Pertanika J. Sci. & Technol. 33 (5): 2069 - 2095 (2025)

PERTANIKA JOURNALS

SCIENCE & TECHNOLOGY

Journal homepage: http://www.pertanika.upm.edu.my/

Modified Cuckoo Search Algorithm Using Sigmoid Decreasing Inertia Weight for Global Optimization

Kalsoom Safdar[1,2], Khairul Najmy Abdul Rani[1,3], Siti Julia Rosli[1,3], Mohd Aminudin Jamlos[1,3]* and Muhammad Usman Younus[4,5]

[1]Faculty of Electronic Engineering and Technology, Universiti Malaysia Perlis, 02600 Arau, Perlis, Malaysia
[2]Department of Computer Science and Information Technology, University of Jhang, 35200, Jhang, Pakistan
[3]Advanced Communication Engineering, Centre of Excellence, Universiti Malaysia Perlis, 01000 Kangar, Perlis, Malaysia
[4]Department of Computer Science and Information Technology, Baba Guru Nanak University, 39120 Nankana Sahib, Pakistan
[5]Ecole Math´ematiques, Informatique, T´el´ecommunications de Toulouse, Universit´e de Toulouse, 31000 de Toulouse, France

ABSTRACT

Cuckoo Search (CS) is an evolutionary computational (EC) algorithm inspired by the behavior of a cuckoo bird, introduced by Yang and Deb in 2009 to solve various engineering-intensive optimization problems. However, this metaheuristic algorithm, CS, still suffers from premature convergence, mainly due to multimodal problems leading to local trap problems. This research introduces an adaptive swarm-based optimization approach to the CS algorithm, using the sigmoid decreasing inertia weight (DIW), which produces the modified Cuckoo Search using decreasing inertia weight (MCS-DIW) algorithm to tackle local trap problems. The paper shows that the proposed MCS-DIW depicts a better-controlled mechanism by adding the DIW with Lévy flight, for balanced exploration and exploitation in the global search domain. Moreover, this study presents an inclusive, experimental analysis of the widely used set of standardized benchmark test problems released by the Institute of Electrical and Electronics Engineers (IEEE) Congress on Evolutionary Computation (CEC) benchmark along with selected mathematical test functions to assess the performance of the MCS algorithm. The MCS-DIW algorithm is compared with other swarm intelligence (SI) algorithms to validate, including the original CS algorithm, Whale Optimization Algorithm (WOA), Sine Cosine Algorithm (SCA), and Search Sparrow Algorithm (SSA). The compiled simulation

ARTICLE INFO

Article history:
Received: 31 December 2024
Accepted: 29 April 2025
Published: 11 August 2025

DOI: https://doi.org/10.47836/pjst.33.5.03

E-mail addresses:
kalsoombajwa11@gmail.com (Kalsoom Safdar)
khairulnajmy@unimap.edu.my (Khairul Najmy Abdul Rani)
sitijulia@unimap.edu.my (Siti Julia Rosli)
mohdaminudin@unimap.edu.my (Mohd Aminudin Jamlos)
usman.younus@bgnu.edu.pk (Muhammad Usman Younus)
* Corresponding author

e-ISSN: 2231-8526

findings showed that the modified proposed CS algorithm, in most cases, performed better in attaining a low mean global minimum value, high convergence rate, and low central processing unit (CPU) processing time compared to other counterparts. The dynamic adjustment of inertia weight enhances optimization performance with an initial high inertia weight (e.g., 0.9) and promotes exploration, gradually decreasing to 0.2 for better exploitation. This proposed MCS-DIW approach provides faster convergence and has been proven to mitigate premature convergence. It reduces the number of iterations by 30-40% and achieves lower fitness values (e.g., 10-2) than static inertia weight, which often stabilizes at higher values (e.g., 10-1). In sum, the proposed MCS-DIW algorithm is proven to mitigate the local trap problems via an improved capability in searching for the global optimum.

Keywords: Cuckoo Search Algorithm, exploration, exploitation, inertia weight, local trap problem, premature convergence, swarm intelligence

INTRODUCTION

Optimization plays an essential role in engineering to solve critical problems, such as communication routing, system design, image reconstruction, network operations, and energy loss (A. Chakraborty & Kar, 2017; Saka et al., 2013). These problems depend upon the minimization or maximization of the given objective functions. Subsequently, proper assessment for algorithmic validation is required, including accuracy, convergence rate, and computational time of the designed system (Sekyere et al., 2024; Zangana et al., 2024). It also ensures efficient problem-solving mechanisms in complex systems under diverse constraints, including energy consumption (Adeyelu et al., 2024), communication limitations, image reconstruction errors (Habeb et al., 2024), and environmental factors related to the diverse changes in the search space. Accordingly, it emphasizes reliability, adaptability, and precision of algorithmic performance (Abualigah et al., 2024).

Moreover, metaheuristic algorithms provide guiding mechanisms to the new trending EA toward solving diverse optimization problems related to engineering (Luo et al., 2024). Generally, the term "metaheuristic" is composed of two Greek words covering two verbs, which are "to find" and "beyond, in an upper level". Moreover, metaheuristics can be defined based on two significant tactics: intensification and diversification (Abdul Rani et al., 2017; Adeyeye & Akanbi, 2024; Brezočnik et al., 2018; Saka et al., 2013). Additionally, intensification intends to choose the best optimal solution while searching for the best existing solution. However, diversification intends to explore the given search region efficiently, often by randomization (Brezočnik et al., 2018). Subsequently, modern evolutionary metaheuristic optimization algorithms such as SSA, Genetic Algorithm (GA) (Sohail, 2023), WOA (Mahmood et al., 2023), SCA, Ant Lion Optimization (ALO), and Particle Swarm Optimization (PSO) are aimed at carrying out a global search for three main reasons: solving diverse and large problems, getting faster convergence, and providing

robustness (Abdul Rani & Malek, 2011; Kwakye et al., 2024; Massat, 2018; Xue et al., 2023). Moreover, algorithmic efficiency can be considered the key attribute of metaheuristic algorithms. Accordingly, they started imitating the optimal features of nature, and mainly, they opted for the natural selection method using the fittest selection criteria, which can be seen in biology-based systems that have evolved over millions of years through natural selection (Adeyeye & Akanbi, 2024; Kwakye et al., 2024) .

Nowadays, some innovative researchers have introduced many nature-inspired optimization algorithms, for example, the Differential Evolution (DE) algorithm developed by Strom and Prince, functioned on crossover, selection and mutation operations using evolving populations. PSO inspired by fish and birds' schooling behavior (S. Chakraborty et al., 2023; Shi & Eberhart, 1998). However, Simulated Annealing (SA) uses a metal annealing mechanism (Chen et al., 2024). Comparatively, the Bat-inspired algorithm has an echolocation capability to sense the distance between its surroundings. Besides, Ant and Bee's algorithms worked through their foraging behavior using pheromone and concentration as a chemical messenger to control the given problem efficiently (Umar et al., 2024).

Though the CS algorithm is a nature-inspired, swarm intelligence-based evolutionary algorithm (EA) introduced by Yang and Deb in 2009 (Huang & Zhou, 2024). Basically, the CS algorithm used a cuckoo bird’s brood reproductive approach to increase their population. In addition, the CS algorithm is more prevalent and computationally efficient in discovering optimum solutions than its counterparts because it has fewer parameters than other nature-inspired algorithms. Moreover, the CS algorithm provides a potential solution using random groups of cuckoos inspired by the cuckoo’s brood parasitism that obligates the behavior of laying eggs in a habitat to the host nest. In this regard, recent research on CS algorithms provides different evolutionary mechanisms for better local and globally optimal solutions using nature-inspired optimization techniques, which provide solutions regarding different complex optimization-related engineering problems. However, its simplicity and balancing mechanism in exploration and exploitation provide ease in regenerating a better solution for various optimization problems (Abdul Rani & Malek, 2011; Aziz, 2022; Mohammed et al., 2023; Yang et al., 2024).

According to the above discussion, this paper aimed to propose an optimized variant of the Modified Cuckoo Search (MCS) algorithm using the sigmoid DIW, yielding MCS-DIW to solve premature convergence and local trap problems. This proposed MCS-DIW algorithm ensures better exploration to efficiently find the global optimal solution. This research investigated a detailed parametric study, which aimed to fine-tune different parameters of the proposed algorithm. Afterwards, the anticipated strategy is validated using several different mathematical benchmark functions. Moreover, the performance of the MCS-DIW algorithm has been verified using different SI algorithms, including original

CS, SCA, WOA, and SSA. Hence, it has been observed that the MCS-DIW algorithm outperformed for most of the testing functions compared to its counterparts.

The organization of this research paper is as follows: Section 2 deliberates research materials and methods along with previous advancements using the different improved/ modified variants of the CS algorithm and the increasing and decreasing inertia weight by shedding light on their innovative contributions to providing efficient algorithms. Moreover, it briefly discusses the proposed methodology using the proposed research design, including the implementation strategy and its working principles. Subsequently, Section 3 exhibits experimental techniques used for the simulation setup. Furthermore, Section 4 presents results and discussions, which provide the interpretation regarding the performance of a series of empirical experimental results using different benchmark functions. Finally, Section 5 concludes the overall findings of this paper.

MATERIALS AND METHODS

The proposed research involves modifying the CS algorithm to improve its performance for complex optimization tasks. The modified CS variant's effectiveness is thoroughly tested using a set of standard mathematical benchmark test functions, which includes unimodal and multimodal problems. Significant performance metrics such as computational efficiency, accuracy, and convergence rate are evaluated to validate the improvements associated with exploration and exploitation capabilities. Furthermore, the materials include benchmark mathematical functions like Ackley, Rosenbrock, Rastrigin, and Sphere, to test and evaluate the optimization performance. MATLAB-2020a is used to implement and simulate the modified CS algorithm.

However, this paper provides Wilcoxon and Friedman statistical analyses of a proposed modified variant of the CS algorithm using the sigmoid DIW. The major objective of conducting this research is to modify and improve the CS algorithm to enhance the performance and competency of the conventional CS algorithm for better exploration to find the global best fitness value in the given problem region. In this regard, a parametric study was conducted to fine-tune the internal parameters of the original CS algorithm. Afterwards, the effectiveness of the proposed MCS-DIW algorithm is evaluated through empirical simulations using seven different well-known mathematical benchmark functions compared with a few chosen SI algorithms, including original CS, SCA, WOA, and SSA.

Research Design

A step-by-step flowchart of the proposed research design is depicted in Figure 1. The flowchart outlines the steps of the MCS-DIW algorithm, incorporating the proposed modification along the overall research optimization process. The process begins with a feasibility study to initialize parameters, define iterations, and determine test function

dimensions. Subsequently, the CS algorithm parameters are fine-tuned to identify the optimal configuration and the maximum number of iterations for enhanced performance. Here is a detailed breakdown of each step:

Feasibility study of CS algorithm : A preliminary investigation is conducted to evaluate the feasibility and potential performance of the basic CS algorithm for the task at hand.

Fine-tune and evaluate the original CS algorithm's internal parameters through parametric studies : The original CS algorithm is fine-tuned and assessed using various population numbers and fraction probability values. These parametric studies will determine the best internal parameters to use for the MCS algorithm in the later stage.

Formulate a MCS algorithm and generate a random population or host nest using increasing inertia weight (IIW) and DIW : The CS algorithm is modified in two versions by introducing both IIW and DIW in generating a random host nest (population) to improve the optimizer's performance in exploring and exploiting potential global optimal solutions.

Validate the solution of the proposed MCS-IIW and MCS-DIW algorithms : The process checks the validity of the global optimal solution of both MCS-IIW and MCS-DIW algorithms iteratively. If the solution is invalid, the process returns to the previous step, fine-tuning and re-evaluating both MCS-IIW and MCS-DIW algorithms. If the solution is valid, the process will proceed until the existing number of iterations reaches the maximum number of iterations. After achieving the maximum number of iterations, the method identifies the best fitness value of both MCS-IIW and MCS-DIW, corresponding to the global optimum solution (best nest). Simulate and compare both MCS-DIW and MCS-IIW algorithms with the original CS algorithm. Finally, the proposed MCS-DIW is compared with other chosen SI algorithms, which include the SCA, WOA, and SSA.

Hence, this flowchart represents a typical process for optimizing solutions using an enhanced version of the CS algorithm with inertia weight, iterating through different potential solutions until the optimal one is found and validating the process along the

way (Figure 1). The final step compares the performance of this algorithm against other optimization-related algorithms.

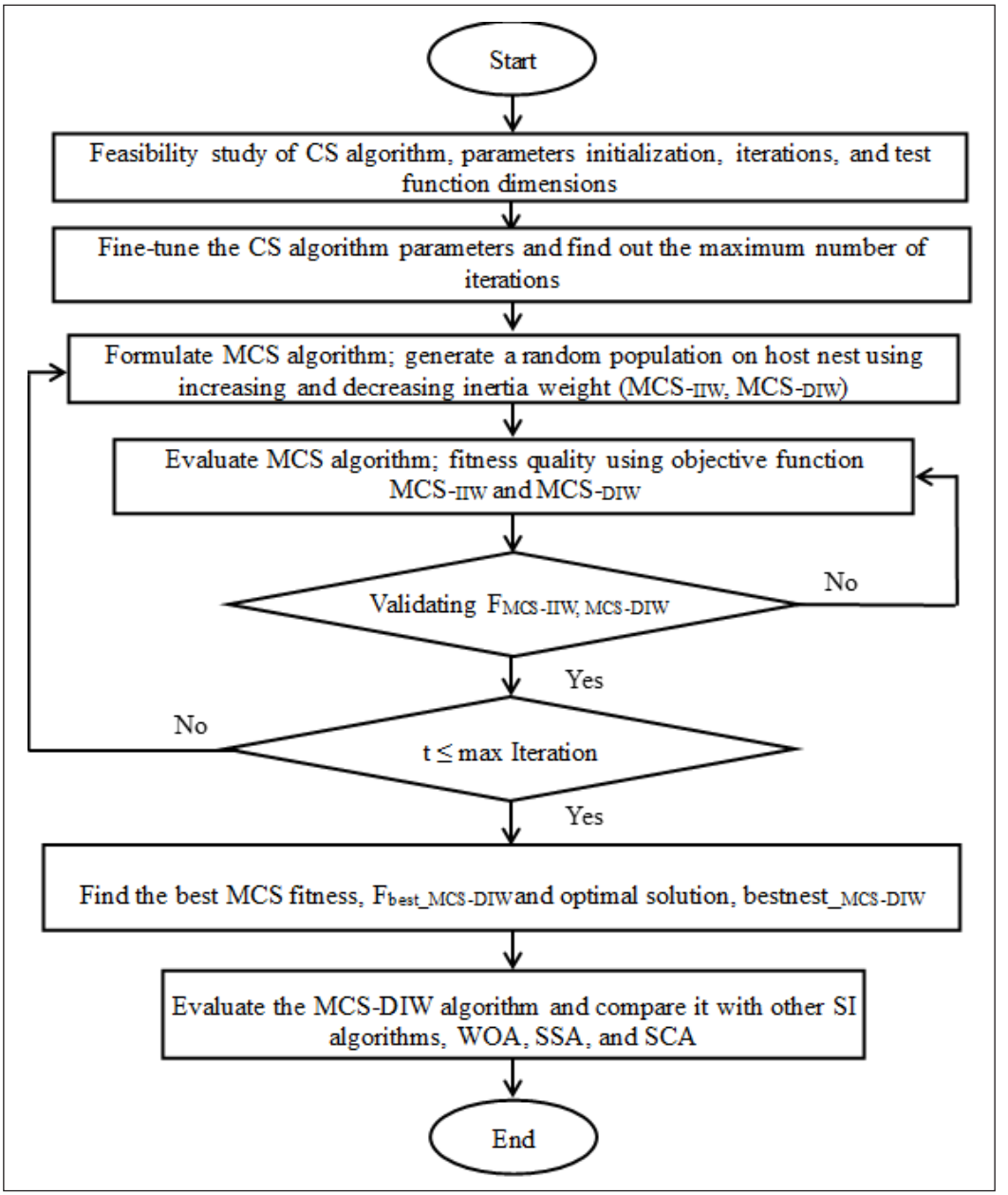


Figure 1. Research flowchart

Note. CS = Cuckoo Search; MCS = Modified Cuckoo Search; MCS-$_{IIW}$ = Modified Cuckoo Search using increasing inertia weight; MCS-$_{DIW}$ = Modified Cuckoo Search using decreasing inertia weight; SI = Swarm intelligence; WOA = Whale Optimization Algorithm; SSA = Search Sparrow Algorithm; SCA = Sine Cosine Algorithm

Modified Variant of CS Using Inertia Weight

The advanced nature-inspired optimization method known as the CS algorithm is motivated by the brood parasitism nature of cuckoo birds, who laid their eggs in the nests of other birds (Joshi et al., 2017; Meena et al., 2024). The CS algorithm, introduced by Yang and Deb in 2009, uses a local search mechanism to fine-tune solutions along with a combination of Lévy flight for global exploration (Almufti et al., 2025; Mareli & Twala, 2018). The algorithm begins by using a population of possible solutions for each representing a nest. Lévy flight, a kind of random walk that permits both tiny and large steps, strike a compromise between exploration and exploitation to produce new candidate solutions (Tian et al., 2024). A new one is substituted if an opted strategy doesn't increase the population's overall fitness. The population is updated iteratively by this method. This breeding behavior along Lévy flight is being applied to improve the efficiency of CS and solve the various optimization problems (Ahmad et al., 2025; Cuong-Le et al., 2021).

Inertia Weight

The idea of an inertia weight was initiated to maintain a balance between the exploration and exploitation mechanisms and eliminate the need for maximum iterations, I_{max}. It is an innovative enhancement of the CS optimization algorithm that integrates the concept of inertia weight commonly utilized in metaheuristic algorithms. The addition of inertia weight introduces an adaptive parameter that modulates the magnitude of step changes during the search, enhancing the exploration and exploitation abilities of the algorithm. Alongside, by incorporating inertia weight, the algorithm dynamically balances the trade-off between local exploitation and global exploration, allowing for smoother convergence and improved convergence accuracy (Choudhary et al., 2023). This novel extension holds significant promise for enhancing the performance of Cuckoo Search in various optimization tasks across diverse domains. The inertia weight (w) ensured a controlled transition of the cuckoos by adding the weight to the contribution of the previous solution (Zdiri et al., 2021). Eq. [1] uses the Lévy flight to offer the new optimal solution using the inertia weight, which is mathematically shown in the following equation. Updated CS algorithm, Lévy flight equations using inertia weight.

$$X_i^{(t+1)} = w * x_i^t + \alpha \cdot Le'vy(\lambda) \quad [1]$$

$$Le'vy(\lambda) \sim \frac{u}{|v|^{1/\lambda}} \quad [2]$$

In Eq. [2], u and v are drawn from normal distributions. The following equations represent the weight update mechanism, typically used in optimization algorithms like

MCS or similar swarm intelligence method. The weight W_k shown in Eq. [3] dynamically changes over iterations to ensure stability to explore and exploit during the search process.

Moreover, Eq. (3) (Y. Zheng et al., 2003) is utilized by DIW and IIW in Eq. [4] (Y. Zheng et al., 2003). Subsequently, the value of u is defined in Eq. [5] (Y. Zheng et al., 2003). Accordingly, as shown in Table 1, which provides all the parameters used in the inertia weight equations.

$$W_k = \frac{W_{Start} - W_{end}}{\left(1+e^{-u*(k-n*iter)}\right)} + W_{end} \quad [3]$$

$$W_k = \frac{W_{Start} - W_{end}}{\left(1+e^{u*(k-n*iter)}\right)} + W_{end} \quad [4]$$

$$u = 10^{(\log(iter)-2} \quad [5]$$

W_k is calculated as a combination of the initial weight W_{start} and the final weight W_{end}, modulated by a sigmoid function. Further the term $e^{-u*(k-n*iter)}$ defines the rate of decay, where k is the existing iteration and n_{iter} is the total number of iterations. This ensures that W_k transitions smoothly from W_{start} to W_{end} as iterations progress.

Moreover, in Eq. [5], the parameter $u = 10^{(\log(iter)-2}$ adapts the decay rate based on the current iteration. It fine-tunes how quickly the weight shifts from exploration (higher weights) to exploitation (lower weights) as the algorithm progresses.

Table 1
Parameters details for the inertia weight equations

Symbol	Name
$X_i^{(t+1)}$	A new solution for the *ith* cuckoo at iteration t + 1
x_i^t	Current solution
α	Scaling factor step size
$Le´vy(\lambda)$	Represents the Lévy flight distribution
W	Inertia weight
W_{start}	Starting inertia weight at the given run
W_{end}	Ending inertia weight at the given run
U	Constant to adjust the shape of the function
N	Constant to set the duration of the function

Later, this weight scheduling mechanism allows the algorithm to focus on early-stage global exploration by assigning higher weights and gradually shifting towards local exploitation in later stages, improving convergence to the optimum solution. This adaptive

weight adjustment improves the CS algorithm's balance between finding diverse solutions and refining the best solutions over iterations.

MCS Algorithm – Pseudocode

The pseudocode of the proposed MCS algorithm is provided below. Previously, the inertia weights were implemented using constant (Sekyere et al., 2024) and dynamic (Nickabadi et al., 2011) values for all possible solutions and dimensions used for the complete search domain. In Eq. [1], the fitness function f(X) is evaluated for each solution X to determine its quality using inertia weight. The goal is to maximize or minimize this fitness, depending on the optimization problem. Conversely, dynamic values used two different increasing and decreasing approaches. For increasing, a small value of inertia weight will increase linearly or nonlinearly to a linearly increasing large value. For decreasing, a large value of inertia weight will decrease linearly or nonlinearly to a linearly small value. A large value of inertia weight will foster the possibility of global search convergence, and a small value of inertia weight has more potential for local search than a large value of inertia weight. As provided, the following MCS algorithm, where the modification is performed at step no. 07 using Eq. [1, 2] to add inertia weight to get the fastest convergence compared to the original CS algorithm.

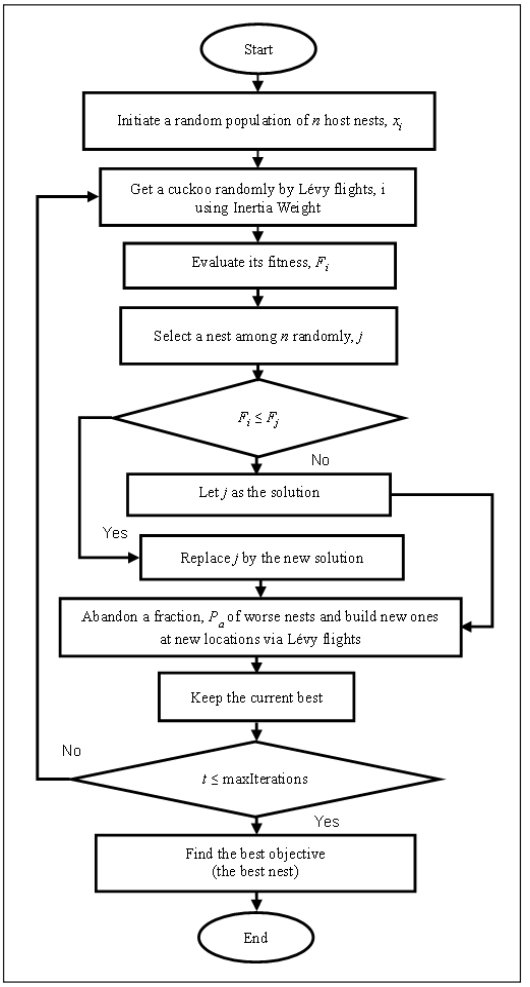


Figure 2. Modified Cuckoo Search Algorithm flow chart

Note. n = Population; t = Iteration; x_i = Host nest; x_j = Random nest; F = Fitness value; P_a = Discovery probability

The nature-inspired modified cuckoo search MCS-DIW metaheuristic algorithm begins by initializing the population nests, which are randomly distributed candidate solutions among the given search space. The next step shows exploration and exploitation using a random walk mechanism, generating new solutions using Lévy flights by adding inertia weight (Figure 2).

Furthermore, the bio inspired optimization strategy of cuckoo species is based on the brood parasitic behavior , which employs random walk using Lévy flight and discovery mechanism to find global optimal solution efficiently.

In addition to enhancing the modified variant of the CS algorithm, the inertia weight is applied to the random walk solution to improve the performance of MCS-DIW. Using the inertia weight, the fitness value of each new solution is assessed repeatedly each time the new and better solution is replaced with the existing one if the performance of the new solution is better than the existing solution. This scenario mimics the strategy of the host bird cuckoo search involving detecting and eliminating foreign eggs. This step is repeated continuously according to the given condition until the stopping criterion is met, including achieving convergence and reaching the maximum number of iterations. In this regard, the first step of modified cuckoo search algorithm includes the definition of objective function and parameter initialization, such as maximum iterations *t*, host nest *n*, discovery probability *Pa*, and the parameters given in the Table 1. In the next step, the control enters the main loop to evaluate the fitness value of each host nest (solution) using the given objective function. Accordingly, inertia weight given in Eq. [1, 2] will be updated dynamically to ensure smooth transition to find the global optima over the given iterations. Further, the control enters a nested For loop to generate a new solution using Lévy flight with controlled step size using inertia weight. Subsequently, the latest fitness value will be evaluated to see if it is better than the existing solution. Under the upper and lower bound conditional check, the better optimal fitness value will be replaced with the existing one to enhance diversity. In other words, the new random solutions will be replaced with the fraction of nests (solutions) as the discovery probability *Pa* is determined. This process of global best selection will be executed until it reaches the maximum number of iterations. Lastly, found the optimal solution. Hence, it has proven that combining MCS-DIW with better selection, randomization with inertia weight, and nest replacement provides an efficient solution to the given problems, ensuring rapid optimal convergence and balanced exploitation and exploration.

Modified Cuckoo Search Algorithm

1. Start
2. Objective function $f(x), x = (x_1, x_2, x_d)^T$
3. Generate, initial population of *n* host nests,
4. $x_i, i = 1,2,3, \ldots n$
5. while *t* < max iteration or stopping criterion
6. Get a cuckoo randomly by Lévy flight
7. **Evaluate its quality/fitness F_i using DIW for each nest Eq. [1, 2]**

8. Select a nest randomly from n (say, j)
9. if $F_i \leq F_j$
10. Replace j with a new solution
end if
11. A fraction, P_a of worst nests are abandoned, and new ones are reconstructed
12. Keep the best solutions (or nests with quality solutions)
13. Rank the solutions and find the current best
14. end while
15. Postprocess results and visualization
16. End

Data Analysis and Interpretation Techniques

To validate the efficiency of the proposed MCS algorithm, real-time performance will be analyzed, and tests will be carried out to identify the improvements. Thus, unimodal and multimodal-based objective functions were used to test the working of the MCS algorithm. Table 2 demonstrates eight out of 23 classical sets of test functions used for the MCS algorithm performance analysis (Cheraghi et al., 2023).

Table 2
Selected eight test functions (Mareli & Twala, 2018)

Problems	Name	Range
F1	Rosenbrock's function	[-2.048, -2.048]
F2	Ackley's function	[-32.768,32.768]
F3	Griewanks's function	[-600,600]
F4	Rastrigin's function	[-5.12,5.12]
F5	Nocontinuous Rastrigin's function	[-5.12,5.12]
F6	Schewfel's function	[-500,-500]
F7	Weierstrass's function	[-0.5,-0.5]
F8	E_Scaffer's F6 function	[-100,100]

The selected test functions encompass a range of optimization landscapes, each posing unique challenges to optimization algorithms. Additionally, the evaluation extends to include the Rotated Elliptic, Rotated Bent Cigar, and Rotated Discus functions (Ghiaskar et al., 2024; Thaher et al., 2024; W. Zheng et al., 2023). The formulas, domains, and ranges of these functions are meticulously defined to provide a consistent basis for comparison.

Further, these benchmark functions facilitate a comprehensive assessment, allowing for thoroughly validating the proposed methods' performance across various optimization landscapes. By subjecting the proposed methods to these standardized tests, the research

aims to establish their effectiveness, efficiency, and adaptability in solving real-world optimization problems (Bharambe et al., 2024; Liu et al., 2022; Wei & Niu, 2022). This validation and verification process highlights the robustness and practical applicability of the proposed method, providing a credible foundation for its integration into optimization tasks.

Experimental Techniques

Experimental Setup

An experimental setup was deployed to validate the effectiveness of the proposed MCS algorithm. Eight different categories of mathematical benchmark functions were used to test the efficiency of the proposed algorithm. Accordingly, MATLAB R2020a was used for coding on a Core (TM) 1.61 GHz system for simulation experiments.

Parametric Study

The parametric study is performed using optimization test functions. In this regard, the original cuckoo search algorithm was tested using different values, including population and probability.

Simulation Findings

The findings of all the initial results are presented in this section. In this regard, 23 optimization mathematical test functions are used to fine-tune the internal parameters of the original CS algorithm, where the F1 to F3 test functions were unimodal. In contrast, the F4 to F16 test functions were multimodal. Accordingly, Figure 3 depicts the comparison of different mathematical test functions, including F1 to F23, using 500 iterations, where the original CS algorithm exhibited the fast convergence curve in four out of 23 test functions, namely, the F6, F12, F13, and F22 test functions. In most evaluations, the convergence exhibited higher performance in F6, F12, F13, and F22 compared with the adjusted benchmark functions and the original CS algorithm. Moreover, to fine-tune the internal parameters, the population is set to 30 and the fraction probability is set to 0.5, running for 3000 iterations. The resultant functions show faster convergence with more exploration of the given problem. A comparison was performed with other Swarm Intelligence (SI) algorithms to ensure a fair evaluation of the metaheuristics.

Furthermore, the analysis of the abovementioned comparison shows that the CS algorithm performed better in four out of 23 functions, including F6, F12, F13, and F22, respectively. As shown in Figure 4 (a), the selected test functions demonstrated the algorithm's performance using logarithmic fitness values to minimize the objective function using 3000 iterations. These resultant functions are compared further to get the

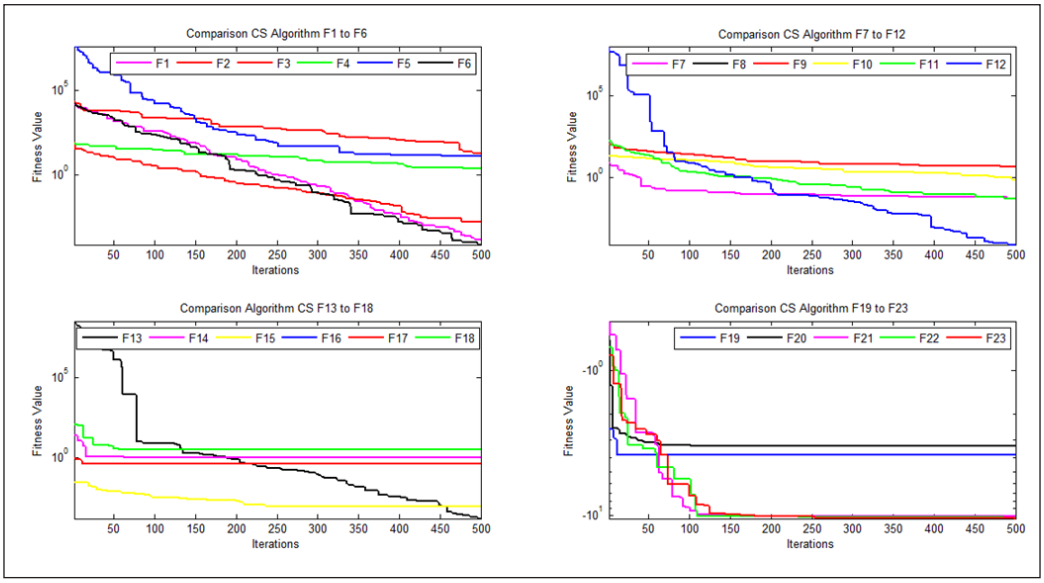


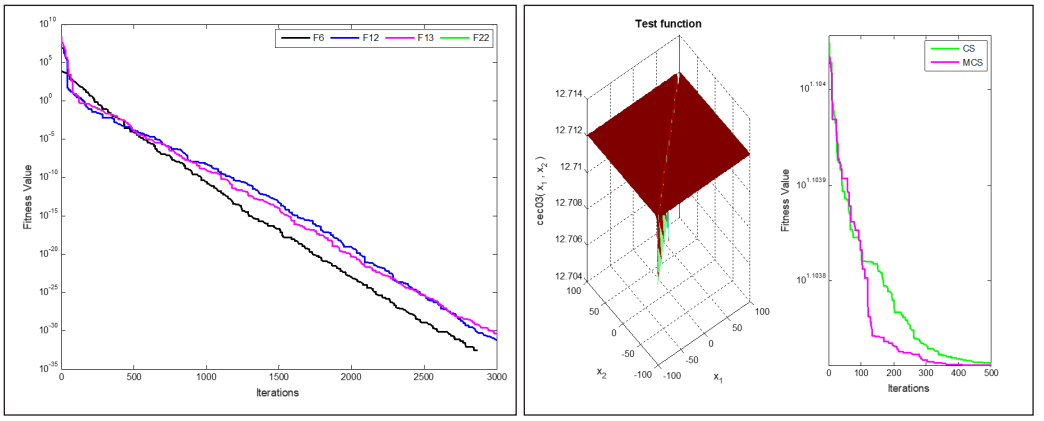
Figure 3. Convergence curve of the original Cuckoo Search (CS) Algorithm using the 23 different mathematical test functions for F1 to F23

best optimal solution to fine-tune the internal parameters. Additionally, the results show that F6 is being explored more deeply in the search area than F12, F13, and F22. Hence, the F6 function indicates a black line depicting the fastest convergence among all other test functions, achieving the lowest fitness value within the fewest iterations.

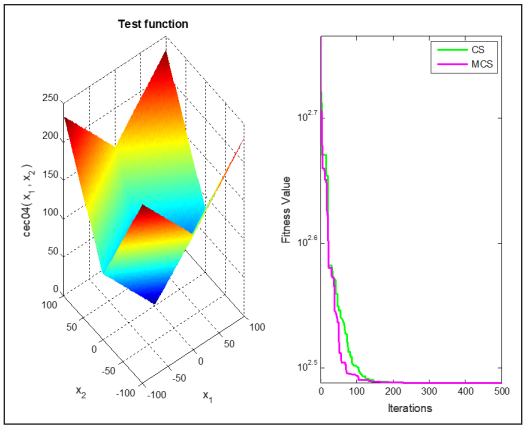
Subsequently, as depicted in Figures 4 (b) and (c), the CEC benchmark functions were evaluated using the proposed MCS algorithm, along with the original CS algorithm, over 500 iterations. Where the performance of MCS is better as compared to the original CS algorithm, this ensures the fastest convergence to find the global optimal solution.

Figure 5 shows the convergence curves with different population values assigned 10, 20, 30, 40, and 50, with a fraction probability value of 0.5 using the F6 test function because F6 outperformed in the above shown Figures 3 and 4 out of 23 mathematical test functions. The fitness value is plotted against the different number of iterations out of 3000 iterations. The results show that as the size of the population decreases, the convergence rate becomes higher. In contrast, an appropriate selection of population size is to balance the solution quality and computational efficiency. Additionally, the given scenario illustrates that the population size of 10 depicts a higher computational cost for the best optimization problem compared to other population values.

Moreover, the probability parameter is also critical in controlling the balance between exploration and exploitation in the algorithm using the F6 test function. For this scenario, a range of probability values (0.1, 0.25, 0.5, 0.75, and 1) has been compared to estimate the better performance of the CS algorithm optimization c=0.5 (red line) achieves the best



(a) (b)



(c)

Figure 4. Convergence curve of the original Cuckoo Search (CS) Algorithm using the best test functions along Congress on Evolutionary Computation (CEC) benchmark functions
Note. MCS = Modified Cuckoo Search

overall performance, combining fast convergence with a low final fitness value. Lower probabilities, while slower, may still be useful for problems requiring more extensive exploration. Thus, the fraction probability of 0.5 had the lowest global optimal solutions compared to different values (Figure 6).

RESULTS AND DISCUSSION

First, a comparison between increasing and decreasing sigmoid inertia weight has been performed using Eqs. (3) and (4), as shown in Figure 7, using the specifications listed in

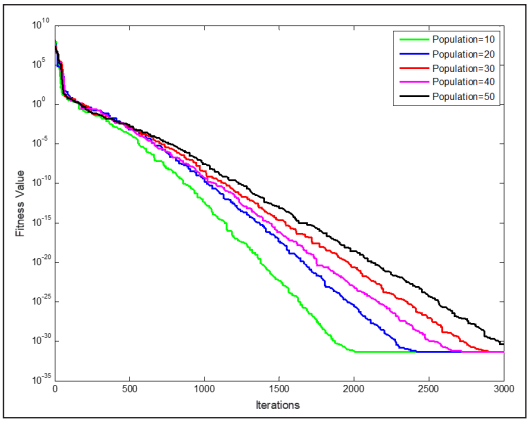


Figure 5. Comparison of parametric results with different values of population in the Cuckoo Search Algorithm

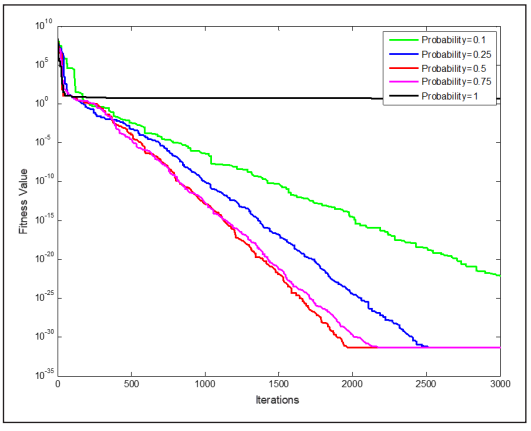


Figure 6. Comparison of parametric results with different values of fraction probability in Cuckoo Search Algorithm

Table 3 for the performance comparison of both increasing and decreasing inertia weight. The MCS algorithm using the DIW performed better than the MCS algorithm using the IIW to attain faster convergence after 1300 iterations (Figure 7).

The MCS-DIW outperforms both MCS-IIW and CS in terms of achieving the lowest fitness value and fastest CPU time due to its dynamic adjustment of the inertia weight. Subsequently, the Big O computational complexity of the proposed MCS-DIW

Table 3
Parameters of the proposed Modified Cuckoo Search Algorithm

Serial no.	Name	Values
1.	Population	10
2.	Probability	0.5
3.	W-Start	400
4.	W-End	200
5.	Iterations	2000

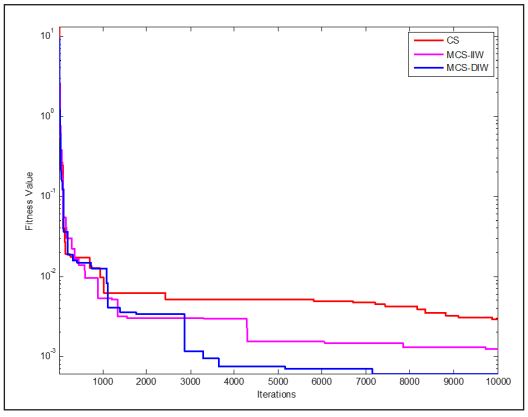


Figure 7. Comparison between original Cuckoo Search (CS) with proposed Modified Cuckoo Search (MCS) Algorithm
Note. MCS-IIW = Modified Cuckoo Search using increasing inertia weight; MCS-DIW = Modified Cuckoo Search using decreasing inertia weight

algorithm using the additional computation of decreasing inertia weight remains the same as compared to the original CS algorithm, which is $O(n \times Maxiter)$, where n represents the number of nests, and *MaxIter* is the maximum number of iterations. The proposed MCS-DIW algorithm also improves the convergence stability, global search efficiency, and solution accuracy. Thus, the modified algorithm is computationally richer and more robust compared to the original CS algorithm. Ensuring improved convergence and better global search behavior with manageable computational load.

Furthermore, as shown in Figure 7, the MCS-DIW (blue line) consistently reaches lower fitness values faster and stabilizes below 10^{-2} approaching 10^{-3}, whereas MCS-IIW (magenta line) plateaus earlier at a higher fitness value, and CS (red line) converges more slowly and stagnates around 10^{-1}. The key advantage of MCS-DIW lies in its adaptive inertia weight mechanism, where the inertia weight decreases over time starting high (e.g., around 0.9) to promote exploration in the early stages, allowing the algorithm to traverse large areas of the search space and avoid local minima. As iterations progress, the inertia weight decreases (e.g., down to 0.2), which helps the algorithm focus on exploitation, refining solutions in promising regions for more precise optimization. This dynamic control prevents premature convergence and ensures that MCS-DIW maintains a balance between exploration and exploitation, resulting in faster convergence to lower fitness values with fewer iterations, which in turn reduces CPU time. In contrast, MCS-IIW has a more static inertia weight adjustment, and CS lacks adaptive mechanisms, leading to slower and less efficient performance.

The results depicted in Figure 7 demonstrate the superior performance of the MCS-DIW algorithm (blue line) compared to MCS-IIW (magenta line) and CS (red line). The

MCS-DIW reaches a significantly lower fitness value of approximately 10^{-3} within 2800 to 7100 iterations, whereas MCS-IIW stabilizes at a fitness value near 10^{-1}, and CS plateaus around one after approximately 1000 iterations out of a total of 10000 iterations. In terms of efficiency, MCS-DIW requires fewer iterations and thus achieves faster CPU time as it converges more quickly toward an optimal solution. In contrast, MCS-IIW and CS take longer to converge and stabilize at suboptimal fitness values, highlighting the advantage of MCS-DIW's dynamic inertia weight in both accuracy and computational efficiency.

Furthermore, the proposed MCS-DIW and original CS, SSA, SCA, and WOA have been compared using four selected optimization test functions, as shown in Figures 8-11. Performance indicators are provided in Tables 4 and 5 for comparison purposes.

Moreover, Figures 8-11 show the results of 4 different mathematical benchmark test functions, including (a) F4, (b) F5, (c) F7, and (d) F9, using different SI algorithms, including MCS-DIW, CSA, WOA, and SSA, to emphasize the significance of balancing between exploration and exploitation to attain accuracy and robustness in optimization related problems. These different swarm intelligence algorithms showed various convergence rates for objective function minimization using the logarithmic scale fitness values against 2000 iterations for each of the four selected functions.

Subsequently, in Figure 8, using the F4 function, the proposed MCS-DIW algorithm has significantly outperformed as compared to other selected SI algorithms and demonstrated by the magenta line, achieving the fastest convergence by the lowest fitness value ~10^{-20} against 2000 iterations and continued to improve the exploration process capabilities along

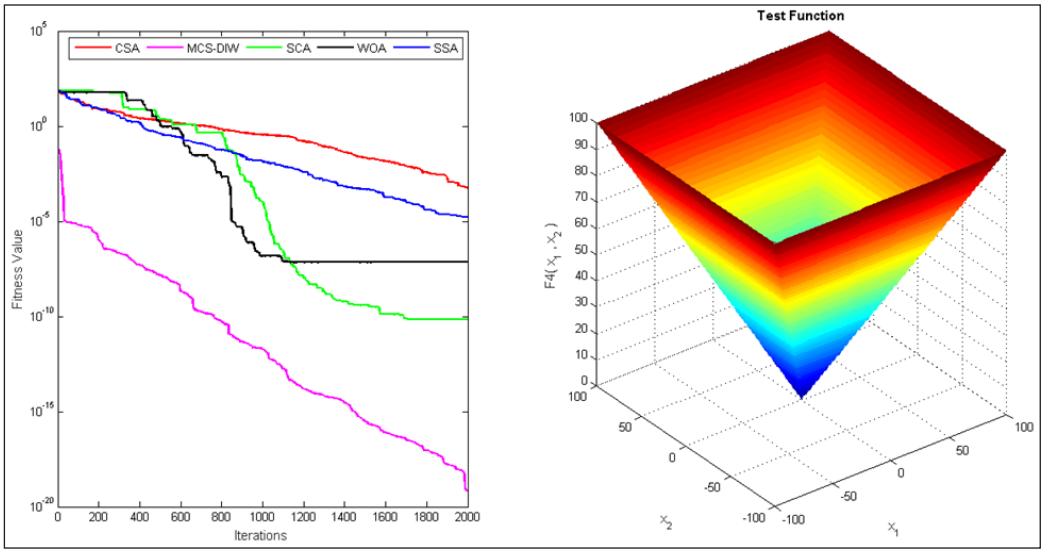


Figure 8. Comparison of the Modified Cuckoo Search using Decreasing Inertia Weight (MCS-DIW) algorithm with Cuckoo Search Algorithm (CSA), Sine Cosine Algorithm (SCA), Search Sparrow Algorithm (SSA), and Whale Optimization Algorithm (WOA) using the test function F4

deep exploitation to find the global optimal value over time. Afterwards, the SCA showed a green line illustrating a slower convergence rate using the moderate fitness value, ~10^{-5}, with early stagnation of the overall search space. Next, the WOA represents a black line and stagnates quickly with ~10^{-2} fitness values over 800 iterations, which depicts a local trap problem and premature convergence. The blue line represents SSA, reflecting an imbalance between exploitation and exploration with slower convergence than MCS and SCA. Hence, SSA reaches ~10^{-3} fitness values over 1200 iterations. Besides, the original CS algorithm represents a red line along its fine-tuned parameters and showed better performance than WOA and SSA around 1500 iterations, achieving the fitness value of ~10^{-4}. However, the red line demonstrates the original CS algorithm depicted, which showed a slower convergence rate as compared to the proposed MCS-DIW algorithm. Additionally, it has shown intermediate performance and is less efficient than the MCS-DIW and SCA algorithms, along with limited exploration capability to find the global optimal value.

In Figure 9, using the F5 function to minimize the objective function, the proposed MCS-DIW algorithm has maintained a good balance between exploration and exploitation to find the global optimal solution as compared to other selected SI algorithms. It has been demonstrated by the magenta line, achieving the fastest convergence by the final lowest fitness value ~10^{-20} by approximately 200 iterations, and it has continued to improve the exploration process capabilities along deep exploitation to find the global optimal value over time by reaching ~10^{-5} against 1000 iterations and drops to ~10^{-15} fitness value. Accordingly, it efficiently achieves the global optimal value compared to the other selected SI algorithms.

Besides, the blue line represents SSA, reflecting a gradual imbalance between exploitation and exploration with slower convergence than MCS and CS. Hence, SSA reaches ~10^{-2} fitness values without further improvements with 500 iterations, leading to suboptimal solutions. Afterwards, the SCA showed a green line exhibiting an initial rapid convergence rate but got trapped in local search space early. As a result, it only reached the moderate fitness value, ~10^{-3} with 200 iterations, with early stagnation of the overall search space, and afterwards there were no significant improvements.

Subsequently, the red line demonstrates that the original CS algorithm depicts a slower convergence rate using the moderate fitness value, ~10^{-12}, with early stagnation of the overall search space over 1500 iterations. Over 500 iterations, the fitness values have been improving from ~10^{-4} to ~10^{-8} at 1000 iterations. However, the original CS algorithm is a bit slower as compared to the modified variant in reaching the global optimal level.

Next, the WOA represents a black line and stagnates quickly with ~10^{-1} fitness value over 200 iterations, which depicts a local trap problem and premature convergence. Hence, the proposed MCS-DIW provides satisfactory results to find the global optimal solution as compared to early stagnation and limited optimization potential of SCA, SSA, and WOA comparative variants.

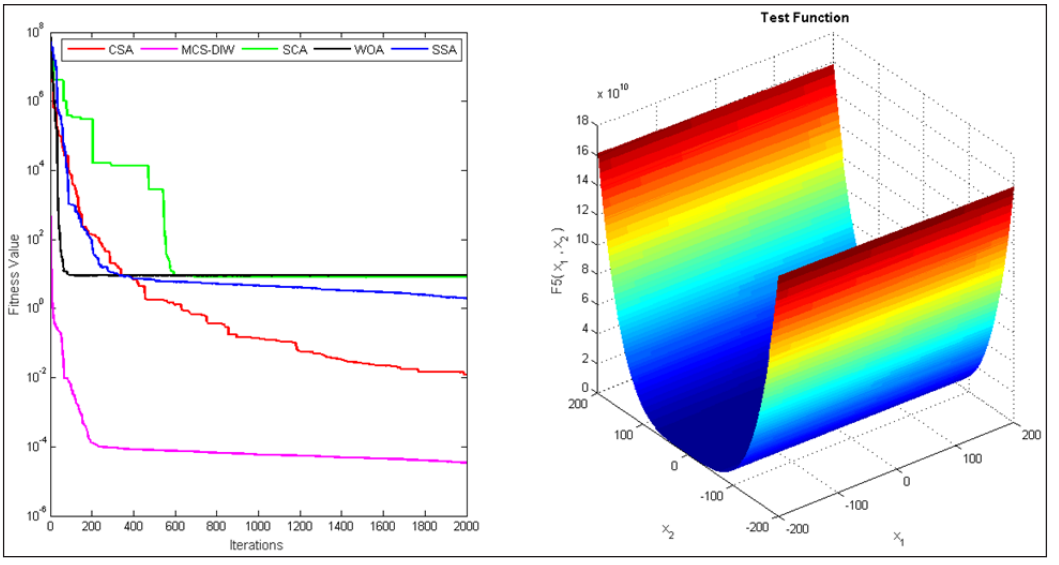


Figure 9. Comparison of the Modified Cuckoo Search using decreasing inertia weight (MCS-DIW) alg1orithm with Cuckoo Search Algorithm (CSA), Sine Cosine Algorithm (SCA), Search Sparrow Algorithm (SSA), and Whale Optimization Algorithm (WOA) using the test function F5

In Figure 10, using the F7 function to minimize the objective function, the proposed MCS-DIW algorithm has maintained a good balance between exploration and exploitation to find the global optimal solution as compared to other selected SI algorithms. It has been demonstrated by the magenta line, achieving the fastest convergence by the lowest fitness value ~10^{-14} by approximately 600 iterations and drops to ~10^{-4} fitness value by 200 iterations. Accordingly, it efficiently achieves the global optimal value compared to the other selected SI algorithms by keeping a balance between exploration and exploitation.

Besides, the blue line represents SSA, again reflecting an imbalance between exploitation and exploration with higher fitness values and slower convergence. Hence, SSA reaches ~10^{-3} fitness values without further improvements with 1000 iterations, leading to early suboptimal solutions. Afterwards, the SCA showed a green line exhibiting an initial rapid convergence rate but got trapped in the local search space early. As a result, it has comparatively demonstrated effective but less robust and slower performance, providing a fitness value of ~10^{-12} with 1200 iterations.

Subsequently, the red line demonstrates that original CS algorithm depicts a slower convergence rate using the moderate fitness value, ~10^{-10}, with moderate performance over 1800 iterations. Still, according to the results, it is less efficient than MCS-DIW and SCA. Next, the WOA represents a black line and stagnates quickly with ~10^{-2} fitness values throughout the iterations, which depicts a local trap problem and premature convergence.

Hence, the proposed MCS-DIW provides satisfactory results with the lowest fitness value to find the global optimal solution as compared to other selected SI algorithms.

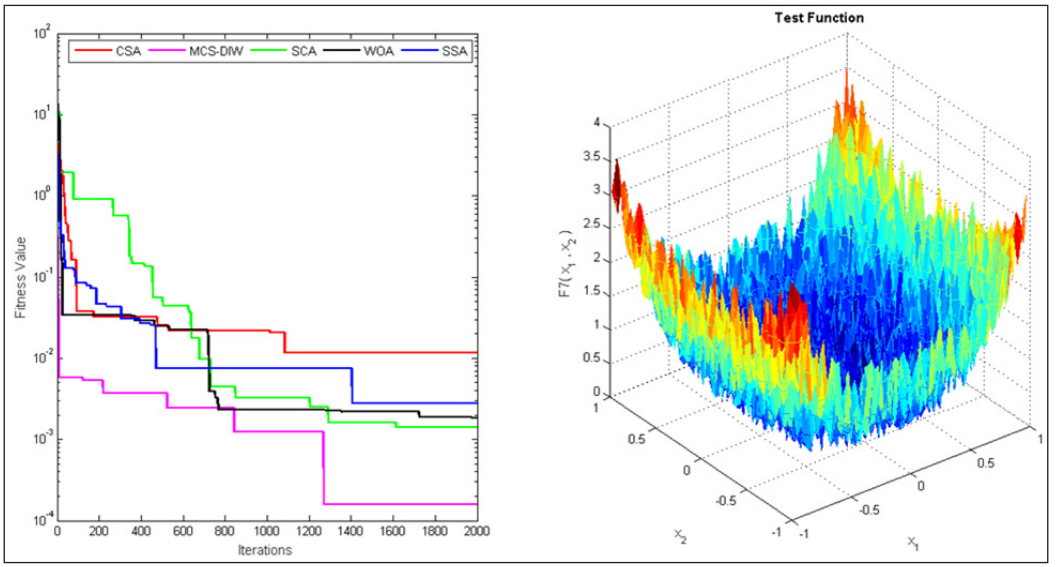


Figure 10. Comparison of the Modified Cuckoo Search using decreasing inertia weight (MCS-DIW) algorithm with Cuckoo Search Algorithm (CSA), Sine Cosine Algorithm (SCA), Search Sparrow Algorithm (SSA), and Whale Optimization Algorithm (WOA) using the test function F7

In Figure 11, using the F9 function to minimize the objective function, the proposed MCS-DIW algorithm has maintained a good balance between exploration and exploitation to find the global optimal solution compared to other selected SI algorithms. It has been demonstrated by the magenta line, achieving the fastest convergence by the lowest fitness value ~10^{-4} by approximately 1200 iterations and exhibiting steady improvement. Accordingly, it efficiently achieves the global optimal value compared to the other selected SI algorithms by balancing exploration and exploitation.

Besides, the blue line represents SSA, exhibiting slower convergence and limited exploration capabilities. Hence, SSA reaches ~10^{-2} fitness values without further improvements with 1200 iterations, leading to early suboptimal solutions. Afterwards, the SCA showed a green line exhibiting an initial rapid convergence rate of around 1000 iterations but got trapped in local search space early. As a result, it has demonstrated an effective but less robust and slower performance, providing a fitness value of ~10^{-2}.

Subsequently, the red line demonstrates the original CS algorithm depicts a slower convergence rate using the moderate fitness value, ~10^{-3}, with moderate performance over 1500 iterations, but according to the results, it is showing better performance as compared to SCA, SSA, and WOA. Next, the WOA represents a black line; it is less effective as it

stagnates quickly with ~10^{-1} fitness value with 600 iterations without further improvements, which depicts a local trap problem and premature convergence.

Hence, the proposed MCS-DIW provides satisfactory results with the lowest fitness value to find the global optimal solution as compared to other selected SI algorithms.

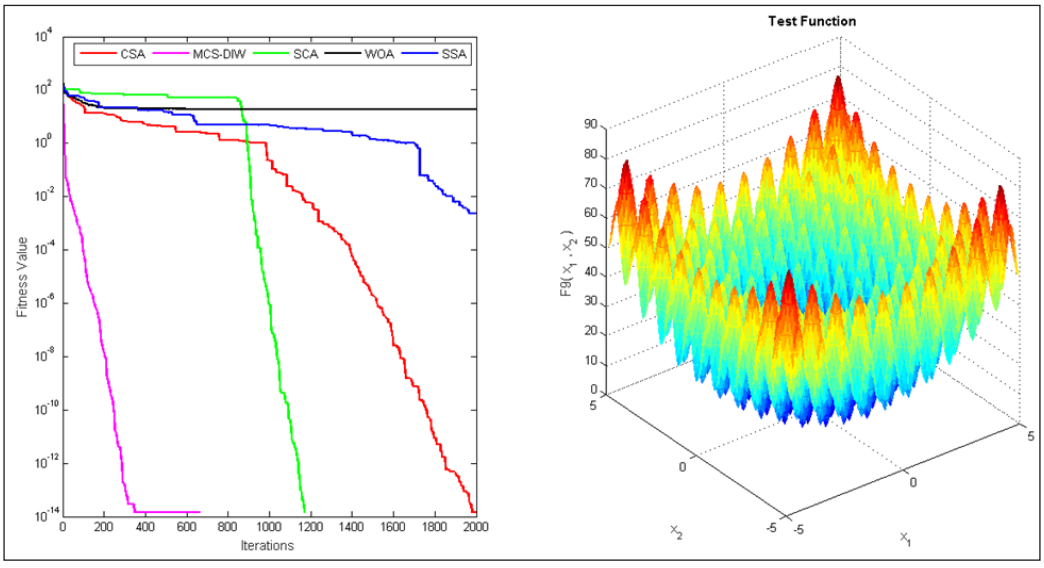


Figure 11. Comparison of the Modified Cuckoo Search using decreasing inertia weight (MCS-DIW) algorithm with Cuckoo Search Algorithm (CSA), Sine Cosine Algorithm (SCA), Search Sparrow Algorithm (SSA), and Whale Optimization Algorithm (WOA) using the test function F9

Additionally, the mean and standard deviation (Std) comparison results are provided in Table 4, where all the comparative SI algorithms' results are compared using 2000 iterations and the 4 selected mathematical test functions. This comparison showed that the proposed MCS-DIW provides better mean values compared to other selected algorithms.

Furthermore, Table 5 depicts the best cost values and CPU processing time comparison, and comparatively, the proposed algorithm showed efficiency with less time compared to the selected SI algorithms.

To validate the proposed MCS-DIW's effectiveness, Figure 12 depicts the statistical analysis results for the Wilcoxon and Friedman statistical tests, where the final fitness values of the different optimization SI algorithms along proposed MCS-DIW have been evaluated. The proposed MCS-DIW showed a better convergence with significantly lower fitness values than the other comparative swarm intelligence algorithms.

Table 6 indicates that the Wilcoxon results provide p-values < 0.05, indicating that the proposed MCS-DIW significantly outperforms its counterparts.

Table 4
Mean and standard deviation global minimum comparison

Test functions	MCS-DIW mean/std.	CS mean/std.	SCA mean/std.	WOA mean/std.	SSA mean/std.	Iterations
F4	150.5354/ 37.02811	211.1881/ 56.34211	519.586/ 33.1708	190.4882/ 12.4564	635.3371/ 38.45453	2000
F5	23802.9639/ 28.5646	58123.004/ 40.5653	1207313.7257/ 39.87865	69099.5753/ 39.65743	173143.7317/ 17.45434	2000
F7	0.058221/ 0.03221	0.12902/ 0.10902	1.1749/ 1.1130	0.11596/ 0.12484	0.19193/ 0.12184	2000
F9	0.035221/ 0.05822	0.22602/ 0.12902	1.1479/ 1.1749	0.12596/ 0.11596	0.18292/ 0.19193	2000

Note. CS = Cuckoo Search; MCS-DIW = Modified Cuckoo Search using decreasing inertia weight; WOA = Whale Optimization Algorithm; SCA = Sine Cosine Algorithm; SSA = Search Sparrow Algorithm; std. = Standard deviation

Table 5
Best cost and CPU processing time comparison

Problems	MCS-DIW best cost/ CPU time consumption	CS best cost/ CPU time consumption	SCA best cost/ CPU time consumption	WOA best cost/ CPU time consumption	SSA best cost/ CPU time consumption	Iterations
F4	1.9557e-114/ 0.57213	2.8581e-17/ 1.4351	0/ 0.14198	1.1235e19/ 1.7858	0.45196/ 0.35833	2000
F5	1.0788e-88/ 0.5828	1.4398e-09/ 1.7483	0/ 0.16401	1.9121e08/ 1.6112	0.00053215/ 0.38028	2000
F7	0.005398912.1/ 4	0.0093989/ 10.0351	0.01339591.267/ 3	6.1368/ 4.9487	1.8422/ 2.761	2000
F9	0.0068648/ 1.3327	0.010044/ 1.2569	0/ 0.12485	0.0097856/ 0.43367	0.082626/ 0.33124	2000

Note. CS = Cuckoo Search; MCS-DIW = Modified Cuckoo Search using decreasing inertia weight; WOA = Whale Optimization Algorithm; SCA = Sine Cosine Algorithm; SSA = Search Sparrow Algorithm; CPU = Central processing unit

Table 6
Test Wilcoxon results for MCS-DIW

Comparative algorithms	Wilcoxon Statistic	*p*-value
MCS-DIW vs CS	0.0	0.0023
MCS-DIW vs SCA	0.0	0.0017
MCS-DIW vs WOA	1.0	0.0034
MCS-DIW vs SSA	0.0	0.0028

Note. MCS-DIW = Modified Cuckoo Search using decreasing inertia weight; CS = Cuckoo Search; SCA = Sine Cosine Algorithm; WOA = Whale Optimization Algorithm; SSA = Search Sparrow Algorithm

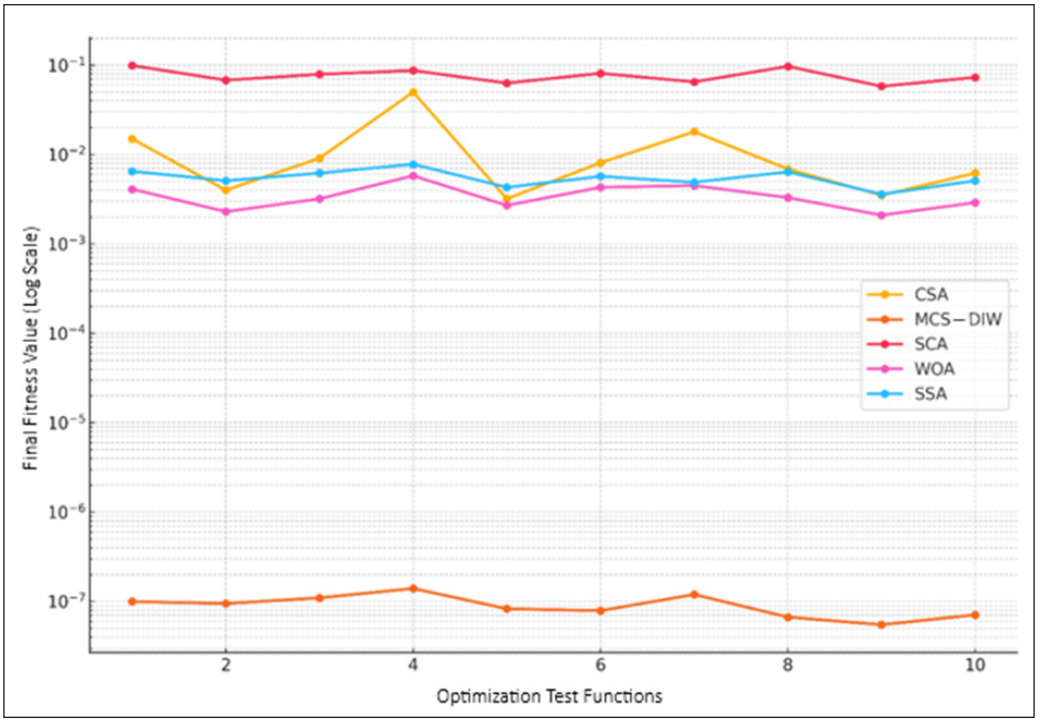


Figure 12. Comparison of the Modified Cuckoo Search using decreasing inertia weight (MCS-DIW) Algorithm with original Cuckoo Search, Sine Cosine Algorithm (SCA), Search Sparrow Algorithm (SSA), and Whale Optimization Algorithm (WOA) using the 10 different optimization test functions
Note. CSA = Cuckoo Search Algorithm

The results shown in Table 7 indicate better performance of the proposed MCS-DIW as the *p*-value ($p < 0.05$) is extremely low compared to other comparative swarm intelligent algorithms, ensuring the effectiveness of the proposed algorithm.

Table 7
Friedman statistical result

Statistical test	**Test value**	***p*-value**
Friedman Test	38.32	9.63×10^{-8}

CONCLUSION AND FUTURE WORKS

This research paper provides an improved variant of the MCS algorithm, proposed using the sigmoid DIW to avoid premature convergence. Extensive investigations are conducted to validate the performance of the MCS-DIW algorithm using optimization test functions. The performance of the MCS-DIW algorithm is compared with a few different SI algorithms, such as original CS, SCA, WOA, and SSA. The simulation experiments show that the

performance, stability, robustness, and convergence speed of the MCS-DIW algorithm surpasses other SI counterparts. In sum, the MCS-DIW algorithm can effectively overcome the local trap problem through its enhanced exploration capacity to find the best global optimum. The proposed research significantly improved the performance of MCS-IW by utilizing fewer tuning parameters and providing the fastest convergence compared to its counterparts. Additionally, MCS-IW's local optima escaping ability ensured strong global search capabilities. However, due to the heavy reliance on the randomized nature of nest and Lévy flight replacements, there exists a problem of inconsistency and population diversity; additional modifications are required to handle the inconsistencies and constrained optimization problems effectively. In this regard, a more in-depth analysis of the proposed MCS-DIW algorithm will be executed in both single-objective (SO) and multi-objective (MO) real-world engineering problems, such as power system optimization, neural network training, image reconstruction, data clustering, data classification, and error minimization. Moreover, the MCS-IW algorithm will be integrated with other swarm intelligence algorithms to hybridize the capabilities to find the maximum global optima. However, the efficiency of the proposed algorithm can be improved by implementing quantum-inspired techniques, parallel computing, and constraint-handling advanced mechanisms to enhance the performance of large-scale optimization tasks for better balancing between exploration and exploitation.

ACKNOWLEDGEMENTS

This research was funded by the Fundamental Research Grant Scheme (FRGS) under a grant number of FRGS/1/2019/ICT02/UNIMAP/02/7.

REFERENCES

Abdul Rani, K. N., Abdulmalek, M., Rahim, H. A., Chin, N. S., & Abd Wahab, A. (2017). Hybridization of strength pareto multiobjective optimization with modified cuckoo search algorithm for rectangular array. *Scientific Reports*, *7*, 46521. https://doi.org/10.1038/srep46521

Abdul Rani, K. N., & Malek, F. (2011). Preliminary study on cuckoo search parameters for symmetric linear array geometry synthesis. In *TENCON 2011-2011 IEEE Region 10 Conference* (pp. 568–572). IEEE. https://doi.org/10.1109/TENCON.2011.6129169

Abualigah, L., Ababneh, A., Ikotun, A. M., Zitar, R. A., Alsoud, A. R., Khodadadi, N., Ezugwu, A. E., Hanandeh, E. S., & Jia, H. (2024). A survey of cuckoo search algorithm: Optimizer and new applications. In L. Abualigah (Ed.), *Metaheuristic optimization algorithms: Optimizers, analysis, and applications* (pp. 45–57). Morgan Kaufmann. https://doi.org/10.1016/B978-0-443-13925-3.00018-2

Adeyelu, A. A., John, Z. S., Uga-Otor, S., Elusakin, O. E., & Godwin, I. R. (2024). An adaptation of Cuckoo Search Algorithm in maximizing energy efficiency of Dynamic Source Routing Algorithm for Mobile AdHoc Network. *International Journal of Computer Applications*, *186*(9), 29-36.

Adeyeye, O. J., & Akanbi, I. (2024). Optimization in systems engineering: A review of how data analytics and optimization algorithms are applied. *Computer Science and IT Research Journal*, *5*(4), 809–823. https://doi.org/10.51594/csitrj.v5i4.1027

Ahmad, T., Sulaiman, M., Bassir, D., Alshammari, F. S., & Laouini, G. (2025). Enhanced numerical solutions for fractional PDEs using Monte Carlo PINNs coupled with cuckoo search optimization. *Fractal and Fractional*, *9*(4), 225. https://doi.org/10.3390/fractalfract9040225

Almufti, S. M., Marqas, R. B., Asaad, R. R., & Shaban, A. A. (2025). Cuckoo search algorithm: Overview, modifications, and applications. *International Journal of Scientific World*, *11*(1), 1-9.

Aziz, R. M. (2022). Cuckoo search-based optimization for cancer classification: A new hybrid approach. *Journal of Computational Biology*, *29*(6), 565–584. https://doi.org/10.1089/cmb.2021.0410

Bharambe, U., Ramesh, R., Mahato, M., & Chaudhari, S. (2024). Synergies Between natural language processing and swarm intelligence optimization: A comprehensive overview. In J. Valadi, K. P. Singh, M. Ojha, & P. Siarry (Eds.), *Advanced machine learning with evolutionary and metaheuristic techniques* (pp. 121–151). Springer. https://doi.org/10.1007/978-981-99-9718-3_6

Brezočnik, L., Fister, I., & Podgorelec, V. (2018). Swarm intelligence algorithms for feature selection: A review. *Applied Sciences*, *8*(9), 1521. https://doi.org/10.3390/app8091521

Chakraborty, A., & Kar, A. K. (2017). Swarm intelligence: A review of algorithms. In S. Patnaik, X. S. Yang, & K. Nakamatsu (Eds.), *Nature-inspired computing and optimization: Theory and applications* (Vol. 10, pp. 475–494). Springer. https://doi.org/10.1007/978-3-319-50920-4_19

Chakraborty, S., Saha, A. K., Ezugwu, A. E., Agushaka, J. O., Zitar, R. A., & Abualigah, L. (2023). Differential evolution and its applications in image processing problems: A comprehensive review. *Archives of Computational Methods in Engineering*, *30*, 985–1040. https://doi.org/10.1007/s11831-022-09825-5

Chen, J., Xia, R., You, J., Yao, Q., Dai, Y., Zhang, J., Yao, J., & Guo, Y. (2024). Automatic optimal design of field plate for silicon on insulator lateral double-diffused metal oxide semiconductor using simulated annealing algorithm. *IET Power Electronics*, *17*(4), 487–493. https://doi.org/10.1049/pel2.12658

Cheraghi, N., Miri, M., & Rashki, M. (2023). An adaptive artificial neural network for reliability analyses of complex engineering systems. *Applied Soft Computing*, *132*, 109866. https://doi.org/10.1016/j.asoc.2022.109866

Choudhary, S., Sugumaran, S., Belazi, A., & Abd El-Latif, A. A. (2023). Linearly decreasing inertia weight PSO and improved weight factor-based clustering algorithm for wireless sensor networks. *Journal of Ambient Intelligence and Humanized Computing*, *14*, 6661-6679. https://doi.org/10.1007/s12652-021-03534-w

Cuong-Le, T., Minh, H.-L., Khatir, S., Wahab, M. A., Tran, M. T., & Mirjalili, S. (2021). A novel version of Cuckoo search algorithm for solving optimization problems. *Expert Systems with Applications*, *186*, 115669. https://doi.org/10.1016/j.eswa.2021.115669

Ghiaskar, A., Amiri, A., & Mirjalili, S. (2024). Polar fox optimization algorithm: A novel meta-heuristic algorithm. *Neural Computing and Applications*, *36*, 20983–21022. https://doi.org/10.1007/s00521-024-10346-4

Habeb, A. A. A. A., Taresh, M. M., Li, J., Gao, Z., & Zhu, N. (2024). Enhancing medical image classification with an advanced feature selection algorithm: A novel approach to improving the cuckoo search algorithm by incorporating Caputo fractional order. *Diagnostics*, *14*(11), 1191. https://doi.org/10.3390/diagnostics14111191

Huang, S., & Zhou, J. (2024). An enhanced stability evaluation system for entry-type excavations: Utilizing a hybrid bagging-SVM model, GP and kriging techniques. *Journal of Rock Mechanics and Geotechnical Engineering*, *17*(4), 2360-2373. https://doi.org/10.1016/j.jrmge.2024.05.024

Joshi, A. S., Kulkarni, O., Kakandikar, G. M., & Nandedkar, V. M. (2017). Cuckoo search optimization - A review. *Materials Today: Proceedings*, *4*(8), 7262–7269. https://doi.org/10.1016/j.matpr.2017.07.055

Kwakye, B. D., Li, Y., Mohamed, H. H., Baidoo, E., & Asenso, T. Q. (2024). Particle guided metaheuristic algorithm for global optimization and feature selection problems. *Expert Systems with Applications*, *248*, 123362. https://doi.org/10.1016/j.eswa.2024.123362

Liu, C., Wang, J., Zhou, L., & Rezaeipanah, A. (2022). Solving the multi-objective problem of IoT service placement in fog computing using cuckoo search algorithm. *Neural Processing Letters*, *54*, 1823–1854. https://doi.org/10.1007/s11063-021-10708-2

Luo, X., Chen, J., Yuan, Y., & Wang, Z. (2024). Pseudo gradient-adjusted particle swarm optimization for accurate adaptive latent factor analysis. In *IEEE Transactions on Systems, Man, and Cybernetics: Systems* (Vol. 54, No. 4, pp. 2213-2226). IEEE. https://doi.org/10.1109/TSMC.2023.3340919

Mahmood, S., Bawany, N. Z., & Tanweer, M. R. (2023). A comprehensive survey of whale optimization algorithm: Modifications and classification. *Indonesian Journal of Electrical Engineering and Computer Science*, *29*(2), 899-910. http://doi.org/10.11591/ijeecs.v29.i2.pp899-910

Mareli, M., & Twala, B. (2018). An adaptive Cuckoo search algorithm for optimisation. *Applied Computing and Informatics*, *14*(2), 107–115. https://doi.org/10.1016/j.aci.2017.09.001

Massat, M. B. (2018). A promising future for AI in breast cancer screening. *Applied Radiology*, *47*(9), 22–25.

Meena, K. S., Singh, S. S., & Singh, K. (2024). Cuckoo search optimization-based influence maximization in dynamic social networks. *ACM Transactions on the Web*, *18*(4), 1–25. https://doi.org/10.1145/3690644

Mohammed, B. A., Zhuk, O., Vozniak, R., Borysov, I., Petrozhalko, V., Davydov, I., Borysov, O., Yefymenko, O., Protas, N., & Kashkevich, S. (2023). Improvement of the solution search method based on the cuckoo algorithm. *Eastern-European Journal of Enterprise Technologies*, *2*(4 (122)), 23-30. https://doi.org/10.15587/1729-4061.2023.277608

Nickabadi, A., Ebadzadeh, M. M., & Safabakhsh, R. (2011). A novel particle swarm optimization algorithm with adaptive inertia weight. *Applied Soft Computing*, *11*(4), 3658–3670. https://doi.org/10.1016/j.asoc.2011.01.037

Saka, M. P., Doğan, E., & Aydogdu, I. (2013). Analysis of swarm intelligence–based algorithms for constrained optimization. In X.-S. Yang, R. Xiao, & M. Karamanoglu (Eds.), *Swarm intelligence and bio-inspired computation: Theory and applications* (pp. 25–48). Elsevier. https://doi.org/10.1016/B978-0-12-405163-8.00002-8

Sekyere, Y. O. M., Effah, F. B., & Okyere, P. Y. (2024). An enhanced particle swarm optimization algorithm via adaptive dynamic inertia weight and acceleration coefficients. *Journal of Electronics and Electrical Engineering*, *3*, 53–67. https://doi.org/10.37256/jeee.3120243868

Shi, Y., & Eberhart, R. C. (1998). Parameter selection in particle swarm optimization. In V. W. Porto, N. Saravanan, D. Waagen, & A. E. Eiben (Eds.), *Evolutionary Programming VII: 7th International Conference* (Vol. 1447, pp. 591–600). Springer. https://doi.org/10.1007/BFb0040810

Sohail, A. (2023). Genetic algorithms in the fields of artificial intelligence and data sciences. *Annals of Data Science*, *10*, 1007–1018. https://doi.org/10.1007/s40745-021-00354-9

Thaher, T., Sheta, A., Awad, M., & Aldasht, M. (2024). Enhanced variants of crow search algorithm boosted with cooperative based island model for global optimization. *Expert Systems with Applications*, *238*(Part A), 121712. https://doi.org/10.1016/j.eswa.2023.121712

Tian, Y., Zhang, D., Zhang, H., Zhu, J., & Yue, X. (2024). An improved cuckoo search algorithm for global optimization. *Cluster Computing*, *27*, 8595-8619. https://doi.org/10.1007/s10586-024-04410-w

Umar, S. U., Rashid, T. A., Ahmed, A. M., Hassan, B. A., & Baker, M. R. (2024). Modified Bat Algorithm: A newly proposed approach for solving complex and real-world problems. *Soft Computing*, *28*, 7983–7998. https://doi.org/10.1007/s00500-024-09761-5

Wei, J., & Niu, H. (2022). A ranking-based adaptive cuckoo search algorithm for unconstrained optimization. *Expert Systems with Applications*, *204*, 117428. https://doi.org/10.1016/j.eswa.2022.117428

Xue, X., Shanmugam, R., Palanisamy, S., Khalaf, O. I., Selvaraj, D., & Abdulsahib, G. M. (2023). A hybrid cross layer with Harris-hawk-optimization-based efficient routing for wireless sensor networks. *Symmetry*, *15*(2), 438. https://doi.org/10.3390/sym15020438

Yang, Q., Wang, Y., Zhang, J., & Gao, H. (2024). An adaptive operator selection cuckoo search for parameter extraction of photovoltaic models. *Applied Soft Computing*, *166*, 112221. https://doi.org/10.1016/j.asoc.2024.112221

Zangana, H. M., Sallow, Z. B., Alkawaz, M. H., & Omar, M. (2024). Unveiling the collective wisdom: A review of swarm intelligence in problem solving and optimization. *Inform: Jurnal Ilmiah Bidang Teknologi Informasi Dan Komunikasi*, *9*(2), 101–110. https://doi.org/10.25139/inform.v9i2.7934

Zdiri, S., Chrouta, J., & Zaafouri, A. (2021). An expanded heterogeneous particle swarm optimization based on adaptive inertia weight. *Mathematical Problems in Engineering*, *2021*(1), 4194263. https://doi.org/10.1155/2021/4194263

Zheng, W., Si, M., Sui, X., Chu, S., & Pan, J. (2023). Application of a parallel adaptive Cuckoo Search algorithm in the rectangle layout problem. *CMES-Computer Modeling in Engineering and Sciences*, *135*(3), 2173-2196. https://doi.org/10.32604/cmes.2023.019890

Zheng, Y.-L., Ma, L.-H., Zhang, L.-Y., & Qian, J.-X. (2003). Empirical study of particle swarm optimizer with an increasing inertia weight. In *The 2003 Congress on Evolutionary Computation* (pp. 221–226). IEEE. https://doi.org/10.1109/CEC.2003.1299578