Cloud Drift Optimization (CDO) Algorithm: A Nature-Inspired Metaheuristic

Mohammad Alibabaei Shahraki

alibabaei_mohammad@yahoo.com

National University of Skills https://orcid.org/0000-0002-0707-7735

Research Article

Keywords: Optimization, Optimization techniques, Metaheuristic Algorithms, Constrained optimization, Algorithm

Posted Date: April 14th, 2025

DOI: https://doi.org/10.21203/rs.3.rs-6430604/v1

Additional Declarations: The authors declare no competing interests.

Cloud Drift Optimization (CDO) Algorithm: A Nature-Inspired Metaheuristic

Mohammad Alibabaei Shahraki[1]

[1] Department of Civil Engineering, National University of Skills (NUS), Tehran, Iran

Abstract

This study introduces the Cloud Drift Optimization (CDO) algorithm, an innovative nature-inspired metaheuristic approach to solving complex optimization problems. The CDO algorithm mimics the dynamic behavior of cloud particles influenced by atmospheric forces, striking a refined balance between exploration and exploitation. It features an adaptive weight adjustment mechanism that alters the cloud's drift behavior in real-time, allowing for efficient navigation through the search space. Using a cloud-based drift strategy, CDO harnesses probabilistic movements to maneuver through the optimization landscape more effectively. The algorithm has undergone rigorous testing against various established unimodal and multimodal benchmark functions, where it showcases outstanding performance characterized by faster convergence rates, high robustness, and exceptional solution accuracy compared to top contemporary optimization techniques. Additionally, CDO applies to numerous real-world engineering optimization tasks, such as designing cantilever beams, three-bar trusses, tension/compression springs, and pressure vessels. The empirical data highlight CDO's ability to deliver innovative solutions across engineering fields, machine learning applications, and other practical optimization scenarios. These results indicate that CDO is a promising tool for tackling highly complex and multidimensional problems in academic and industrial environments.

Keywords: Optimization; Optimization techniques; Metaheuristic Algorithms; Constrained optimization; Algorithm

1. Introduction

Optimization problems, particularly in engineering and computational sciences, have driven the development of advanced solution techniques. Traditional optimization methods often struggle with non-convex, high-dimensional, and computationally expensive problems. Consequently, researchers have increasingly turned to metaheuristic algorithms to tackle these challenges effectively [1,2]. Metaheuristics offer flexibility, adaptability, and robustness in handling diverse optimization tasks, and their popularity has surged in recent decades. Their

success in providing near-optimal solutions with reasonable computational effort has led to their widespread adoption across multiple disciplines, including structural engineering, renewable energy, healthcare, and financial modeling [3,4].

Metaheuristic algorithms draw inspiration from various natural, biological, and physical phenomena. Evolutionary algorithms (EAs) have been at the forefront of this development, with genetic algorithms [5] being among the earliest and most widely used. Other evolutionary techniques, such as evolutionary programming [6] and differential evolution [7], have further expanded the capabilities of these approaches. Multi-objective evolutionary algorithms have also been extensively studied to address optimization problems with conflicting objectives [8,9]. These methods have been applied in various fields, including mechanical design, civil infrastructure optimization, and renewable energy systems. Additionally, hybrid evolutionary approaches combining genetic algorithms with particle swarm optimization (PSO) and other heuristics have shown promising improvements in convergence speed and accuracy [10,11].

Swarm intelligence-based algorithms have emerged as another powerful class of metaheuristics inspired by the collective behavior of social animals. Particle swarm optimization [12], ant colony optimization [13], and arithmetic optimization algorithms[14] are prominent examples. More recent developments in swarm-based approaches include the grey wolf optimizer [15], whale optimization algorithm[16], and the salp swarm algorithm [17], all of which have demonstrated superior performance in solving complex optimization problems. These techniques have been successfully utilized in large-scale structural analysis, medical imaging, and smart grid optimization, showcasing their adaptability across multiple domains. Furthermore, hybrid models that integrate swarm intelligence with deep learning frameworks have been developed to enhance predictive accuracy in various applications, including climate modeling and financial forecasting [18,19].

Physics-based and chemistry-inspired algorithms have also contributed significantly to optimization research. Simulated annealing [20], and tabu search [21] introduced innovative search mechanisms based on thermodynamics and memory structures. More recent contributions, such as the water cycle algorithm [22], colliding bodies optimization [23], and gravitational search algorithm [24], leverage physical principles to enhance search efficiency. These methods have been extensively applied in aerodynamics, seismic design optimization, and bioinformatics, further expanding their applicability. Notably, physics-inspired hybrid models combining simulated annealing with other metaheuristic techniques have enhanced

computational efficiency in solving non-linear engineering problems [25,26].

Furthermore, bio-inspired metaheuristics have gained attention for their ability to mimic natural selection and ecological interactions. Cuckoo search [27], bat algorithm [28], firefly algorithm [29], and equilibrium optimizer [30] are notable examples that have been successfully applied across various domains. In addition, hybrid and ensemble approaches, such as the symbiotic organisms search [31] and chaotic hybrid multi-objective optimization [32], integrate multiple strategies to enhance convergence speed and solution quality. These techniques have enabled advancements in robotics, medical diagnostics, and geotechnical engineering, demonstrating their broad relevance in real-world problems. Moreover, hybrid bio-inspired optimization techniques integrating chaos theory and fuzzy logic have shown remarkable improvements in solving real-time scheduling and disaster management problems [33,34].

Recent advancements in metaheuristics have emphasized improving the balance between exploration and exploitation. The hunger games search algorithm [35], colony predation algorithm [36], and dolphin echolocation [37] have introduced novel mechanisms to enhance search diversity and prevent premature convergence. Additionally, data-driven and surrogate-assisted approaches, such as perturbation-based ensemble surrogates [38] and Kriging-assisted evolutionary algorithms [39], have further refined optimization methodologies. These advancements have led to increased efficiency in solving real-time scheduling problems, automated machine learning, and disaster resilience planning. With the increasing integration of artificial intelligence and reinforcement learning, metaheuristics have been applied to optimize autonomous control systems, industrial manufacturing, and predictive analytics [40,41].

The extensive application of metaheuristics in engineering optimization problems underscores their versatility and efficiency. From structural optimization [42] to cloud computing scheduling [43], these algorithms continue to revolutionize computational problem-solving. As research progresses, hybridization, adaptivity, and problem-specific enhancements are expected to advance the field of metaheuristic optimization further [44,45]. Integrating artificial intelligence, deep learning-assisted metaheuristics, and quantum-inspired algorithms holds promise for tackling increasingly complex optimization tasks in the future. With continued interdisciplinary collaboration, metaheuristic approaches are poised to contribute to next-generation computational intelligence and decision-making systems significantly. As the field evolves, metaheuristic frameworks incorporating quantum computing principles and

neuromorphic computing are expected to push the optimization boundaries, offering unprecedented computational capabilities for solving real-world challenges [46][47].

This study presents the Cloud Drift Optimization (CDO) algorithm, a novel nature-inspired metaheuristic that simulates the dynamic behavior of cloud particles under atmospheric forces. By integrating an adaptive weight adjustment mechanism, CDO effectively balances exploration and exploitation, enabling efficient search space traversal. The algorithm's performance is rigorously evaluated through a series of unimodal and multimodal benchmark functions, demonstrating its superiority over established optimization techniques, including Particle Swarm Optimization (PSO), Seagull Optimization Algorithm (SOA), Marine Predators Algorithm (MPA), Harris Hawks Optimization (HHO), Grasshopper Optimization Algorithm (GOA), Grey Wolf Optimizer (GWO), Fox Optimization Algorithm (FOX), Dragonfly Algorithm (DA), and Chimp Optimization Algorithm (ChOA). Furthermore, its applicability is validated in complex engineering design problems, including three-bar truss optimization, cantilever beam design, tension/compression spring optimization, and pressure vessel design. The results highlight CDO's exceptional convergence speed, robustness, and accuracy, establishing it as a competitive approach for solving high-dimensional and nonlinear optimization problems.

2. Cloud Drift Optimization (CDO)

Clouds are dynamic formations in the atmosphere, consisting of water droplets or ice crystals suspended in the air. They are crucial in the Earth's weather patterns and climate regulation. The life cycle and movement of clouds are complex processes influenced by various meteorological factors. Although clouds are often observed individually, they can also form vast systems that span large geographical areas, impacting weather conditions on a continental scale. The unique aspect of cloud movement is its adaptability to different atmospheric conditions. In their initial stages, clouds form slowly and move gradually under the influence of local weather patterns. As they mature and grow in size, clouds can exhibit rapid and unpredictable movements driven by strong winds and pressure gradients. This behavior allows clouds to travel long distances, influencing weather conditions across regions. One of the main characteristics of cloud movement is its ability to seek favorable atmospheric conditions for growth and development. Clouds naturally gravitate towards areas with higher moisture content and suitable temperature gradients, which provide the necessary conditions for their formation and persistence. This target-seeking behavior is essential for the survival and evolution of clouds in

the dynamic atmosphere. In the context of nature-inspired algorithms, the search process can be logically divided into two tendencies: exploration and exploitation. Exploration encourages search agents to move abruptly and explore new areas, while exploitation focuses on local refinement and optimization. Clouds naturally perform these two functions and target-seeking through their adaptive movement and environmental interactions. Therefore, if we can mathematically model this behavior, we can design a new nature-inspired algorithm based on the principles of cloud movement. While the next simulations and discussions highlight the CDO algorithm's effectiveness in locating the global optimum within a search space, the subsequent sections will evaluate the algorithm's performance using a variety of mathematical functions and three demanding real-world problems. Additionally, the source codes for the CDO algorithm are available at (https://www.mathworks.com/matlabcentral/fileexchange/180220-cloud-drift-optimization-cdo).

2.1. Problem Definition

Given an optimization problem where aim to minimize (or maximize) an objective function $f(x)$ over a bounded search space Ω, the objective is to find:

$$\min_{x\in\Omega} f(x) \tag{1}$$

where $\Omega = \left\{ x \in \mathbf{R}^d \mid lb \le x_i \le ub \right\}$, and lb and ub are the lower and upper bounds for the solution's i-th dimension, respectively.

2.2. Initialization

The algorithm begins with random initialization of N candidate solutions (clouds) within the search space, ensuring an unbiased exploration of the entire space:

$$X_i^0 = lb + (ub - lb).U(0,1), \qquad \forall i \in \{1, 2, ..., N\} \tag{2}$$

where $U(0,1)$ denotes a uniform random distribution over the range [0, 1]. This step mimics the random initial distribution of cloud particles in the sky, whose initial positions are unknown and spread out.

2.3. Weight Adaptation

Each cloud (or particle) is assigned a dynamic weight based on its fitness value relative to other particles in the swarm. This weight determines how strongly the particle is influenced by its own position and the positions of others. The weight for each cloud is updated based on

the fitness function:

$$w_{i,j} = \begin{cases} 1+(0.3+0.7.U(0,1)).\log_{10}\left(\dfrac{f^* - f(X_i)}{S}+1\right), & \text{if } i \le \dfrac{N}{2} \\ 1-(0.3+0.7.U(0,1)).\log_{10}\left(\dfrac{f(X_i) - f^*}{S}+1\right), & \text{otherwise} \end{cases} \quad (3)$$

$$S = f^* - f^{\max} + \varepsilon$$

Where f^* is the best fitness value found so far. $S = f^* - f^{\max} + \varepsilon$, where $f^{\max}$ is the worst fitness value among all clouds and ε is a small value to avoid division by zero. This weight adaptation allows the algorithm to focus more on particles with better fitness while allowing less fit particles to explore new areas.

2.4. Position Update

The position of each cloud is updated iteratively, and a combination of exploration and exploitation behaviors determines each update.

2.4.1 Exploitation Phase: Local Refinement

In this phase, clouds move toward the best solution found so far (i.e., the global best position). The position update equation is:

$$X_i^{(t+1)}(j) = X^*(j) + 0.8.v_b(j).(w_{i,j}.X_A(j) - X_B(j)) \quad (4)$$

where $X^*(j)$ is the best-known solution at iteration t. $v_b(j) \sim U(-0.2a, 0.2a)$ represents a small random adjustment influenced by a factor $a = \text{atanh}(-t/T+1)$, controlling the degree of exploitation. This refinement behavior mimics how clouds gradually refine their formations after initial movement, adjusting their positions based on the best nearby solutions.

2.4.2 Exploration Phase: Global Search

For the exploration phase, the algorithm ensures diversity in the search by allowing clouds to move randomly within the search space:

$$X_i^{(t+1)}(j) = v_c(j).X_i(j) \quad (5)$$

Where $v_c(j) \sim U(-0.2b, 0.2b)$ is another random factor influencing the particle's movement. $b = (1-t/T)$ is a parameter that decreases over time, transitioning the algorithm from

exploration to exploitation. This phase simulates the random drift of clouds in the atmosphere, ensuring the algorithm avoids being trapped in local optima by encouraging the exploration of new regions.

2.5. Random Perturbations

Some clouds may be randomly reinitialized to prevent premature convergence further and introduce exploration in later stages of the algorithm. This happens with a probability z, which decreases as the algorithm progresses:

$$X_i(j) = lb + (ub - lb)U.(0,1), \quad \text{with probability } z$$
$$z = 0.002 + 0.003.\left(1 - \frac{t}{T}\right) \tag{6}$$

This randomization mimics the unpredictable changes in atmospheric conditions that can redirect the course of drifting clouds, ensuring the search continues even in the final stages.

2.6. Convergence: Global Optimum

The algorithm converges when the fitness value of the best solution reaches a predefined threshold ε, or when the maximum number of iterations T is reached:

$$Stop \text{ if } f^* \le \varepsilon \text{ or } t = T \tag{7}$$

In the final stages, the search becomes more localized as the clouds converge toward the global optimum, similar to how clouds form stable, stationary patterns. The overall flow of the CDO algorithm can be summarized as:

- Initialization: Randomly initialize N clouds within the search space.
- Weight Adaptation: Update the weights based on the fitness values.
- Exploration & Exploitation: Update the positions based on the two phases.
- Random Perturbation: Introduce random reinitialization to avoid premature convergence.
- Convergence Check: Stop when the fitness reaches a threshold or the iteration limit is reached.

The Cloud Drift Optimization (CDO) algorithm mimics clouds' natural behavior, balancing exploration and exploitation to optimize complex functions. Mathematical modeling of cloud

movements, combined with adaptive behaviors (such as weight adjustment and random perturbations), allows the algorithm to avoid local optima and converge toward global solutions in a variety of optimization problems. A conceptual framework illustrating the interactions between cloud and the aim zone shown in Figure 1.

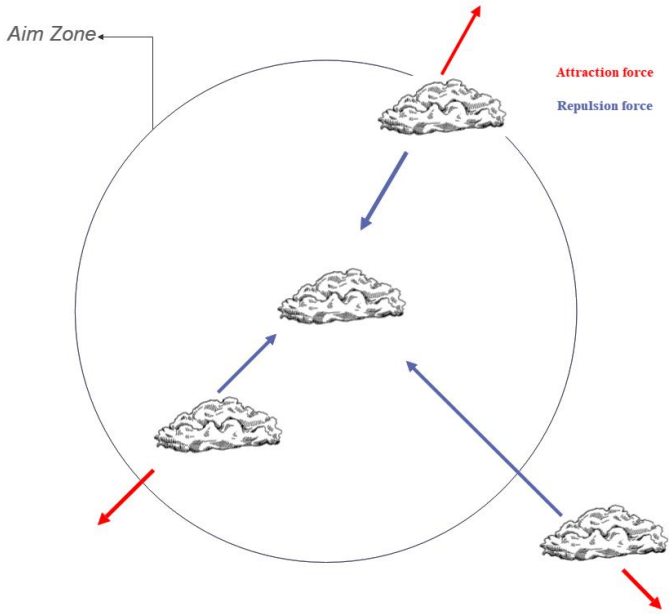


Figure 1. Basic corrective behaviors among individuals in a swarm of clouds

Figure 1 illustrates the fundamental interactions and corrective behaviors within a cloud swarm. It highlights three key forces: Attraction Force: Clouds are attracted to each other when they are at a certain distance, encouraging them to move closer together. Repulsion Force: When clouds get too close, a repulsion force pushes them apart, preventing overcrowding and ensuring effective exploration of the search space. Aim Zone: This represents the target area that the clouds aim to reach. The interplay between attraction and repulsion helps guide the clouds towards this optimal region. The diagram shows how these forces dynamically balance exploration (searching for new areas) and exploitation (focusing on promising regions), enabling the swarm to efficiently converge on the best solution. Figure 2 demonstrates the behavior of swarms in a 2D space using this equation. In this figure, 20 artificial clouds are tasked with moving over a duration of 10 time units.

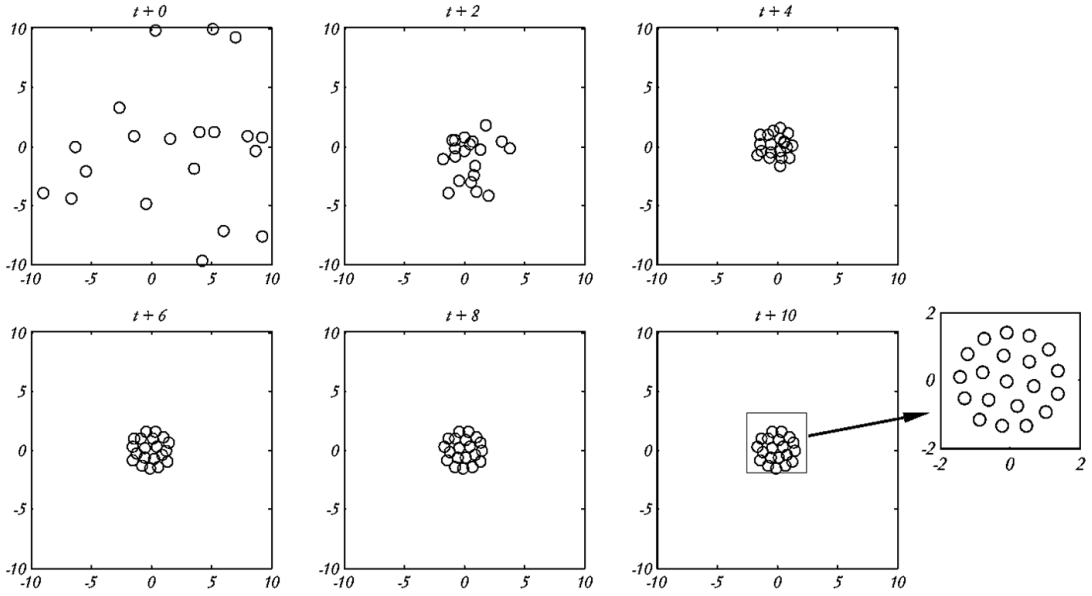


Figure 2. Behavior of a swarm in a two-dimensional space

The pseudo code of the CDO algorithm is shown in Figure 3 . This figure describes a process for optimizing solutions using a group of agents. It starts by defining a population of agents that will work together. Each agent's performance is evaluated using a specific assessment function, which calculates their fitness. The cloud are then analyzed to find the best and worst performers. Based on their success, the agents move and try to explore their environment better. Throughout the process, the best solution is continuously updated as new results are obtained. This entire operation occurs within a repetitive loop that continues until specific conditions are met. This method helps the clouds find the most optimal outcome.

```
1. Initialize the population X(i), for i = 1, 2, ..., N
   Initialize Best_position, Best_fitness = inf
   Initialize weight, AllFitness, and the maximum number of iterations
   Set algorithm parameters (z, StoppingThreshold, Max_iter, lb, ub)

2. Calculate the fitness of each search agent (X(i)):
   - For each particle X(i), calculate its fitness using the objective function fobj

3. T = Best search agent (Best_position)

4. While (iteration < Max_iter):
   a. Evaluate fitness for each particle:
      - Ensure each particle X(i) stays within boundaries (lb, ub)
      - Calculate the fitness value AllFitness(i)

   b. Sort the fitness values and calculate weights:
      - Sort AllFitness to find best and worst fitness values
      - Normalize fitness values using weight adjustments

   c. Update positions:
      - For each particle:
         - Normalize the movement probability based on fitness
         - Update position based on the cloud drift mechanism
         - Ensure the particle stays within the boundaries after update

   d. Update the best solution found (Best_position) and its fitness (Best_fitness):
      - If the current fitness is better than Best_fitness, update T (Best_position)

   e. Check for early stopping:
      - If Best_fitness < StoppingThreshold, exit the loop early

   f. Track the convergence:
      - Store the best fitness value for each iteration (Convergence_curve)

   g. Update iteration count

5. Return Best_position, Best_fitness, Convergence_curve
```

Figure 3. Pseudo codes of the CDO algorithm

While the simulations and discussions above illustrate the CDO algorithm's effectiveness in locating the global optimum within a search space, the following sections examine its performance further using a series of mathematical functions and four complex real-world problems.

3. Result

This section introduces the test problems and performance metrics to evaluate the proposed CDO algorithm. The experimental results are then presented and analyzed comprehensively.

3.1. Unimodal and multimodal result

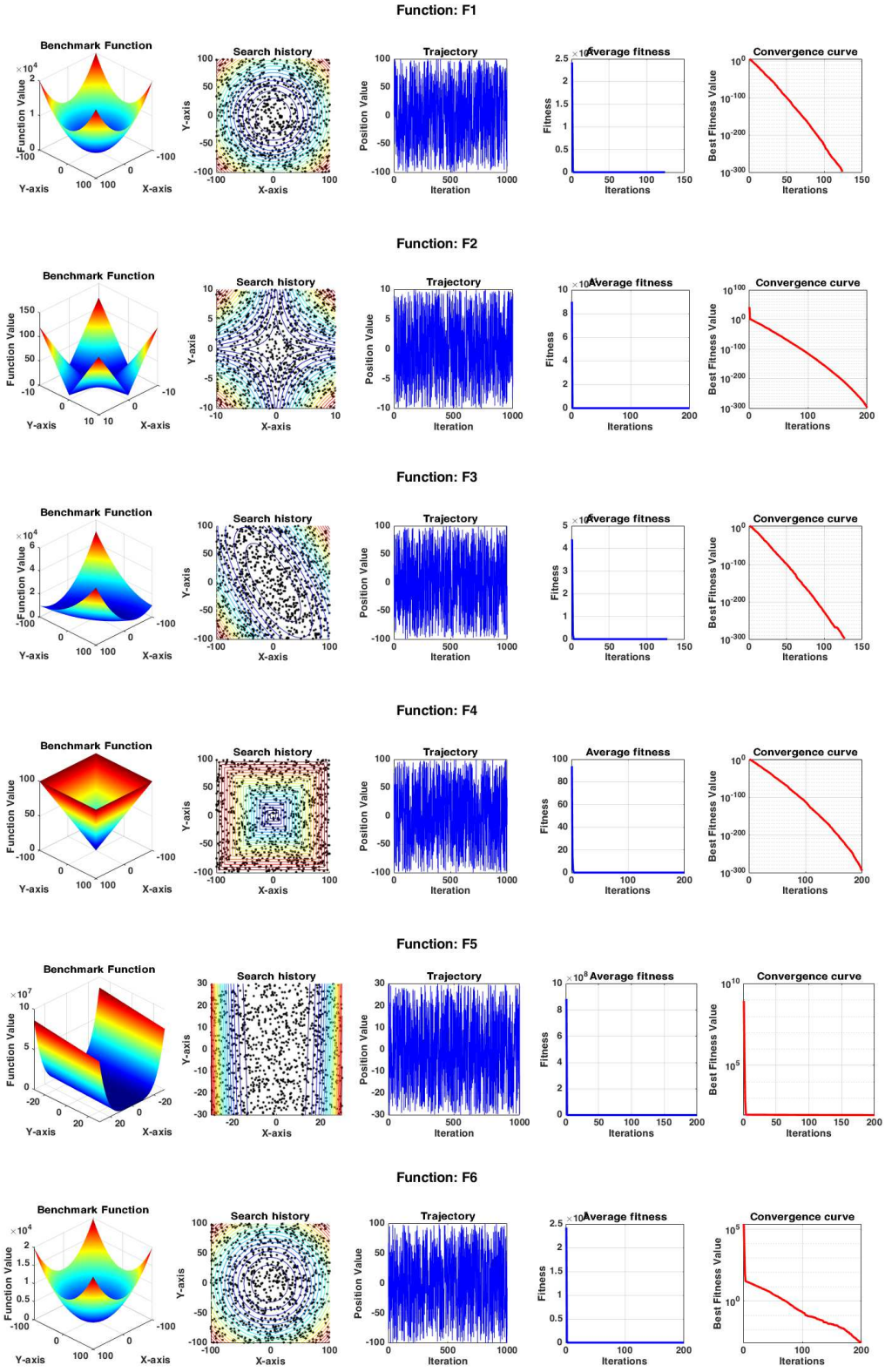
In the realm of stochastic optimization, it is standard practice to use a collection of

mathematical test functions with known optimal values to assess the performance of various algorithms quantitatively. However, these test functions must exhibit diverse characteristics to draw robust conclusions. This study uses two sets of test functions with distinct features to effectively benchmark the proposed algorithm's performance. These functions include both unimodal and multimodal types. The mathematical formulations for these test functions are provided in Table 1. To solve these functions, 1,000 search agents and 200 iterations were used, each being solved 30 times to obtain the best outcomes. Qualitative results, such as convergence curves, cloud trajectories, search history, and average population fitness, are illustrated and analyzed in the subsequent subsection. Nine algorithms from the literature were employed to validate the results, including both well-established and recent ones: ChOA, DA, FOX, GWO, GOA, HHO, MPA, SOA, and PSO.

Table 1. Unimodal and multimoda benchmark functions

Type	Function	Dim	Range	f_{min}
Unimodal	$f_1(x)=\sum_{i=1}^{n}x_i^2$	100	$[-100,100]$	0
	$f_2(x)=\sum_{i=1}^{n}\lvert x_i\rvert+\prod_{i=1}^{n}\lvert x_i\rvert$	100	$[-10,10]$	0
	$f_3(x)=\sum_{i=1}^{n}\left(\sum_{j-1}^{i}x_j\right)^2$	100	$[-100,100]$	0
	$f_4(x)=\max\{\lvert x_i\rvert,1\le i\le n\}$	100	$[-100,100]$	0
	$f_5(x)=\sum_{i=1}^{n-1}\left[100(x_{i+1}-x_i^2)^2+(x_i-1)^2\right]$	100	$[-30,30]$	0
	$f_6(x)=\sum_{i=1}^{n}\left(\lfloor x_i+0.5\rfloor\right)^2$	100	$[-100,100]$	0
	$f_7(x)=\sum_{i=1}^{n}(ix_i^4+rand[0,1])$	100	$[-1.28,1.28]$	0
	$f_8(x)=\sum_{i=1}^{n}\left[x_i^2-10\cos(2\pi x_i)+10\right]$	100	$[-5.12,5.12]$	0
Multimodal	$f_9(x)=-20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2})-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right)+20+e$	100	$[-32,32]$	0
	$f_{10}(x)=\frac{\pi}{n}\left\{10\sin(\pi y_1)+\sum_{i=1}^{n}(y_i-1)^2\left[1+10\sin^2(\pi y_{i+1})\right]+(y_n-1)^2\right\}$ $+\sum_{i=1}^{n}u(x_i,10,100,4)+\sum_{i=1}^{n}u(x_i,10,100,4)$ $y_i=1+\frac{x_i+1}{4}$ $u(x_i,a,k,m)=\begin{cases}k(x_i-a)^m & x_i<a\\ 0 & -a<x_i<a\\ k(-x_i-a)^m & x_i<-a\end{cases}$	100	$[-50,50]$	0
	$f_{11}(x)=0.1\left\{\sin^2(3\pi x_i)+\sum_{i=1}^{n}(x_i-1)^2\left[1+\sin^2(3\pi x_i+1)\right]+(x_n-1)^2\left[1+\sin^2(2\pi x_n)\right]\right\}$ $+\sum_{i=1}^{n}u(x_i,5,100,4)$	100	$[-50,50]$	0

Figure 4 presents the response of the CDO algorithm under the unimodal and multimodal functions listed in Table 1.



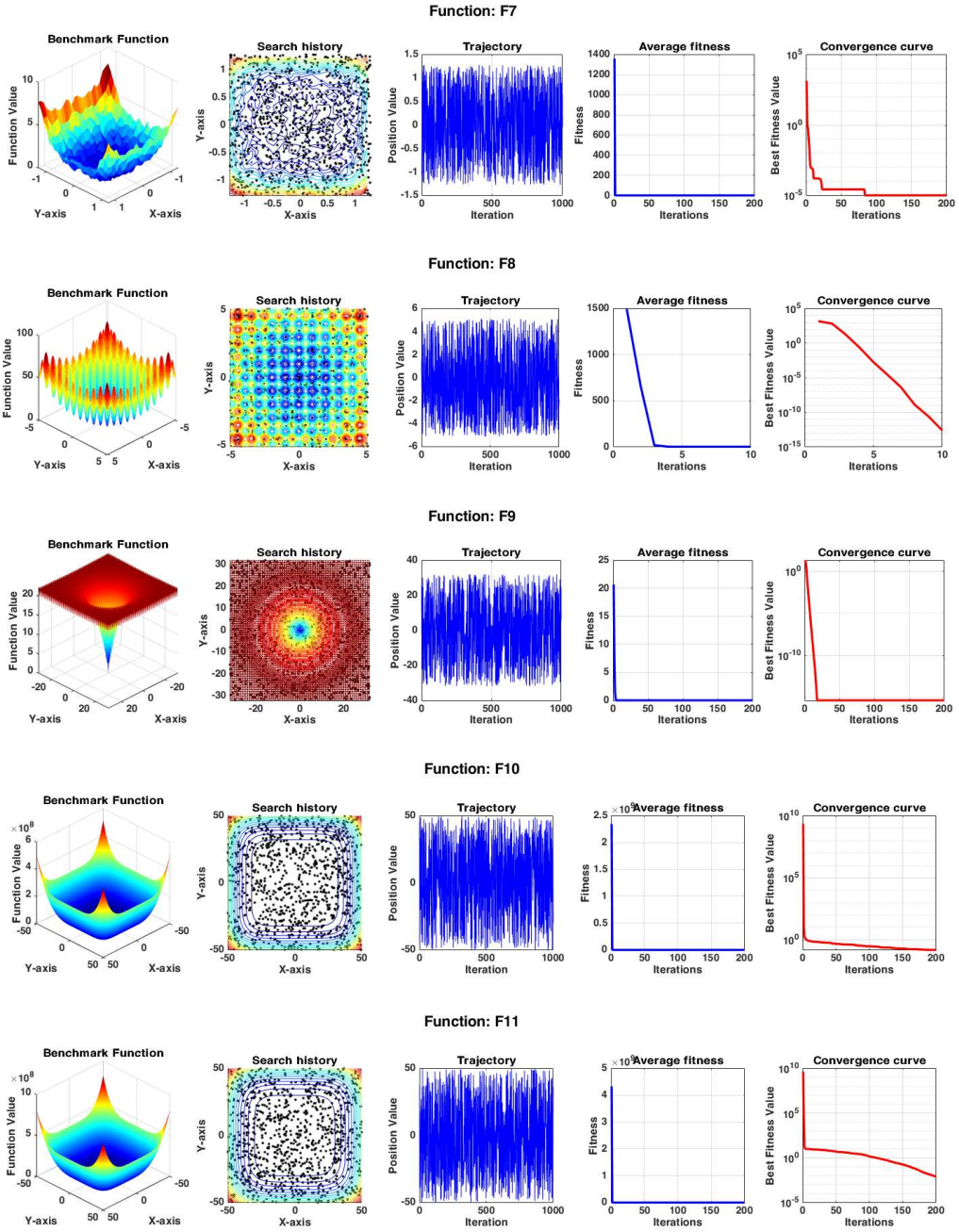


Figure 4. Behaviour of CDO on the benchmark problems

Figure 4 comprehensively analyses the CDO (Cloud Droplet Optimization) algorithm's performance across benchmark functions. A deeper dive into the search history, convergence curves, and overall dynamics of agent behavior provides a clearer understanding of the optimization process.

-Convergence Curves

The convergence curves displayed in the rightmost column are pivotal for evaluating the efficiency of the CDO algorithm. These curves typically depict the best fitness value obtained

during each iteration, allowing us to assess how quickly and effectively the algorithm converges to the global optimum.

Rapid Decline: A steep decline in the convergence curve indicates that the algorithm quickly identifies solutions that are approaching the optimum. This is significant for optimization as it suggests that the agents effectively explore the search space and refine their positions based on fitness feedback. For example, in functions with clear peaks, such as F1 or F2, the curves illustrate a rapid descent, reflecting the ease with which the algorithm navigates relatively uncomplicated landscapes.

Plateaus and Oscillations: In contrast, functions like F7 or F9 show more gradual convergence or sustained plateaus. This behavior may suggest that the algorithm encounters local optima or flat regions in the landscape where minimal improvements occur. It can indicate a challenge in navigating complex functions. Such patterns necessitate further exploration strategies, like increasing diversity among agents or adopting adaptive mechanisms to escape local optima.

Stability and Reaching Global Optimum: The convergence curves also help identify stability in achieving the global optimum. The algorithm demonstrates robustness if the best fitness value continues to improve without significant fluctuations. However, if the curve experiences erratic changes, this can signal issues with the optimization strategy, requiring adjustment in parameters or strategies applied in the algorithm.

-Search History

The search history visualizations reveal the paths followed by the agents during the optimization process. Analyzing this history provides insights into the algorithm's exploration and exploitation behaviors.

Exploratory Behavior: The search history plots typically show agent positions scattered throughout the search space in initial iterations. This wide distribution indicates a robust exploratory phase, where agents sample different regions to gather information about the fitness landscape. This behavior is crucial in avoiding premature convergence to suboptimal solutions.

Refinement Phase: As iterations progress, agents' positions often cluster around promising areas of the search space. This clustering represents the algorithm's exploitation phase, where agents capitalize on previously discovered high-fitness regions. Functions with

distinct, well-defined optima show a more pronounced transition from exploration to exploitation.

Diversity Maintenance: The nature of the search history can also reflect the algorithm's ability to maintain diversity among agents. If all agents converge too quickly to a single region, it may hinder the overall optimization process, risking convergence to a local rather than global optimum. A balanced approach that facilitates sufficient exploration while refining promising areas is critical for success.

-Trajectory of Position Values

The trajectory plots add another layer of understanding to the optimization process. They show how the individual agents' positions change over time concerning their fitness values.

Dynamics of Agent Movement: Consistent agent movement towards better positions often reflects a learning pattern, where positions are adjusted based on accumulated knowledge of the fitness landscape. For instance, if an agent's position fluctuates but generally trends upward in fitness, it suggests effective adaptations based on feedback.

Stagnation or Reverting Trends: If trajectories indicate stagnation or shift backward, this could highlight stagnation in the search process. Agents may be stuck in local optima or poorly exploring the landscape, necessitating adjustments to the algorithm's parameters or strategies to enhance exploration.

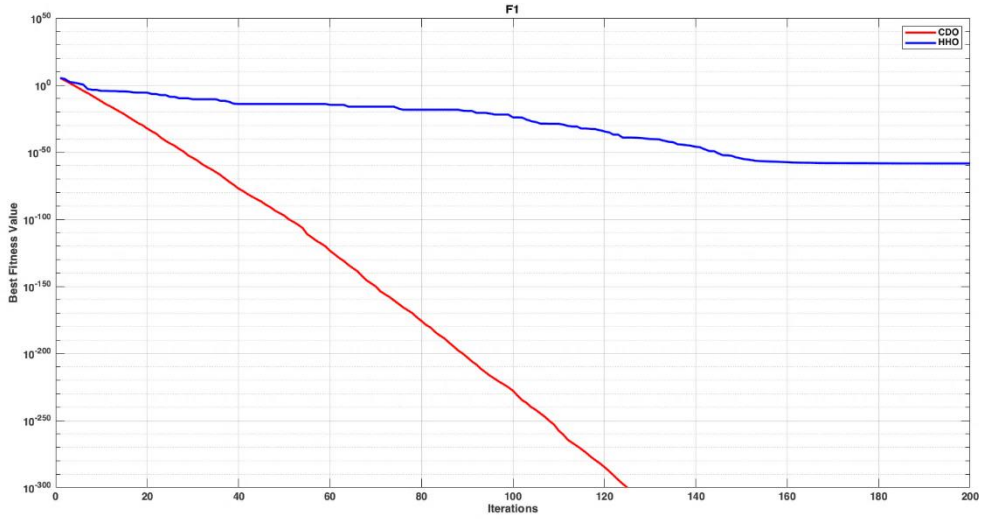
The collective analysis of convergence curves, search history, and trajectory graphs provides a layered understanding of CDO's effectiveness across benchmark functions. The behavior exhibited indicates how well the algorithm adapts to various search landscapes, balances exploration and exploitation, and ultimately converges to optimal solutions. In conclusion, thorough interpretation of these metrics not only illustrates the performance of the CDO algorithm but also informs potential improvements or adaptations in methodology for handling complex optimization problems in the future. Such insights are invaluable for researchers and practitioners aiming to enhance optimization strategies across various applications.

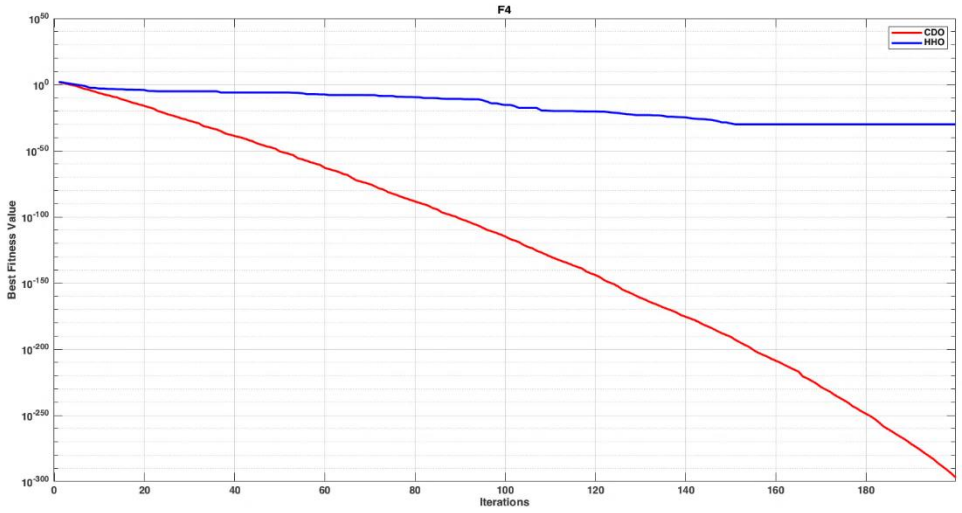
Table 3. Results of unimodal and multimodal benchmark functions

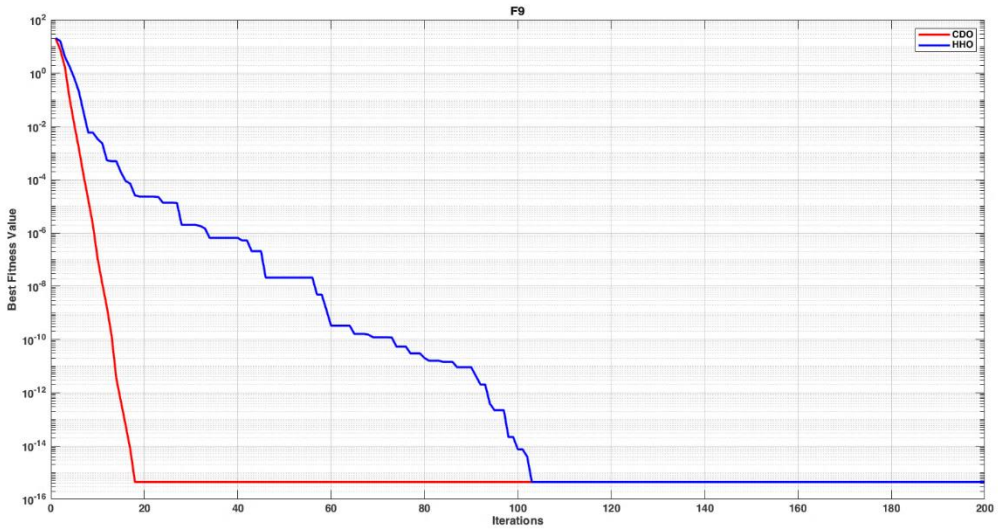
Function	Comparative Algorithms									
	CDO	ChOA	DA	FOX	GWO	GOA	HHO	MPA	SOA	PSO
F1	**3.3478e-303**	2.5942e-23	6832.9649	2.102e-54	1.079e-27	11773.2075	1.7064e-45	0.00010866 12378	14.2606	0.0001545
F2	**4.067e-299**	2.6045e-16	49.2886	3.7119e-52	8.5163e-17	88.1418	5.9421e-24	0.00273861 4188	0.59128	0.044772
F3	**1.1696e-303**	2.2304e-07	149588.454 2	170.20	5.2676e-06	32258.9053	8.5073e-36	1018.78106 1	10797.0246	68.5558
F4	**4.2177e-296**	2.9805e-06	58.6074	20.1045	7.6592e-07	22.1783	1.2456e-24	0.01951755 41	91.1506	1.1346
F5	94.737	132.922	28316948.8 289	86.5500	27.1946	2626954.37 65	**0.50579**	98.4651537 8	1469.6574	139.5597
F6	0.0013292	1.0086	3020.2748	0.11678	0.7408	12221.7733	**0.00090438**	8.70555924	35.1585	7.5646e-05
F7	**1.0139e-05**	0.000450	14.1567	0.19808	0.0014876	0.71128	0.0011455	0.00662295 1612	0.079796	0.18263
F8	**0**	**0**	963.6891	**0**	3.6314	526.3356	**0**	1.94242104 5e-05	163.0408	64.747
F9	**4.4409e-16**	2.524e-13	14.7042	1.2305	9.743e-14	11.7128	**4.4409e-16**	0.00050132 56751	19.9657	0.3812
F10	0.14847	0.05607	10777380.3 006	9.6587	0.05592	53.0676	**4.1182e-07**	4.94776397 2e-05	1.0647	2.2667e-06
F11	0.0079427	1.04980	0.54258	0.1387	0.56801	0.024964	0.00019915	**1.42211681 8e-09**	2.4863	0.013164

Table 3 illustrates the comparative performance of the studied algorithms across unimodal

(functions 1 to 8) and multimodal (functions 9 to 11) functions. The results demonstrate that the CDO algorithm outperforms all others in functions 1 to 4. In functions 5 and 6, however, the HHO algorithm claims the top performance, with CDO following closely behind in function 5. The CDO algorithm maintains its superiority in functions 7 and 8, while the CHOA algorithm ranks next in function 7. In function 10, HHO leads in performance, with the MPA and PSO algorithms trailing. Finally, in function 11, MPA takes the lead, followed by HHO and CDO. Overall, CDO consistently ranks high across seven objective functions, showcasing its reliable performance. While HHO also demonstrates strong results in certain functions, Figure 5 provides a comparative analysis of the convergence results for the algorithms under functions 1, 4, 9, and 10.




F4
CDO
HHO
Best Fitness Value
Iterations


F9
CDO
HHO
Best Fitness Value
Iterations

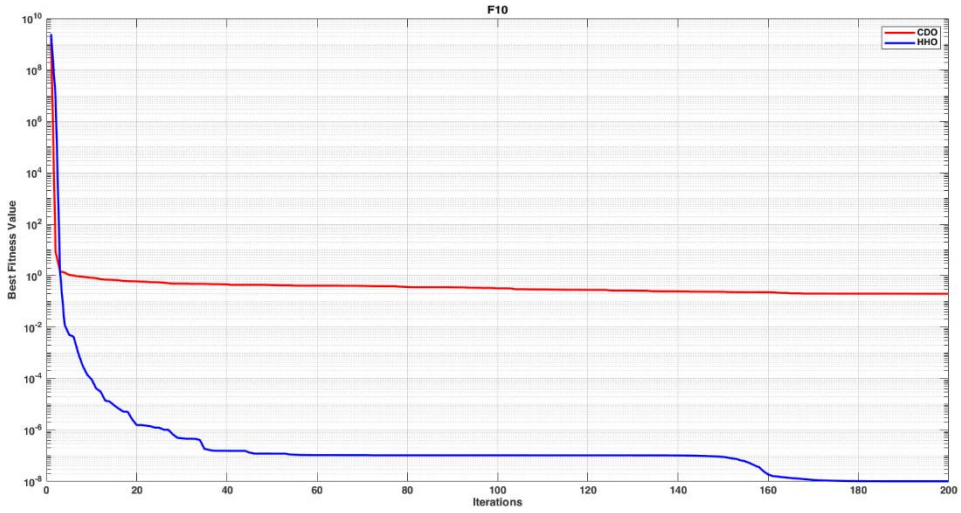


Figure 5. Convergence of CDO and HHO on the test functions (F1, F4, F9, and F10)

Figure 5 compares the performance of two optimization algorithms, Cloud Drift Optimization (CDO) and Harris Hawks Optimization (HHO), across four benchmark functions: F1, F4, F9, and F10. The y-axis indicates the best fitness value achieved by each algorithm, where lower values signify better performance, while the x-axis represents the number of iterations. Overall, CDO consistently demonstrates faster convergence and greater accuracy in all functions. It reduces the fitness value to near zero, showcasing its ability to find optimal solutions quickly. In contrast, HHO often struggles to improve its results, especially in functions F1 and F4, where CDO distinctly outperforms it. The Cloud Drift Optimization (CDO) algorithm outperforms Harris Hawks Optimization (HHO) in terms of convergence speed, optimization accuracy, and overall robustness across different benchmark functions.

3.2. Three-bar truss design

This structural design problem is one of the most widely used case studies [48,49]. This problem is formulated as follows:

Consider $$\vec{x} = [x_1, x_2] = [A_1, A_2] \quad (8)$$

Minimise $$f(\vec{x}) = (2\sqrt{2}x_1 + x_2) \times 1 \quad (9)$$

$$g_1(\vec{x}) = \frac{\sqrt{2}x_1 + x_2}{\sqrt{2}x_1^2 + 2x_1x_2} P - \sigma \le 0$$

Subject to $$g_2(\vec{x}) = \frac{x_2}{\sqrt{2}x_1^2 + 2x_1x_2} P - \sigma \le 0 \quad (10)$$

$$g_3(\vec{x}) = \frac{1}{\sqrt{2}x_2 + x_1} P - \sigma \le 0$$

Variable range $$0 \le x_1, x_2 \le 1 \quad (11)$$

where $1 = 100cm,\ P = 2KN/cm^2,\ \sigma = 2KN/cm^2$

Figure 6 illustrates the configuration of the truss and the forces applied to it. As indicated by the figure and the problem description, there are two structural parameters to consider: bars 1 and 3 and the area of bar 2. The goal here is to minimize the truss's weight. This optimization problem is subject to various constraints, including stress, deflection, and buckling. The CDO method was implemented with 20 search agents over 650 iterations to tackle this challenge. Given the constrained nature of the problem, a suitable constraint handling technique was necessary to incorporate with CDO; thus, a death penalty approach was employed. This method imposes significant penalties on search agents that breach constraints, resulting in a higher objective value. For validation purposes, the outcomes were compared with those from Ant Lion Optimizer (ALO), DEDS, PSO-DE, MBA, CS, Ray and Sain, and Tsa methods, with the results in Table 4. This table highlights the optimal values for the variables and the truss weight. A review of the algorithm results demonstrates that CDO delivered notably competitive outcomes compared to ALO, DEDS, PSO-DE, and MBA, showing superior maximum function evaluation. Additionally, CDO significantly outperformed the other methods. These findings indicate that the CDO algorithm effectively manages the complexities of a constrained search space.

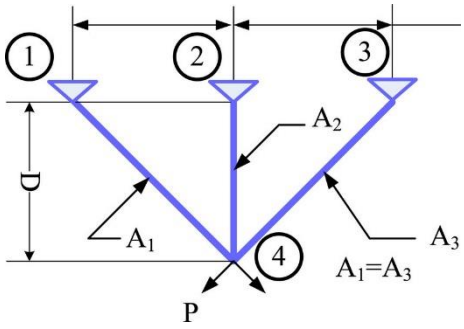


Figure 6. Three-bar truss design problem

Table 4. Outcomes of the comparison for the three-bar truss design issue

Algorithm	Optimal values for variables		Optimal weight
	x_1	x_2	
CDO	0.78711	0.40795	**263.615777**
ALO	0.788662816	0.408283134	263.8958434
DEDS	0.78867513	0.40824828	263.8958434
PSO-DE	0.7886751	0.4082482	263.8958433
MBA	0.788565	0.4085597	263.8958522
Ray and Sain	0.795	0.395	264.3
Tsa	0.788	0.408	263.68
CS	0.78867	0.40902	263.9716

3.3. Cantilever beam design

This is another well-known structural design problem in the literature, which can be formulated as follows:

Consider $$\vec{x} = [x_1, x_2, x_3, x_4, x_5] \quad (12)$$

Minimise $$f(\vec{x}) = 0.6224(x_1 + x_2 + x_3 + x_4 + x_5) \quad (13)$$

Subject to $$g(\vec{x}) = \frac{61}{x_1^3} + \frac{27}{x_2^3} + \frac{19}{x_3^3} + \frac{7}{x_4^3} + \frac{1}{x_5^3} - 1 \le 0 \quad (14)$$

Variable range $$0.01 \le x_1, x_2, x_3, x_4, x_5 \le 100 \quad (15)$$

Figure 7 illustrates that the cantilever beam is constructed using five hollow box girders with a square cross-section, and the lengths of these girders serve as the design parameters for this analysis. There is one constraint associated with the problem. The CDO algorithm is utilized to identify the optimal solution, featuring 20 search agents and a maximum of 650 iterations. The findings are presented and compared with results from ALO, MMA, GCA-I, GCA-II, CS, and SOS for validation in Table 5. The data in Table 5 indicates that CDO successfully determines the lightest optimal weight.

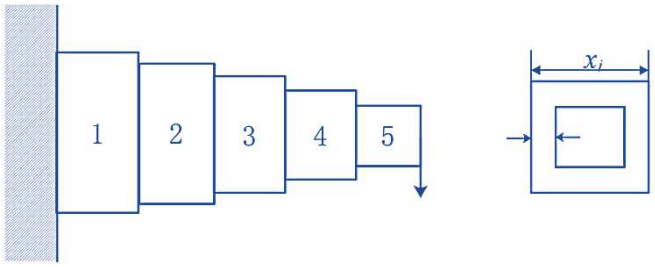


Figure 7. Cantilever beam design problem

Table 5.Outcomes of the comparison for the Cantilever beam design issue

Algorithm	Optimal values for variables					Optimal weight
	x_1	x_2	x_3	x_4	x_5	
CDO	6.00157	5.31241	4.47307	3.48274	2.14841	**1.33116**
ALO	6.01812	5.31142	4.48836	3.49751	2.158329	1.33995
MMA	6.0100	5.3000	4.4900	3.4900	2.1500	1.3400
GCA_I	6.0100	5.3000	4.4900	3.4900	2.1500	1.3400
GCA_II	6.0100	5.3000	4.4900	3.4900	2.1500	1.3400
CS	6.0089	5.3049	4.5023	3.5077	2.1504	1.3339
SOS	6.01878	5.30344	4.49587	3.4989	2.1555	1.3399

3.4. Tension/compression spring design

This is another well-known structural design problem in the literature, which can be formulated as follows:

Consider $$\vec{x} = [x_1, x_2, x_3] \quad (16)$$

Minimise $$f(\vec{x}) = (x_3 + 2)x_2 x_1^2 \quad (17)$$

Subject to
$$g_1(\vec{x}) = \frac{1 - x_2^3 x_3}{71785 x_1^4} \le 0$$
$$g_2(\vec{x}) = 1 - \left(\frac{4x_2^2 - x_1 x_2}{12566(x_2 x_1^3 - x_1^4)} \right) + \frac{1}{5108 x_1^2} \le 0 \quad (18)$$
$$g_3(\vec{x}) = \frac{1 - 140.45 x_1}{x_2^3 x_3} \le 0$$
$$g_4(\vec{x}) = \frac{x_1 + x_2}{1.5 - 1} \le 0$$

Variable range $$0.05 \le x_1 \le 2, 0.25 \le x_2 \le 1.3, 2 \le x_3 \le 15 \quad (19)$$

Figure 8 shows that the spring consists of a member characterized by length, width, and thickness. The problem is subject to four constraints. The CDO algorithm is employed to find the best solution, utilizing 20 search agents and allowing for a maximum of 650 iterations. The results are presented and compared with those obtained from EGO, GWO, MFO, WOA, MVO, and HS for validation, as detailed in Table 6. The information in Table 6 reveals that CDO effectively identifies the lightest optimal weight.

Table 6. Outcomes of the comparison for the Tension/compression spring design issue

Algorithm	Optimal values for variables			Optimal weight
	d	*D*	*N*	
CDO	0.050001	0.304784	13.28541	**0.0126048**
EGO	0.05	0.315786	14.29092	0.0126611
GWO	0.05169	0.356737	11.28885	0.0126660
MFO	0.0519944	0.3641093	10.29092	0.0126669
WOA	0.051207	0.345215	12.004032	0.0126763
MVO	0.0500	0.315956	14.22623	0.0128169

HS	0.051154	0.349871	12.076432	0.0126706

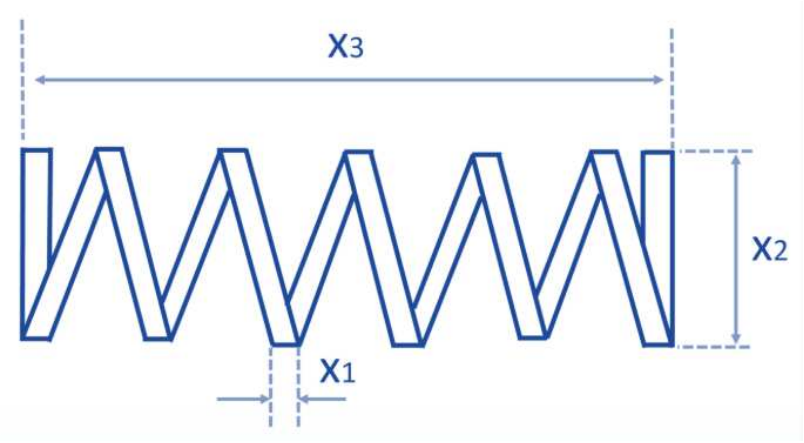


Figure 8. Tension/compression spring design problem

3.5. Pressure vessel design

The design of pressure vessels is a well-known engineering challenge in meta-heuristics. This project aims to minimize the total cost of a cylinder-shaped pressure vessel while adhering to specific pressure requirements outlined in Figure 9. Four structural parameters need to be optimized: the head thickness (T_h), the shell thickness (T_s), the length of the cylindrical section without the head (L), and the inner radius (R). The optimization process is subject to four constraints.

Consider
$$\vec{x} = [x_1, x_2, x_3, x_4] = [T_s, T_h, R, L] \tag{20}$$

Minimise
$$f(\vec{x}) = 0.6224 x_1 x_2 x_3 + 1.7781 x_2 x_3^2 + 3.1661 x_1^2 x_4 + 19.84 x_1^2 x_3 \tag{21}$$

Subject to
$$g_1(\vec{x}) = -x_1 + 0.0193 x_3 \le 0$$
$$g_2(\vec{x}) = -x_3 + 0.00954 x_3 \le 0$$
$$g_3(\vec{x}) = -\pi x_3^2 x_4 - \left(4/3\right)\pi x_3^3 + 1296000 \le 0 \tag{22}$$
$$g_4(\vec{x}) = x_4 - 240 \le 0$$

Variable range
$$0 \le x_1 \le 99, 0 \le x_2 \le 99, 10 \le x_3 \le 200, 10 \le x_4 \le 200 \tag{23}$$

Figure 9 shows the Pressure vessel. The problem is subject to four constraints. Numerous researchers have applied different meta-heuristic strategies to determine the optimal design for this test case, including techniques such as GWO, GA, MFO, WOA, MVO, and HS. Table 7 presents the outcomes of this problem, comparing CDO with other methods previously mentioned. The data indicates that CDO produces the most effective design at the lowest cost, demonstrating that our proposed algorithm outperforms all other algorithms.

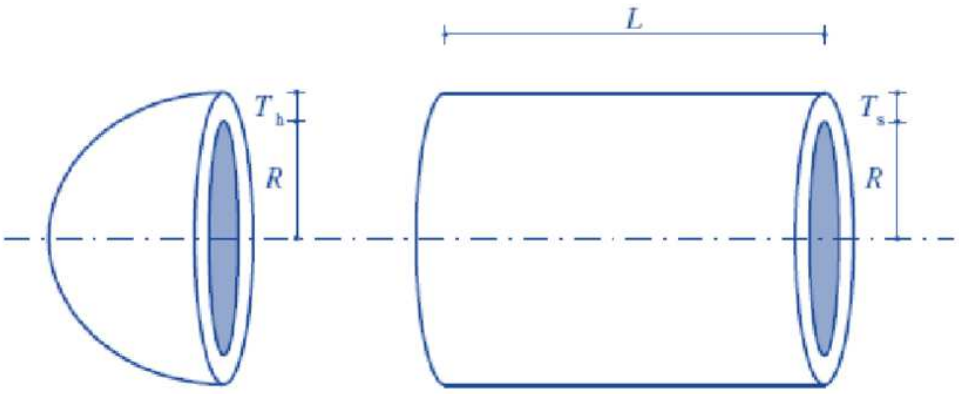


Figure 9. Pressure vessel design problem

Table 7. Outcomes of the comparison for the Pressure vessel design issue

Algorithm	Optimal values for variables				Optimal weight
	Ts	*Th*	*R*	*L*	
CDO	0.819965	0.40361	42.48527	171.9015	**5755.2296**
GWO	0.812500	0.434500	42.103624	176.75873	6051.5639
GA	0.812500	0.434500	40.3239	200.000	6288.7445
MFO	12.57917	6.748778	42.098445	176.63659	6059.7413
WOA	0.812500	0.437500	42.098269	176.63899	6059.7410
MVO	0.812500	0.437500	42.090738	176.73869	6060.8066
HS	1.099523	0.906579	44.456397	179.65887	6550.0230

4. Conclusion

This study presents the Cloud Drift Optimization (CDO) algorithm, an innovative nature-inspired metaheuristic to solve complex optimization challenges. Drawing inspiration from the fluid dynamics of clouds influenced by atmospheric conditions, CDO skillfully balances exploration and exploitation through a mechanism that adaptively adjusts weights. The algorithm's effectiveness was meticulously assessed using benchmark functions and practical engineering design scenarios, showcasing its robustness, efficiency, and exceptional convergence characteristics. Unimodal and multimodal benchmark tests indicate that CDO consistently outshines leading metaheuristic algorithms across diverse optimization landscapes. In unimodal functions, which evaluate an algorithm's ability to exploit, CDO achieved the lowest function values with quicker convergence rates than rival methods. In multimodal functions, where striking a balance between exploration and exploitation is essential, CDO effectively navigated local optima to identify global solutions. Comparative assessments with algorithms like PSO, HHO, GWO, and MPA validated CDO's enhanced accuracy and optimization effectiveness. Beyond benchmark tests, CDO's application to real-world structural optimization challenges further affirmed its practical efficacy. In the three-bar truss design

scenario, CDO provided an optimal structural configuration with reduced weight compared to other methods. The cantilever beam design realized the lightest structure while adhering to design constraints, surpassing approaches such as ALO, GCA, and SOS. The optimization of the tension/compression spring design illustrated CDO's capability to maneuver through constrained search spaces to achieve minimal weight with precise parameter choices. Finally, in the pressure vessel design issue, CDO yielded the most cost-effective solution, demonstrating its efficiency in optimizing large engineering structures. CDO's performance across various optimization challenges underscores its potential for wider applications in engineering, artificial intelligence, and computational sciences. Future research may consider hybridizing CDO with machine learning techniques to bolster adaptability in dynamic optimization situations. Moreover, incorporating reinforcement learning strategies could enhance efficiency in real-time decision-making contexts. With its demonstrated excellence, CDO signifies a major step forward in metaheuristic optimization and is set to play a pivotal role in addressing increasingly intricate engineering and computational challenges.

5. Declaration

Conflict of Interest: The authors declare that there is no conflict of interest regarding the publication of this paper.

Funding: This research did not receive a specific grant from public, commercial, or not-for-profit funding agencies.

Data Availability: The datasets generated and analyzed during the current study are available from the corresponding author upon reasonable request.

Authors' Contributions: Mohammad Alibabaei Shahraki: Writing – Original Draft, Writing – Review & Editing.

6. Reference

[1] C.A. Coello Coello, Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art, Comput Methods Appl Mech Eng 191 (2002) 1245–1287. https://doi.org/10.1016/S0045-7825(01)00323-1.

[2] R.T. Marler, J.S. Arora, Survey of multi-objective optimization methods for engineering, Structural and Multidisciplinary Optimization 26 (2004) 369–395. https://doi.org/10.1007/S00158-003-0368-6/METRICS.

[3] X.-She. Yang, Nature-inspired metaheuristic algorithms, (2010) 148. https://books.google.com/books/about/Nature_inspired_Metaheuristic_Algorithms.html?id=iVB_ETlh4ogC (accessed February 23, 2025).

[4] I. Boussaïd, J. Lepagnot, P. Siarry, A survey on optimization metaheuristics, Information Sciences—Informatics and Computer Science, Intelligent Systems, Applications: An International Journal 237 (2013) 82–117. https://doi.org/10.1016/J.INS.2013.02.041.

[5] J.H. Holland, Genetic algorithms, Sci Am 267 (1992) 66–72. https://doi.org/10.1038/SCIENTIFICAMERICAN0792-66.

[6] L.J. Fogel, A.J. Owens, M.J. Walsh, Artificial intelligence through a simulation of evolution, Evolutionary Computation: The Fossil Record (1998) 230–248. https://doi.org/10.1109/9780470544600.CH7.

[7] R. Storn, K. Price, Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces, Journal of Global Optimization 11 (1997) 341–359. https://doi.org/10.1023/A:1008202821328/METRICS.

[8] A. Zhou, B.Y. Qu, H. Li, S.Z. Zhao, P.N. Suganthan, Q. Zhangd, Multiobjective evolutionary algorithms: A survey of the state of the art, Swarm Evol Comput 1 (2011) 32–49. https://doi.org/10.1016/J.SWEVO.2011.03.001.

[9] N. Khanduja, B. Bhushan, Recent Advances and Application of Metaheuristic Algorithms: A Survey (2014–2020), Studies in Computational Intelligence 916 (2021) 207–228. https://doi.org/10.1007/978-981-15-7571-6_10.

[10] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Transactions on Evolutionary Computation 6 (2002) 182–197. https://doi.org/10.1109/4235.996017.

[11] E.-G. Talbi, Metaheuristics: From Desing to Implementation. John Wiley & Sons, Inc., (2009) 624. https://www.wiley.com/en-au/Metaheuristics%3A+From+Design+to+Implementation+-p-9780470278581 (accessed February 23, 2025).

[12] R. Eberhart, J. Kennedy, New optimizer using particle swarm theory, Proceedings of the International Symposium on Micro Machine and Human Science (1995) 39–43. https://doi.org/10.1109/MHS.1995.494215.

[13] A. Colorni, M. Dorigo, V. Maniezzo, F. Varela, P. Bourgine, Distributed Optimization by Ant Colonies, (1992).

[14] L. Abualigah, A. Diabat, S. Mirjalili, M. Abd Elaziz, A.H. Gandomi, The Arithmetic Optimization Algorithm, Comput Methods Appl Mech Eng 376 (2021) 113609. https://doi.org/10.1016/J.CMA.2020.113609.

[15] S. Mirjalili, S.M. Mirjalili, A. Lewis, Grey Wolf Optimizer, Advances in Engineering Software 69 (2014) 46–61. https://doi.org/10.1016/J.ADVENGSOFT.2013.12.007.

[16] S. Mirjalili, A. Lewis, The Whale Optimization Algorithm, Advances in Engineering Software 95 (2016) 51–67. https://doi.org/10.1016/J.ADVENGSOFT.2016.01.008.

[17] S. Mirjalili, A.H. Gandomi, S.Z. Mirjalili, S. Saremi, H. Faris, S.M. Mirjalili, Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems, Advances in Engineering Software 114 (2017) 163–191. https://doi.org/10.1016/J.ADVENGSOFT.2017.07.002.

[18] B. Naheliya, P. Redhu, K. Kumar, A Hybrid Deep Learning Method for Short-Term Traffic Flow Forecasting: GSA-LSTM, Indian J Sci Technol 16 (2023) 4358–4368. https://doi.org/10.17485/IJST/V16I46.2520.

[19] G.N.. Kouziokas, Swarm intelligence and evolutionary computation : theory, advances and applications in machine learning and deep learning, (2022).

[20] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, Optimization by Simulated Annealing, Science (1979) 220 (1983) 671–680. https://doi.org/10.1126/SCIENCE.220.4598.671.

[21] F. Glover, Tabu Search: A Tutorial, Https://Doi.Org/10.1287/Inte.20.4.74 20 (1990) 74–94. https://doi.org/10.1287/INTE.20.4.74.

[22] H. Eskandar, A. Sadollah, A. Bahreininejad, M. Hamdi, Water cycle algorithm – A novel metaheuristic optimization method for solving constrained engineering optimization problems, Comput Struct 110–111 (2012) 151–166. https://doi.org/10.1016/J.COMPSTRUC.2012.07.010.

[23] A. Kaveh, V.R. Mahdavi, Colliding bodies optimization: A novel meta-heuristic method, Comput Struct 139 (2014) 18–27. https://doi.org/10.1016/J.COMPSTRUC.2014.04.005.

[24] E. Rashedi, H. Nezamabadi-pour, S. Saryazdi, GSA: A Gravitational Search Algorithm, Inf Sci (N Y) 179 (2009) 2232–2248. https://doi.org/10.1016/J.INS.2009.03.004.

[25] R.N. Asl, R. Assistant, M. Aslani, R.A. Masoud, S. Panahi, Sizing Optimization of Truss Structures using a Hybridized Genetic Algorithm, (2013). https://arxiv.org/abs/1306.1454v1 (accessed February 23, 2025).

[26] F.J. Rodriguez, C. García-Martinez, M. Lozano, Hybrid metaheuristics based on evolutionary algorithms and simulated annealing: Taxonomy, comparison, and synergy test, IEEE Transactions on Evolutionary Computation 16 (2012) 787–800. https://doi.org/10.1109/TEVC.2012.2182773.

[27] X.S. Yang, S. Deb, Cuckoo search via Lévy flights, 2009 World Congress on Nature and Biologically Inspired Computing, NABIC 2009 - Proceedings (2009) 210–214. https://doi.org/10.1109/NABIC.2009.5393690.

[28] X.S. Yang, A.H. Gandomi, Bat Algorithm: A Novel Approach for Global Engineering Optimization, Engineering Computations (Swansea, Wales) 29 (2012) 464–483. https://doi.org/10.1108/02644401211235834.

[29] X.S. Yang, Firefly Algorithms for Multimodal Optimization, Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes

in Bioinformatics) 5792 LNCS (2009) 169–178. https://doi.org/10.1007/978-3-642-04944-6_14.

[30] A. Faramarzi, M. Heidarinejad, B. Stephens, S. Mirjalili, Equilibrium optimizer: A novel optimization algorithm, Knowl Based Syst 191 (2020) 105190. https://doi.org/10.1016/J.KNOSYS.2019.105190.

[31] M.Y. Cheng, D. Prayogo, Symbiotic Organisms Search: A new metaheuristic optimization algorithm, Comput Struct 139 (2014) 98–112. https://doi.org/10.1016/J.COMPSTRUC.2014.03.007.

[32] A. Mohammadzadeh, D. Javaheri, J. Artin, Chaotic hybrid multi-objective optimization algorithm for scientific workflow scheduling in multisite clouds, Journal of the Operational Research Society 75 (2024) 314–335. https://doi.org/10.1080/01605682.2023.2195426.

[33] M. Babaei, M. Mollayi, M. Babaei, Multi-objective optimization of reinforced concrete frames using NSGA-II algorithm, Engineering Structures and Technologies 8 (2016) 157–164. https://doi.org/10.3846/2029882X.2016.1250230.

[34] M. Ghasemi, S. kadkhoda Mohammadi, M. Zare, S. Mirjalili, M. Gil, R. Hemmati, A new firefly algorithm with improved global exploration and convergence with application to engineering optimization, Decision Analytics Journal 5 (2022) 100125. https://doi.org/10.1016/J.DAJOUR.2022.100125.

[35] Y. Yang, H. Chen, A.A. Heidari, A.H. Gandomi, Hunger games search: Visions, conception, implementation, deep analysis, perspectives, and towards performance shifts, Expert Syst Appl 177 (2021) 114864. https://doi.org/10.1016/J.ESWA.2021.114864.

[36] J. Tu, H. Chen, M. Wang, A.H. Gandomi, The Colony Predation Algorithm, J Bionic Eng 18 (2021) 674–710. https://doi.org/10.1007/S42235-021-0050-Y/METRICS.

[37] A. Kaveh, N. Farhoudi, A new optimization method: Dolphin echolocation, Advances in Engineering Software 59 (2013) 53–70. https://doi.org/10.1016/J.ADVENGSOFT.2013.03.004.

[38] J.Y. Li, Z.H. Zhan, H. Wang, J. Zhang, Data-Driven Evolutionary Algorithm with Perturbation-Based Ensemble Surrogates, IEEE Trans Cybern 51 (2021) 3925–3937. https://doi.org/10.1109/TCYB.2020.3008280.

[39] Z. Song, H. Wang, C. He, Y. Jin, A Kriging-Assisted Two-Archive Evolutionary Algorithm for Expensive Many-Objective Optimization, IEEE Transactions on Evolutionary Computation 25 (2021) 1013–1027. https://doi.org/10.1109/TEVC.2021.3073648.

[40] S. Mirjalili, J.S. Dong, Multi-Objective Optimization using Artificial Intelligence Techniques, (2020). https://doi.org/10.1007/978-3-030-24835-2.

[41] A. Viswanathan, Reinforcement Learning Based Metaheuristic Algorithm For Optimized Load Balancing In Cloud Environment, Journal of Electrical Systems 20 (2024) 1827–1840. https://doi.org/10.52783/JES.2520.

[42] A.H. Gandomi, X.S. Yang, S. Talatahari, A.H. Alavi, Metaheuristic Algorithms in Modeling and Optimization, Metaheuristic Applications in Structures and Infrastructures (2013) 1–24. https://doi.org/10.1016/B978-0-12-398364-0.00001-2.

[43] A. Mohammadzadeh, M. Masdari, Scientific workflow scheduling in multi-cloud computing using a hybrid multi-objective optimization algorithm, J Ambient Intell Humaniz Comput 14 (2023) 3509–3529. https://doi.org/10.1007/S12652-021-03482-5/METRICS.

[44] E.H. Houssein, M.K. Saeed, G. Hu, M.M. Al-Sayed, Metaheuristics for Solving Global and Engineering Optimization Problems: Review, Applications, Open Issues and Challenges, Archives of Computational Methods in Engineering 2024 31:8 31 (2024) 4485–4519. https://doi.org/10.1007/S11831-024-10168-6.

[45] J.A. Parejo, A. Ruiz-Cortés, S. Lozano, P. Fernandez, Metaheuristic optimization frameworks: A survey and benchmarking, Soft Comput 16 (2012) 527–561. https://doi.org/10.1007/S00500-011-0754-8/TABLES/11.

[46] K.H. Han, J.H. Kim, Genetic quantum algorithm and its application to combinatorial optimization problem, Proceedings of the IEEE Conference on Evolutionary Computation, ICEC 2 (2000) 1354–1360. https://doi.org/10.1109/CEC.2000.870809.

[47] J.D. Hidary, Quantum Computing: An Applied Approach, Quantum Computing: An Applied Approach (2019) 1–379. https://doi.org/10.1007/978-3-030-23922-0/COVER.

[48] A.H. Gandomi, X.S. Yang, A.H. Alavi, Cuckoo search algorithm: A metaheuristic approach to solve structural optimization problems, Eng Comput 29 (2013) 17–35. https://doi.org/10.1007/S00366-011-0241-Y/METRICS.

[49] A. Sadollah, A. Bahreininejad, H. Eskandar, M. Hamdi, Mine blast algorithm: A new population based algorithm for solving constrained engineering optimization problems, Appl Soft Comput 13 (2013) 2592–2612. https://doi.org/10.1016/J.ASOC.2012.11.026.