefly–BAT algorithms for traffic forecasting. Similarly, Fanian and Rafsanjani [76] proposed a fuzzy-metaheuristic clustering approach to improve energy efficiency and network lifetime in wireless sensor networks.
5) Finance, risk, and operations management: Metaheuristics are also applied in financial optimization and operations management. Doering et al. [77] reviewed their use in portfolio optimization and risk management. These methods are also employed in job scheduling, inventory control, and resource planning problems.
6) Environmental and scientific modeling: Metaheuristics are increasingly used in environmental modeling and scientific applications. Liu et al. [78] combined ML and metaheuristics for optimizing recycled aggregate concrete. Oliveira et al. [79] reviewed evolutionary optimization methods for greenhouse climate control, emphasizing multi-objective optimization trends.
7) Robotics and autonomous systems: In robotics, metaheuristics are used for path planning, swarm coordination, and motion control. Fong et al. [80] provided a taxonomy of metaheuristic applications in robotics. Kiani et al. [81] proposed improved GWO variants for 3D path planning, achieving significant performance improvements.
8) Cybersecurity and software engineering: Metaheuristics are applied in intrusion detection, firewall optimization, and software engineering tasks. Savanović et al. [82] combined ML with a modified Firefly algorithm for IoT security. Diaba et al. [83] optimized deep learning models for cyberattack detection. In software engineering, Khan et al. [84] used metaheuristics to improve software effort estimation models.

These application domains highlight the versatility of metaheuristics across both traditional optimization problems and modern data-driven systems. A key observation is the growing integration of ML techniques, particularly in dynamic and complex environments. This trend reinforces the importance of hybrid approaches, such as learnheuristics, in addressing real-world optimization challenges.

5. Discussions on Challenges and Open Research Issues

This section discusses several important issues surrounding recent metaheuristic algorithms, including similarities among optimization methods, parameter sensitivity, the tendency to become trapped in local optima, and the use of binary representations.

The increasing number of newly proposed metaheuristics has also led to growing criticism concerning their similarity to previously established algorithms. Although newly introduced methods may use different mathematical formulations or biological metaphors, the underlying search mechanisms can often be closely related to those of existing approaches. Table 2 summarizes some of these relationships and illustrates the extent to which commonly used search strategies have been adopted or adapted in recent algorithms. In particular, mechanisms originally associated with widely used methods such as Particle Swarm Optimization (PSO), GWO, and WOA appear repeatedly in newer approaches.

Parameter settings are another important factor that can substantially affect the performance of metaheuristic algorithms. Some methods, such as the EO, Snake Optimizer, and Slime Mold Algorithm, use relatively few control parameters and therefore require less effort for configuration. This can simplify implementation and reduce the computational cost associated with parameter tuning. However, a smaller number of parameters does not necessarily translate into better performance across all problem types, particularly when the optimization landscape is highly complex or diverse.

In contrast, algorithms such as the AO, BOA, and HBA rely on a larger set of parameters. These additional parameters can provide greater flexibility in controlling the search process and may allow the algorithm to adapt more effectively to specific optimization problems. At the same time, they increase the number of settings that need to be considered and may make the algorithm more sensitive to parameter choices. Consequently, achieving good performance may require more extensive experimentation and careful parameter calibration.

Algorithms such as Marine Predators Algorithm and MRFO occupy a relatively intermediate position, relying on a limited number of influential parameters while maintaining comparatively straightforward tuning procedures. Overall, the choice between parameter-light and parameter-rich metaheuristics should not be viewed simply as a matter of minimizing or maximizing the number of parameters. Rather, it involves balancing ease of configuration, adaptability, and the effort required for effective tuning. Algorithms with fewer parameters generally reduce the configuration burden and are easier to deploy, whereas parameter-rich methods may offer greater flexibility and control over the search process. However, this additional flexibility can also increase sensitivity to parameter settings and make performance more dependent on appropriate configuration.

Escaping local optima remains a central challenge. Hybridizing global and local search improves exploration and refinement, while memory mechanisms help prevent revisiting poor regions. Randomization—such as accepting worse moves—enhances search diversity. Population diversity techniques (crowding, niche sharing) and adaptive control of population size further reduce stagnation. Learning-based and memory-driven hybrid models increasingly support dynamic escape from local optima.

Binary encoding (discretization) converts continuous variables for binary decision tasks such as feature selection. S-shape functions offer smooth transitions, supporting gradual exploration, while V-shape functions enforce sharper thresholds for clearer decisions. Both approaches help metaheuristics navigate continuous landscapes while satisfying binary constraints, with the choice depending on problem needs.

5.1. Multi-objectivity

Many real-world problems require optimizing conflicting objectives. Multi-objective metaheuristics such as Non-dominated Sorting Genetic Algorithm (NSGA-II), Multi-Objective Evolutionary Algorithm Based on Decomposition (MOEA/D), and Multi-Objective Particle Swarm Optimization (MOPSO) approximate Pareto fronts by maintaining diversity and convergence. Recent ML advances—surrogate modeling, reinforcement learning, and hybrid frameworks—strengthen metaheuristic performance and enable applications such as hyperparameter tuning, neural architecture search, and multi-metric optimization. The growing integration of learning and metaheuristics is shaping new directions in intelligent, adaptive optimization for complex, data-driven systems.

5.2. Benchmarking challenges and common evaluation practices

A persistent challenge in metaheuristic research is the lack of standardized and reproducible benchmarking practices. Although many studies report that newly proposed algorithms outperform existing methods, meaningful comparisons across different studies are often difficult to establish. Differences in experimental configurations, benchmark problems, performance measures, and implementation details can have a substantial effect on the reported results.

Most recent metaheuristics are assessed using established benchmark suites, particularly the test functions developed by the Congress on Evolutionary Computation (CEC). These suites typically include unimodal, multimodal, hybrid, and composition functions that are intended to examine different aspects of an algorithm's search behavior, including its exploration and exploitation capabilities. In addition to these synthetic benchmarks, UCI datasets are frequently used to evaluate metaheuristics in feature selection and classification tasks, while real-world engineering and application-specific problems provide an opportunity to assess their practical effectiveness.

Despite the widespread use of these benchmarks, considerable inconsistencies remain in the way experiments are conducted and reported. Factors such as the number of function evaluations, population size, parameter tuning procedures, stopping criteria, and the number of independent runs can vary substantially between studies. Furthermore, statistical validation is not always included, making it difficult to determine whether observed performance differences are statistically meaningful or simply the result of experimental variability. To improve the reliability of reported findings, recent research has emphasized the use of standardized benchmark suites, such as CEC2017 and CEC2020, together with appropriate statistical procedures, including the Wilcoxon signed-rank and Friedman tests. Performing multiple independent runs is also important for assessing the robustness of stochastic optimization algorithms.

In this survey, we do not attempt to establish a single unified experimental ranking of all reviewed algorithms. Such a comparison would require the reimplementation of a large number of methods and their evaluation under carefully controlled and identical experimental conditions, which is beyond the scope of the present study. Instead, we adopt a qualitative comparative framework based on algorithmic properties that are consistently discussed or reported in the literature, including convergence behavior, parameter sensitivity, and robustness. This approach allows us to compare a relatively large number of recent metaheuristics while avoiding potentially misleading conclusions arising from inconsistent experimental settings across different studies.

Future research would benefit from greater standardization in the evaluation of newly proposed metaheuristics. In particular, the development of common benchmarking protocols, transparent reporting guidelines, and open-source evaluation platforms could facilitate more reliable and reproducible comparisons. Such efforts would also make it easier to distinguish genuine methodological advances from improvements that may primarily result from differences in experimental design or parameter configuration.

6. Conclusion

The development of new metaheuristic algorithms is likely to continue, driven by the broad range of complex optimization problems that arise in science, engineering, and real-world decision-making. The diversity, high dimensionality, and nonlinearity of these problems provide substantial opportunities for developing new search strategies, hybrid approaches, and problem-specific adaptations. At the same time, the growing incorporation of ML within optimization has opened new horizons for creating self-adaptive algorithms that continuously modify their search behavior based on data gathered during the optimization process or from previous optimization experiences.

In this evolving context, learnheuristics represent a particularly promising research avenue. By seamlessly embedding predictive, classification, and reinforcement learning mechanisms into metaheuristic frameworks, learnheuristics utilize previously acquired knowledge to intelligently guide exploration and exploitation, dynamically tune internal control parameters, and effectively respond to environmental changes in dynamic combinatorial optimization. Such data-driven mechanisms significantly enhance the adaptability and robustness of heuristics, especially in uncertain, noisy, and real-time environments.

However, the rapid influx of newly proposed metaheuristics—often framed around novel biological or physical metaphors—presents major methodological challenges. A critical analysis reveals that many recent algorithms rely on underlying search operators remarkably similar to established techniques such as PSO, GWO, and Differential Evolution. This metaphorical saturation makes it increasingly difficult to discern genuine algorithmic novelty from incremental repackaging. Consequently, the long-term relevance and academic footprint of a metaheuristic will not depend merely on competitive empirical performance over standard benchmark suites (e.g., CEC benchmark functions). Instead, its lasting contribution will be determined by its ability to generalize across diverse problem settings, demonstrate mathematical rigor and theoretical novelty, offer efficient parameter configurations, and gain widespread open-source adoption within the broader operational research and engineering communities.

The primary objective of this review study is to provide a critical, structured, and focused synthesis of the most influential metaheuristics introduced between 2019 and 2026. By rigorously dissecting their mathematical mechanisms, parameter sensitivity, structural similarities, discretization approaches (binary encodings), and real-world application footprints, this study offers researchers a transparent overview of recent developments while establishing a comprehensive five-dimensional taxonomy for learnheuristics.

Looking ahead, several vital research directions and technological synergies are expected to shape the future of metaheuristic optimization:

Synergy with Generative AI and Foundation Models: The convergence of metaheuristics with large language models and generative AI introduces revolutionary opportunities for automated algorithm design, dynamic operator selection, and intelligent solution initialization. Generative models can act as hyper-heuristics, generating novel search operators on-the-fly tailored to specific problem landscapes.

Adaptive Parallelism and Cloud/GPU Computing: As evaluation costs for objective functions rise in high-dimensional engineering and deep learning tasks, distributed and parallel metaheuristics become essential. Future efforts must prioritize multi-core GPU acceleration, asynchronous island models, and adaptive cloud-based resource allocation frameworks to overcome computational bottlenecks and enhance population diversity.

Standardized Benchmarking and Open Science: To address persistent evaluation flaws, the research community must transition toward standardized benchmarking protocols, transparent statistical testing (e.g., non-parametric Wilcoxon/Friedman tests), open-source code availability, and real-world industrial validation rather than over-relying on synthetic unimodal/multimodal functions.

In conclusion, as optimization problems grow increasingly data-intensive and dynamic, the field is transitioning from purely metaphor-driven metaheuristics toward intelligent, hybrid, parallelized, and self-learning optimization ecosystems. Embracing rigorous comparative methodologies and deeper integration with artificial intelligence will ensure that future optimization tools deliver meaningful, sustainable, and high-impact solutions to real-world industrial challenges.

Acknowledgment

We would like to thank Özlem Tekdemir Dökeroglu for her artistic illustrations in Figure 6.

This manuscript was previously made available as a preprint on arXiv [85]. We acknowledge the preprint version of this work.

Ethical Statement

This study does not contain any studies with human or animal subjects performed by any of the authors.

Conflicts of Interest

The authors declare that they have no conflicts of interest to this work.

Data Availability Statement

Data sharing is not applicable to this article, as no new data were created or analyzed in this study.

Author Contribution Statement

Tansel Dokeroglu: Conceptualization, Methodology, Software, Formal analysis, Investigation, Resources, Writing – original draft, Writing – review & editing, Supervision, Project administration. **Deniz Canturk:** Conceptualization, Methodology, Software, Validation, Investigation, Resources, Data curation, Writing – original draft, Visualization, Supervision, Project administration.

Declaration of generative AI and AI-assisted technologies in the writing process

During the preparation of this work, the author(s) used the Quillbot AI text tool to improve the quality of the sentences written by the authors. After using this tool/service, the author(s) reviewed and edited the content as needed and took full responsibility for the content of the publication.

References

[1] Agrawal, P., Abutarboush, H. F., Ganesh, T., & Mohamed, A. W. (2021). Metaheuristic algorithms on feature selection: A survey of one decade of research (2009–2019). *IEEE Access*, *9*, 26766–26791. https://doi.org/10.1109/ACCESS.2021.3056407

[2] Abed-Alguni, B. H. (2026). Evomapx: An explainable framework for metaheuristic optimization algorithms. *Expert Systems with Applications*, *298*, 129514. https://doi.org/10.1016/j.eswa.2025.129514

[3] Hou, Q., Shao, Q., Heidari, A. A., Liu, L., Chen, H., & Liang, G. (2026). Integrating diversity-integrated weighted ranking in metaheuristic algorithms for medical applications. *Expert Systems with Applications*, *311*, 131301. https://doi.org/10.1016/j.eswa.2026.131301

[4] Talbi, E. G. (2021). Machine learning into metaheuristics: A survey and taxonomy. *ACM Computing Surveys*, *54*(6), 1–32. https://doi.org/10.1145/3459664

[5] Tripathy, B., Reddy Maddikunta, P. K., Pham, Q. V., Gadekallu, T. R., Dev, K., Pandya, S., & ElHalawany, B. M. (2022). Harris hawk optimization: a survey on variants and applications. *Computational Intelligence and Neuroscience*, 2218594. https://doi.org/10.1155/2022/2218594 *2022*(1)

[6] Makhadmeh, S. N., Al-Betar, M. A., Abasi, A. K., Awadallah, M. A., Doush, I. A., Alyasseri, Z. A. A., . . . , & Alomari, O. A. (2023). Recent advances in butterfly optimization algorithm, its versions and applications. *Archives of Computational Methods in Engineering*, *30*(2), 1399–1420. https://doi.org/10.1007/s11831-022-09843-3

[7] Deb, S., Abdelminaam, D. S., Said, M., & Houssein, E. H. (2021). Recent methodology-based gradient-based optimizer for economic load dispatch problem. *IEEE Access*, *9*, 44322–44338. https://doi.org/10.1109/ACCESS.2021.3066329

[8] Gharehchopogh, F. S., Ucan, A., Ibrikci, T., Arasteh, B., & Isik, G. (2023). Slime mould algorithm: A comprehensive survey of its variants and applications. *Archives of Computational Methods in Engineering*, *30*(4), 2683–2723. https://doi.org/10.1007/s11831-023-09883-3

[9] Mugemanyi, S., Qu, Z., Rugema, F. X., Dong, Y., Wang, L., Bananeza, C., . . . , & Mutabazi, E. (2023). Marine predators algorithm: A comprehensive review. *Machine Learning with Applications*, *12*, 100471. https://doi.org/10.1016/j.mlwa.2023.100471

[10] Al-Betar, M. A., Abu Doush, I., Makhadmeh, S. N., Al-Naymat, G., Alomari, O. A., & Awadallah, M. A. (2024). Equilibrium optimizer: A comprehensive survey. *Multimedia Tools and Applications*, *83*(10), 29617–29666. https://doi.org/10.1007/s11042-023-16764-1

[11] Abualigah, L., Yousri, D., Abd Elaziz, M., Ewees, A. A., Al-Qaness, M. A., & Gandomi, A. H. (2021). Aquila optimizer: A novel meta-heuristic optimization algorithm. *Computers & Industrial Engineering*, *157*, 107250. https://doi.org/10.1016/j.cie.2021.107250

[12] Dhiman, G., Singh, K. K., Slowik, A., Chang, V., Yildiz, A. R., Kaur, A., & Garg, M. (2021). Emosoa: A new evolutionary multi-objective seagull optimization algorithm for global optimization. *International Journal of Machine Learning and Cybernetics*, *12*(2), 571–596. https://doi.org/10.1007/s13042-020-01189-1

[13] Gharehchopogh, F. S., Ghafouri, S., Namazi, M., & Arasteh, B. (2024). Advances in manta ray foraging optimization: A comprehensive survey. *Journal of Bionic Engineering*, *21*(2), 953–990. https://doi.org/10.1007/s42235-024-00481-y

[14] Qian, L., Khishe, M., Huang, Y., & Mirjalili, S. (2024). Sebchoa: An improved chimp optimization algorithm using spiral exploitation behavior. *Neural Computing and Applications*, *36*(9), 4763–4786. https://doi.org/10.1007/s00521-023-09236-y

[15] Oladipo, S., & Sun, Y. (2026). A concise look at the honey badger algorithm. *International Journal of Swarm Intelligence Research (IJSIR)*, *17*(1), 1–34. https://doi.org/10.4018/IJSIR.403422

[16] Hashim, F. A., Hussain, K., Houssein, E. H., Mabrouk, M. S., & Al-Atabany, W. (2021). Archimedes optimization algorithm: A new metaheuristic algorithm for solving optimization problems. *Applied Intelligence*, *51*, 1531–1551. https://doi.org/10.1007/s10489-020-01893-z

[17] Jena, B., Naik, M. K., Wunnava, A., & Panda, R. (2021). A differential squirrel search algorithm. In *Advances in Intelligent Computing and Communication: Proceedings of ICAC 2020*, 143–152. https://doi.org/10.1007/978-981-16-0695-3_15

[18] Houssein, E. H., Helmy, B. E. D., Elngar, A. A., Abdelminaam, D. S., & Shaban, H. (2021). An improved tunicate swarm algorithm for global optimization and image segmentation. *IEEE Access*, *9*, 56066–56092. https://doi.org/10.1109/ACCESS.2021.3072336

[19] Hashim, F. A., Houssein, E. H., Mabrouk, M. S., Al-Atabany, W., & Mirjalili, S. (2019). Henry gas solubility optimization: A novel physics-based algorithm. *Future Generation Computer Systems*, *101*, 646–667. https://doi.org/10.1016/j.future.2019.07.015

[20] Juan, A. A., Men, Y., Medina, V., & Escoto, M. (2026). Solving a multi-period dynamic pricing problem using learning-augmented exact methods and learnheuristics. *Algorithms*, *19*(4), 284. https://doi.org/10.3390/a19040284

[21] Eleazar, C., & Gomez, J. F. (2026). Enhancing hotel profitability: Dynamic pricing with a sim-learnheuristic approach. *International Journal of Hospitality Management*, *133*, 104472. https://doi.org/10.1016/j.ijhm.2025.104472

[22] Hussain, K., Mohd Salleh, M. N., Cheng, S., & Shi, Y. (2019). Metaheuristic research: A comprehensive survey. *Artificial Intelligence Review*, *52*, 2191–2233. https://doi.org/10.1007/s10462-017-9605-z

[23] Halim, A. H., Ismail, I., & Das, S. (2021). Performance assessment of the metaheuristic optimization algorithms: an exhaustive review. *Artificial Intelligence Review*, *54*(3), 2323–2409. https://doi.org/10.1007/s10462-020-09906-6

[24] Wong, W., & Ming, C. I. (2019). A review on metaheuristic algorithms: recent trends, benchmarking and applications. In *2019 7th International Conference on Smart Computing & Communications (ICSCC)*, 1–5. https://doi.org/10.1109/ICSCC.2019.8843624

[25] Agushaka, J. O., & Ezugwu, A. E. (2022). Initialisation approaches for population-based metaheuristic algorithms: a comprehensive review. *Applied Sciences*, *12*(2), 896. https://doi.org/10.3390/app12020896

[26] Das, A., Rai, R., Sasmal, B., Dhal, K. G., Khurma, R. A., & Saha, R. (2025). Metaheuristic algorithms since 2020: Development, taxonomy, analysis, and applications. *Archives of Computational Methods in Engineering*, 1–69. https://doi.org/10.1007/s11831-025-10408-3

[27] Tomar, V., Bansal, M., & Singh, P. (2024). Metaheuristic algorithms for optimization: A brief review.

Engineering Proceedings, *59*(1), 238. https://doi.org/10.3390/engproc2023059238

[28] Ezugwu, A. E., Shukla, A. K., Nath, R., Akinyelu, A. A., Agushaka, J. O., Chiroma, H., & Muhuri, P. K. (2021). Metaheuristics: A comprehensive overview and classification along with bibliometric analysis. *Artificial Intelligence Review*, *54*, 4237–4316. https://doi.org/10.1007/s10462-020-09952-0

[29] Hussain, K., Salleh, M. N. M., Cheng, S., & Shi, Y. (2019). On the exploration and exploitation in popular swarm-based metaheuristic algorithms. *Neural Computing and Applications*, *31*(11), 7665–7683. https://doi.org/10.1007/s00521-018-3592-0

[30] Pellerin, R., Perrier, N., & Berthaut, F. (2020). A survey of hybrid metaheuristics for the resource-constrained project scheduling problem. *European Journal of Operational Research*, *280*(2), 395–416. https://doi.org/10.1016/j.ejor.2019.01.063

[31] Mohammed, H. M., Umar, S. U., & Rashid, T. A. (2019). A systematic and meta-analysis survey of whale optimization algorithm. *Computational Intelligence and Neuroscience*, 8718571. https://doi.org/10.1155/2019/8718571 *2019*(1)

[32] Abualigah, L., Elaziz, M. A., Khasawneh, A. M., Alshinwan, M., Ibrahim, R. A., Al-Qaness, M. A., . . . , & Gandomi, A. H. (2022). Meta-heuristic optimization algorithms for solving real-world mechanical engineering design problems: a comprehensive survey, applications, comparative analysis, and results. *Neural Computing and Applications*, *34*, 4081–4110. https://doi.org/10.1007/s00521-021-06747-4

[33] Dokeroglu, T., Deniz, A., & Kiziloz, H. E. (2022). A comprehensive survey on recent metaheuristics for feature selection. *Neurocomputing*, *494*, 269–296. https://doi.org/10.1016/j.neucom.2022.04.083

[34] Juan, A. A., Keenan, P., Martí, R., McGarraghy, S., Panadero, J., Carroll, P., & Oliva, D. (2023). A review of the role of heuristics in stochastic optimisation: From metaheuristics to learnheuristics. *Annals of Operations Research*, *320*(2), 831–861. https://doi.org/10.1007/s10479-021-04142-9

[35] Calvet, L., Armas, J. d., Masip, D., & Juan, A. A. (2017). Learnheuristics: hybridizing metaheuristics with machine learning for optimization with dynamic inputs. *Open Mathematics*, *15*(1), 261–280. https://doi.org/10.1515/math-2017-0029

[36] Kamboj, V. K., Nandi, A., Bhadoria, A., & Sehgal, S. (2020). An intensify harris hawks optimizer for numerical and engineering optimization problems. *Applied Soft Computing*, *89*, 106018. https://doi.org/10.1016/j.asoc.2019.106018

[37] Daoud, M. S., Shehab, M., Al-Mimi, H. M., Abualigah, L., Zitar, R. A., & Shambour, M. K. Y. (2023). Gradient-based optimizer (GBO): A review, theory, variants, and applications. *Archives of Computational Methods in Engineering*, *30*(4), 2431–2449. https://doi.org/10.1007/s11831-022-09872-y

[38] Premkumar, M., Jangir, P., & Sowmya, R. (2021). MOGBO: A new multiobjective gradient-based optimizer for real-world structural optimization problems. *Knowledge-Based Systems*, *218*, 106856. https://doi.org/10.1016/j.knosys.2021.106856

[39] Jiang, Y., Luo, Q., Wei, Y., Abualigah, L., & Zhou, Y. (2021). An efficient binary gradient-based optimizer for feature selection. *Mathematical Biosciences and Engineering*, *18*(4), 3813–3854. https://doi.org/10.3934/mbe.2021192

[40] Chen, H., Li, C., Mafarja, M., Heidari, A. A., Chen, Y., & Cai, Z. (2023). Slime mould algorithm: a comprehensive review of recent variants and applications. *International Journal of Systems Science*, *54*(1), 204–235. https://doi.org/10.1080/00207721.2022.2153635

[41] Premkumar, M., Jangir, P., Sowmya, R., Alhelou, H. H., Heidari, A. A., & Chen, H. (2020). Mosma: Multi-objective slime mould algorithm based on elitist non-dominated sorting. *IEEE Access*, *9*, 3229–3248. https://doi.org/10.1109/ACCESS.2020.3047936

[42] Su, Y., & Liu, Y. (2026). A novel marine predator whale optimization algorithm for global numerical optimization. *Engineering Optimization*, *58*(1), 203–239. https://doi.org/10.1080/0305215X.2025.2464862

[43] Ramezani, M., Bahmanyar, D., & Razmjooy, N. (2021). A new improved model of marine predator algorithm for optimization problems. *Arabian Journal for Science and Engineering*, *46*(9), 8803–8826. https://doi.org/10.1007/s13369-021-05688-3

[44] Makhadmeh, S. N., Al-Betar, M. A., Assaleh, K., & Kassaymeh, S. (2022). A hybrid white shark equilibrium optimizer for power scheduling problem based IoT. *IEEE Access*, *10*, 132212–132231. https://doi.org/10.1109/ACCESS.2022.3229434

[45] Wang, J., Yang, B., Li, D., Zeng, C., Chen, Y., Guo, Z., . . . , & Yu, T. (2021). Photovoltaic cell parameter estimation based on improved equilibrium optimizer algorithm. *Energy Conversion and Management*, *236*, 114051. https://doi.org/10.1016/j.enconman.2021.114051

[46] Sasmal, B., Hussien, A. G., Das, A., & Dhal, K. G. (2023). A comprehensive survey on aquila optimizer. *Archives of Computational Methods in Engineering*, *30*(7), 4449–4476. https://doi.org/10.1007/s11831-023-09945-6

[47] Al-qaness, M. A., Ewees, A. A., Fan, H., AlRassas, A. M., & Abd Elaziz, M. (2022). Modified aquila optimizer for forecasting oil production. *Geo-Spatial Information Science*, *25*(4), 519–535. https://doi.org/10.1080/10095020.2022.2068385

[48] Mahajan, S., Abualigah, L., Pandit, A. K., & Altalhi, M. (2022). Hybrid aquila optimizer with arithmetic optimization algorithm for global optimization tasks. *Soft Computing*, *26*(10), 4863–4881. https://doi.org/10.1007/s00500-022-06873-8

[49] Panagant, N., Pholdee, N., Bureerat, S., Kaen, K., Yıldız, A. R., & Sait, S. M. (2020). Seagull optimization algorithm for solving real-world design optimization problems. *Materials Testing*, *62*(6), 640–644. https://doi.org/10.3139/120.111529

[50] Tang, A., Zhou, H., Han, T., & Xie, L. (2021). A modified manta ray foraging optimization for global optimization problems. *IEEE Access*, *9*, 128702–128721. https://doi.org/10.1109/ACCESS.2021.3113323

[51] Houssein, E. H., Emam, M. M., & Ali, A. A. (2021). Improved manta ray foraging optimization for multi-level thresholding using COVID-19 CT images. *Neural Computing and Applications*, *33*(24), 16899–16919. https://doi.org/10.1007/s00521-021-06273-3

[52] Khishe, M., Nezhadshahbodaghi, M., Mosavi, M. R., & Martín, D. (2021). A weighted chimp optimization algorithm. *IEEE Access*, *9*, 158508–158539. https://doi.org/10.1109/ACCESS.2021.3130933

[53] Jia, H., Sun, K., Zhang, W., & Leng, X. (2021). An enhanced chimp optimization algorithm for continuous optimization domains. *Complex & Intelligent Systems*, *8*, 65–82. https://doi.org/10.1007/s40747-021-00346-5

[54] Duzenli, T., Onay, F. K., & Aydemir, S. B. (2022). Improved honey badger algorithms for parameter extraction in

photovoltaic models. *Optik*, *268*, 169731. https://doi.org/10.1016/j.ijleo.2022.169731

[55] Xu, Y., Zhong, R., Cao, Y., Zhang, C., & Yu, J. (2025). Symbiotic mechanism-based honey badger algorithm for continuous optimization. *Cluster Computing*, *28*(2), 133. https://doi.org/10.1007/s10586-024-04765-0

[56] Sakthivel, V., Suman, M., & Sathya, P. (2021). Combined economic and emission power dispatch problems through multi-objective squirrel search algorithm. *Applied Soft Computing*, *100*, 106950. https://doi.org/10.1016/j.asoc.2020.106950

[57] Kalyan, C. N. S., Goud, B. S., Kumar, M. K., Thakur, P., Bajaj, M., & Bansal, R. C. (2025). Squirrel search algorithm based intelligent controller for interconnected power system. *International Journal of Modelling and Simulation*, *45*(2), 420–440. https://doi.org/10.1080/02286203.2023.2205989

[58] Yıldız, B. S., Pholdee, N., Bureerat, S., Erdaş, M. U., Yıldız, A. R., & Sait, S. M. (2021). Comparison of the political optimization algorithm, the Archimedes optimization algorithm and the Levy flight algorithm for design optimization in industry. *Materials Testing*, *63*(4), 356–359. https://doi.org/10.1515/mt-2020-0053

[59] Akdag, O. (2022). A improved Archimedes optimization algorithm for multi/single-objective optimal power flow. *Electric Power Systems Research*, *206*, 107796. https://doi.org/10.1016/j.epsr.2022.107796

[60] Rizk-Allah, R. M., Saleh, O., Hagag, E. A., & Mousa, A. A. A. (2021). Enhanced tunicate swarm algorithm for solving large-scale nonlinear optimization problems. *International Journal of Computational Intelligence Systems*, *14*(1), 189. https://doi.org/10.1007/s44196-021-00039-4

[61] Karasu, S., & Altan, A. (2022). Crude oil time series prediction model based on LSTM network with chaotic Henry gas solubility optimization. *Energy*, *242*, 122964. https://doi.org/10.1016/j.energy.2021.122964

[62] Yıldız, B. S., Pholdee, N., Panagant, N., Bureerat, S., Yildiz, A. R., & Sait, S. M. (2021). A novel chaotic Henry gas solubility optimization algorithm for solving real-world engineering problems. *Engineering with Computers*, *38*, 871–883. https://doi.org/10.1007/s00366-020-01268-5

[63] Dhiman, G. (2021). Esa: A hybrid bio-inspired metaheuristic optimization approach for engineering problems. *Engineering with Computers*, *37*, 323–353. https://doi.org/10.1007/s00366-019-00826-w

[64] Xu, D., & Yin, J. (2023). An improved black widow optimization algorithm for engineering constrained optimization problems. *IEEE Access*, *11*, 32476–32495. https://doi.org/10.1109/ACCESS.2023.3262600

[65] Güven, A. F., & Samy, M. M. (2022). Performance analysis of autonomous green energy system based on multi and hybrid metaheuristic optimization approaches. *Energy Conversion and Management*, *269*, 116058. https://doi.org/10.1016/j.enconman.2022.116058

[66] Minai, A. F., & Malik, H. (2021). Metaheuristics paradigms for renewable energy systems: advances in optimization algorithms. *Metaheuristic and Evolutionary Computation: Algorithms and Applications*, 35–61. https://doi.org/10.1007/978-981-15-7571-6_2

[67] Das, M., Singh, M. A. K., & Biswas, A. (2019). Techno-economic optimization of an off-grid hybrid renewable energy system using metaheuristic optimization approaches–case of a radio transmitter station in India. *Energy Conversion and Management*, *185*, 339–352. https://doi.org/10.1016/j.enconman.2019.01.107

[68] Ray, P., Reddy, S. S., & Banerjee, T. (2021). Various dimension reduction techniques for high dimensional data analysis: a review. *Artificial Intelligence Review*, *54*(5), 3473–3515. https://doi.org/10.1007/s10462-020-09928-0

[69] Akay, B., Karaboga, D., & Akay, R. (2022). A comprehensive survey on optimizing deep learning models by metaheuristics. *Artificial Intelligence Review*, *55*(2), 829–894. https://doi.org/10.1007/s10462-021-09992-0

[70] Nematzadeh, S., Kiani, F., Torkamanian-Afshar, M., & Aydin, N. (2022). Tuning hyperparameters of machine learning algorithms and deep neural networks using metaheuristics: A bioinformatics study on biomedical and biological cases. *Computational Biology and Chemistry*, *97*, 107619. https://doi.org/10.1016/j.compbiolchem.2021.107619

[71] Pereira, V. G., Alves-Junior, O. C., & Baldo, F. (2024). An approach to solve the heterogeneous fixed fleet vehicle routing problem with time window based on adaptive large neighborhood search meta-heuristic. *IEEE Transactions on Intelligent Transportation Systems*, *25*(7), 8148–8157. https://doi.org/10.1109/TITS.2024.3386457

[72] Sadeghi-Moghaddam, S., Hajiaghaei-Keshteli, M., & Mahmoodjanloo, M. (2019). New approaches in metaheuristics to solve the fixed charge transportation problem in a fuzzy environment. *Neural Computing and Applications*, *31*(Suppl 1), 477–497. https://doi.org/10.1007/s00521-017-3027-3

[73] Juntama, P., Delahaye, D., Chaimatanan, S., & Alam, S. (2022). Hyperheuristic approach based on reinforcement learning for air traffic complexity mitigation. *Journal of Aerospace Information Systems*, *19*(9), 633–648. https://doi.org/10.2514/1.I011048

[74] Iwendi, C., Maddikunta, P. K. R., Gadekallu, T. R., Lakshmanna, K., Bashir, A. K., & Piran, M. J. (2021). A metaheuristic optimization approach for energy efficiency in the IoT networks. *Software: Practice and Experience*, *51*(12), 2558–2571. https://doi.org/10.1002/spe.2797

[75] Alizadeh, M., Beheshti, M. T., Ramezani, A., & Bolouki, S. (2023). An optimized hybrid methodology for short-term traffic forecasting in telecommunication networks. *Transactions on Emerging Telecommunications Technologies*, *34*(12), e4860. https://doi.org/10.1002/ett.4860

[76] Fanian, F., & Rafsanjani, M. K. (2023). Cfmcrs: Calibration fuzzy-metaheuristic clustering routing scheme simultaneous in on-demand wrsns for sustainable smart city. *Expert Systems with Applications*, *211*, 118619. https://doi.org/10.1016/j.eswa.2022.118619

[77] Doering, J., Kizys, R., Juan, A. A., Fito, A., & Polat, O. (2019). Metaheuristics for rich portfolio optimisation and risk management: Current state and future trends. *Operations Research Perspectives*, *6*, 100121. https://doi.org/10.1016/j.orp.2019.100121

[78] Liu, K., Zheng, J., Dong, S., Xie, W., & Zhang, X. (2023). Mixture optimization of mechanical, economical, and environmental objectives for sustainable recycled aggregate concrete based on machine learning and metaheuristic algorithms. *Journal of Building Engineering*, *63*, 105570. https://doi.org/10.1016/j.jobe.2022.105570

[79] Oliveira, P. M., Solteiro Pires, E., Boaventura-Cunha, J., & Pinho, T. M. (2020). Review of nature and biologically inspired metaheuristics for greenhouse environment control.

Transactions of the Institute of Measurement and Control, *42*(12), 2338–2358. https://doi.org/10.1177/0142331220909010

[80] Fong, S., Deb, S., & Chaudhary, A. (2015). A review of metaheuristics in robotics. *Computers & Electrical Engineering*, *43*, 278–291. https://doi.org/10.1016/j.compeleceng.2015.01.009

[81] Kiani, F., Seyyedabbasi, A., Nematzadeh, S., Candan, F., Çevik, T., Anka, F. A., . . . , & Muzirafuti, A. (2022). Adaptive metaheuristic-based methods for autonomous robot path planning: sustainable agricultural applications. *Applied Sciences*, *12*(3), 943. https://doi.org/10.3390/app12030943

[82] Savanović, N., Toskovic, A., Petrovic, A., Zivkovic, M., Damaševičius, R., Jovanovic, L., . . . , & Nikolic, B. (2023). Intrusion detection in healthcare 4.0 internet of things systems via metaheuristics optimized machine learning. *Sustainability*, *15*(16), 12563. https://doi.org/10.3390/su151612563

[83] Diaba, S. Y., Shafie-Khah, M., & Elmusrati, M. (2023). Cyber security in power systems using meta-heuristic and deep learning algorithms. *IEEE Access*, *11*, 18660–18672. https://doi.org/10.1109/ACCESS.2023.3247193

[84] Khan, M. S., Jabeen, F., Ghouzali, S., Rehman, Z., Naz, S., & Abdul, W. (2021). Metaheuristic algorithms in optimizing deep neural network model for software effort estimation. *IEEE Access*, *9*, 60309–60327. https://doi.org/10.1109/ACCESS.2021.3072380

[85] Dokeroglu, T., Canturk, D., & Kucukyilmaz, T. (2025). A survey on pioneering metaheuristic algorithms between 2019 and 2024. https://doi.org/10.48550/arXiv.2501.14769 arXiv Preprint: 2501.14769

How to Cite: Dokeroglu, T., & Canturk, D. (2026). Advances in Metaheuristics and Learnheuristics (2019–2026): A Comprehensive Review, Taxonomy, and Future Directions. *Journal of Data Science and Intelligent Systems*. https://doi.org/10.47852/bonviewJDSIS62028069

Appendix A. Other recent metaheuristic algorithms proposed between 2019 and 2026

Table A1
Metaheuristic algorithms developed between 2019 and 2026, compiled from recent literature to highlight emerging trends and newly proposed optimization methods

Metaheuristic	Year	# citations
Mayfly Optimization	2020	1,970
Coati Optimization Algorithm	2023	1,720
African Vultures Optimization Algorithm	2021	1,620
Pelican Optimization Algorithm	2022	1,610
Dung Beetle Optimizer	2023	1,550
Golden Jackal Optimization	2022	1,430
Beluga Whale Optimization	2022	1,260
Chaos Game Optimization	2024	938
Gazelle Optimization Algorithm	2023	776
Sunflower Optimization	2019	1,580
Black Widow Optimization Algorithm	2020	1,470
Artificial Ecosystem-Based Optimization	2020	1,200
Artificial Gorilla Troops Optimizer	2021	1,200
Political Optimizer	2020	1,110
Sailfish Optimizer	2019	1,040
Snake Optimizer	2022	916
Remora Optimization	2021	816
Artificial Rabbits Optimization	2022	815
RUNge Kutta Optimizer	2021	801
Chameleon Swarm Algorithm	2021	790
Flow Direction Algorithm	2021	765
Wild Horse Optimizer	2022	745
Barnacles Mating Optimizer	2020	720
Horse Herd	2021	703
Bald Eagle Search Optimization	2020	667
Deer Hunting Optimization Algorithm	2019	605
Red Fox Optimization	2021	594
Gaining Sharing Knowledge-Based Algorithm	2020	593
Adam Optimization Algorithm	2019	589
Prairie Dog Optimization	2022	584
Transient Search Optimization	2020	570
Water Strider Algorithm	2020	557
Dandelion Optimizer	2022	554
White Shark Optimizer	2022	547
Golden Eagle Optimizer	2021	542
COOT Bird	2021	510
Capuchin Search Algorithm	2021	510
Ebola Optimization Search Algorithm	2022	491
Tuna Swarm Optimization	2021	486
Coronavirus Herd Immunity Optimizer	2021	459

(*Continued*)

Table A1
(*Continued*)

Metaheuristic	Year	# citations
Jellyfish in Ocean	2021	446
Atomic Orbital Search	2021	415
Spider Wasp Optimizer	2023	397
RIME: A Physics-Based Optimization	2023	375
Mountain Gazelle Optimizer	2022	362
Fire Hawk Optimizer	2023	359
Student Psychology-Based Optimization	2020	359
Pathfinder Algorithm	2019	329
Osprey Optimization Algorithm	2023	304
War Strategy Optimization Algorithm	2022	303
Lichtenberg Algorithm	2021	282
Poor and Rich Optimization Algorithm	2019	270
Emperor Penguins Colony	2019	267
Shuffled Shepherd Optimization	2020	264
Rain Optimization Algorithm	2020	253
Kepler Optimization Algorithm	2023	244
Adolescent Identity Search Algorithm	2020	244
Forensic-Based Investigation	2020	234
Dingo Optimizer	2021	232
Starling Murmuration Optimizer	2022	228
Gannet Optimization Algorithm	2022	223
Sea-Horse Optimizer	2023	222
Growth Optimizer	2023	219
Levy Flight Distribution	2020	211
Giza Pyramids Construction	2021	206
Giza Pyramids Construction	2021	205
Tasmanian Devil Optimization	2022	201
Orca Predation Algorithm	2022	194
Binary Chimp Optimization Algorithm	2021	193
Nomadic People Optimizer	2020	191
Cheetah Optimizer	2022	189
Golden Ratio Optimization	2020	184
Material Generation Algorithm	2021	174
Crystal Structure Algorithm (CryStAl)	2021	167
Stochastic Paint Optimizer	2022	166
Waterwheel Plant Algorithm	2023	161
Dynamic Differential Annealed Optimization	2020	158
Nutcracker Optimizer	2023	157
Carnivorous Plant Algorithm	2021	153
Strawberry Algorithm	2021	153
Liver Cancer Algorithm	2023	148
Energy Valley Optimizer	2023	146
Subtraction-Average-Based Optimizer	2023	143
Tiki-Taka Algorithm	2021	143

(*Continued*)

Table A1
(*Continued*)

Metaheuristic	Year	# citations
Newton Metaheuristic Algorithm	2020	143
Tornado Optimizer	2025	142
Walrus Optimization Algorithm	2023	138
Cuckoo Catfish Optimizer	2025	80
Goat Optimization Algorithm	2025	30
Logarithmic Mean-Based Optimization	2025	22
Farthest Better or Nearest Worse Optimizer	2026	18
Basketball Team Optimization	2025	18
Secant Optimization Algorithm	2026	15
Offensive Defensive Optimization	2012	12
Raindrop Optimizer	2025	11
Hare Escape Optimization	2025	5
Heart Rate Optimizer	2026	3