Islamic University Journal of Applied Sciences (IUJAS)
https://journals.iu.edu.sa/jesc

Volume VIII, Issue I, July 2026, Pages 230-248

OCSA: An Optimized Cuckoo Search Algorithm with Guided Exploration and Refined Local Search

Riad Bouakacha[1], Sofiane Haddad[2,*], Ammar Boulaiche[3] and Abdelhamid Rabhi[4]

[1] Electrical Department, MSB Jijel University, BP 98, Ouled Aissa, Jijel-1800 Algeria
[2] REL Laboratory, Electronics Department, MSB Jijel University, Algeria
[3] LaRIA Laboratory, Faculty of Exact Sciences, MSB Jijel University, Jijel 18000, Algeria.
[4] MIS Laboratory, Université de Picardie Jules Verne, Amiens, Picardie, France

*Corresponding author: (Sofiane Haddad), *Email Address*: s_haddad@univ-jijel.dz

Abstract

This paper introduces an optimized version of the Cuckoo Search Algorithm (CSA), termed the Optimized CSA (OCSA), designed to address key limitations of traditional population-based optimization methods, including slow convergence, limited precision, and high memory usage. OCSA improves both exploration and exploitation through two main contributions. First, it employs a multi-directional guided displacement strategy around the current best solution, enhancing the algorithm's ability to explore promising regions more effectively. Second, it strengthens local search by having each cuckoo generate multiple candidate solutions and selecting the best among them, which improves solution accuracy and accelerates convergence. The performance of OCSA is evaluated using benchmark functions, engineering design problems, and a real-world application in economic dispatch. Comparative evaluations against established algorithms, including Cooperative Coevolutionary CS (CCCS), Particle Swarm Optimization (PSO), and Genetic Algorithm (GA), as well as mathematical solvers such as Interior Point OPTimizer (IPOPT) and Linear and Objective Quadratic (based on an infeasible primal-dual interior-point) Optimizer (LOQO), demonstrate that OCSA achieves superior convergence speed, solution quality, and scalability. These results establish OCSA as an efficient optimization approach for complex problems.

Keywords: Optimized Cuckoo Search Algorithm; Metaheuristic Optimization; Exploration; Exploitation; Enhanced Local Search.

https://doi.org/10.63070/jesc.2026.025
Received 09 April 2026; Revised 18 April 2026; Accepted 25 April 2026.
Available online 30 April 2026.
Published by Islamic University of Madinah on behalf of *Islamic University Journal of Applied Sciences.*
This is an open access article published under the Creative Commons Attribution 4.0 International license (CC BY 4.0):https://creativecommons.org/licenses/by/4.0/

List of abbreviations

Abbreviation	**Full Form**
ABC	Artificial Bee Colony
AIS	Artificial Immune System
AIS-GA	Artificial Immune System combined with Genetic Algorithm
ALPSO	Augmented Lagrangian Particle Swarm Optimization
BBA	Binary Bat Algorithm
BFO	Bacterial Foraging Optimization
CCCS	Cooperative Coevolutionary Cuckoo Search
CCGA	Cooperative Coevolutionary Genetic Algorithm
Charibde	Cooperative Hybrid Algorithm using Reliable Interval-Based Methods and Differential Evolution
CHc	Constraint Handling (coevolutionary)
CHp	Constraint Handling (penalty)
COPs	Constrained Optimization Problems
CPSO	Cooperative Particle Swarm Optimization
CPSO-SK	Cooperative Particle Swarm Optimization – Single K (collaborative model)
CS / CSA	Cuckoo Search (Algorithm)
DE	Differential Evolution
EA	Evolutionary Algorithm
EMS	Evolutionary Multi-agent System
FA	Firefly Algorithm
GA	Genetic Algorithm
GDACS	Grouped Dynamic Adaptive Cuckoo Search
HAGCS	Hierarchical Adaptive Cuckoo Search
HS	Harmony Search
HS-SQP	Harmony Search combined with Sequential Quadratic Programming
IPOPT	Interior Point Optimizer (primal-dual interior-point algorithm)
LOQO	Optimization solver based on an infeasible primal-dual interior-point method
MECA	M-elite Coevolutionary Algorithm
MINOS	Modular In-core Nonlinear Optimization System
MOCSA	Multi-Objective Cuckoo Search Algorithm
OCSA	Optimized Cuckoo Search Algorithm
OEA	Organizational Evolutionary Algorithm
PSO	Particle Swarm Optimization
PSOA	Particle Swarm Optimization Algorithm
PSOAN	Particle Swarm Optimization with Adaptive Neighborhood
PSOSTR	Particle Swarm Optimization with Self-adaptive Technique for Mixed-variable Response
RA	Random/Restart Approach *(context-dependent)*
SA	Simulated Annealing
SA-DS	Simulated Annealing with Downhill Simplex
SBM	Socio-Behavioral Model
SCA	Sine Cosine Algorithm
SMES	Simple Multi-member Evolution Strategy
SQP	Sequential Quadratic Programming
UCOPs	Unconstrained Optimization Problems
YCSA	Yang's Cuckoo Search Algorithm *(CSA attributed to Xin-She Yang)*

1. Introduction

The Cuckoo Search Algorithm (CSA), proposed by Yang and Deb, is a nature-inspired metaheuristic that models the brood parasitism behavior of certain cuckoo species. These species select high-quality host nests to maximize the survival probability of their eggs, a strategy that CSA emulates to efficiently explore the search space and identify optimal solutions.

CSA has emerged as an effective global optimization technique, integrating fundamental concepts from algorithms such as Particle Swarm Optimization (PSO), Differential Evolution (DE), and Simulated Annealing (SA). Its simplicity and ease of implementation, combined with minimal parameter tuning requirements—typically limited to population size and discovery probability—contribute to its robustness across a wide range of applications [1–2].

The algorithm has been successfully applied in numerous domains, including engineering design optimization [3], economic load dispatch [4], unit commitment [5], multi-objective power scheduling [6], optimal power flow [7], manufacturing optimization [8], financial modeling [9], image processing [10], and aerodynamic design [11]. Comparative studies [12] have demonstrated its superior performance in terms of convergence speed, robustness, and solution accuracy relative to several other metaheuristic approaches.

Nevertheless, similar to many population-based optimization methods, CSA encounters limitations when addressing large-scale and high-dimensional problems, primarily due to increased computational cost and slower convergence rates [2], [9]. To mitigate these issues, this study proposes an enhanced variant of CSA that reduces the population size while improving convergence efficiency without compromising solution quality.

Numerous modifications of CSA have been reported in the literature to improve its performance. Lévy flight-based mechanisms were introduced in [4] to strengthen exploration capabilities, while improvements in exploitation have been achieved through guided selection and evaluation strategies [2], [9]. The authors in [13] proposed a hierarchical adaptive cuckoo search algorithm (HAGCS) aimed at enhancing global search capability and adaptability across diverse optimization scenarios. Their approach incorporates distributional entropy-based analysis, weighted fitness–distance balanced ranking, an adaptive global search strategy, and a hierarchical nest replacement scheme to improve convergence and avoid local optima. Adaptive and hybrid variants, including self-adaptive techniques [14], co-evolutionary models [15], and dynamic parameter tuning strategies [16], have also demonstrated improved performance. Additionally, parameter sensitivity analyses have been investigated in [17].

Further advancements include the Grouped Dynamic Adaptive Cuckoo Search (GDACS) introduced in [18], which enhances population diversity during initialization using chaotic maps and replaces the standard update mechanism with Cauchy and Lévy-based operators. The method further employs dynamic population grouping based on fitness levels and applies adaptive step-size mechanisms to each group, thereby achieving a better balance between exploration and exploitation.

In addition, [19] presented a comprehensive survey of multi-objective CSA (MOCSA) variants and their applications. The study reviewed 123 distinct variants reported in the literature, analyzing their theoretical developments, methodological adaptations, and applications across various domains. It further categorizes these variants into original, modified, and hybrid approaches, evaluates their performance characteristics, and outlines potential directions for future research.

Despite these advancements, the standard CSA remains heavily reliant on stochastic processes, which tend to favor exploration at the expense of exploitation. To address this imbalance, the proposed method introduces enhancements to both phases. Specifically, a guided exploration mechanism is employed to direct the search toward promising regions of the solution space, while an improved exploitation strategy generates multiple candidate solutions around elite individuals and selects the most promising ones.

These improvements result in the Optimized Cuckoo Search Algorithm (OCSA), which achieves a more effective balance between exploration and exploitation while preserving the inherent strengths of the original CSA.

The remainder of this paper is organized as follows. Section II reviews the standard CSA. Section III presents the proposed enhancements. Section IV discusses the simulation results obtained from benchmark and real-world problems. Finally, Section V concludes the paper and outlines directions for future work.

2 Cuckoo search algorithm

The CSA, proposed by Yang and Deb in 2009, is a metaheuristic optimization method inspired by the brood parasitism of certain cuckoo species, which lay their eggs in the nests of other birds instead of raising their own offspring.

2.1 Behaviour of Cuckoo Bird

Certain cuckoo species, such as Ani and Guira, lay their eggs in the nests of other birds. If detected, the host may remove the eggs or abandon the nest. To increase survival, some cuckoos mimic the appearance of host eggs and time their laying so their chicks hatch first and dominate the nest [20].

Based on this behavior, the CSA follows three main rules [3]: each cuckoo lays one egg in a randomly chosen nest, the best nests are kept for the next generation, and a fraction of nests is replaced when parasitic eggs are discovered with probability $\boldsymbol{P_a}$. In practice, this is modeled by replacing a portion $\boldsymbol{P_a}$ of the population with new random solutions.

CSA combines local and global random search, controlled by the parameter $\boldsymbol{P_a}$, to balance exploration and exploitation [1].

$$x_i^{t+1} = \boldsymbol{x_i^t} + \alpha\, \boldsymbol{\rho} \otimes H(P_a, \varepsilon) \otimes \left(\boldsymbol{x_j^t} - \boldsymbol{x_k^t}\right) . \quad (1)$$

where x^t_j and x^t_k are two randomly selected solutions, $H(u)$ is the Heaviside function, ε is a uniformly distributed random number, and ρ is the step size. The global search is performed using Lévy flights [1].

$$\boldsymbol{x_i^{t+1}} = \boldsymbol{x_i^t} + \alpha L(s) . \quad (2)$$

Here, α > 0 is the step size scaling factor, usually set relative to the problem's characteristic scale LLL. Typically, α = O(L/10), though smaller values like α = O(L/100) can be more effective to avoid overly large steps [1].

To generate the Lévy flight random walk from a nest, we:

- ✓ Select a random direction sampled from a uniform distribution
- ✓ Generate step sizes using the Mantegna algorithm, which yields:

$$\boldsymbol{s} = \boldsymbol{u} / \left(\boldsymbol{v}^{\frac{1}{\beta}}\right) . \quad (3)$$

While $1 \leq \beta \leq 2$ and $|s| \geq |s_0| >> 0$, and u and v are drawn from normal distribution:

$$u \sim N(0, \sigma_u^2); v \sim N(0, \sigma_v^2)$$

Where

$$\sigma_u = \left[\frac{\Gamma(1+\beta) sin\left(\frac{\pi\beta}{2}\right)}{\Gamma\left(\frac{1+\beta}{2}\right)\beta 2^{\frac{\beta-1}{2}}}\right]^{1/\beta} ;\ \sigma_v = 1 . \quad (4)$$

2.2 Pseudo Code of the Cuckoo Search Algorithm

Based on the aforementioned rules, the CSA pseudo-code is thoroughly detailed in [4] and can be summarized as follows (Algorithm 1):

Algorithm 1- Pseudo Code of the Cuckoo Search Algorithm.

```
BEGIN

// 1. Objective Definition
DEFINE objective function:
   Minimize f(x), where x ∈ ℝ^d

// 2. Initialization
GENERATE n host nests x_i (i = 1, 2, ..., n)
SET discovery probability P_a ∈ [0, 1]
SET maximum generations MaxGen
SET t = 0

// 3. Initial Fitness Evaluation
EVALUATE fitness F_i = f(x_i) for all nests

// 4. Main Loop
WHILE (t < MaxGen) DO

   // Cuckoo search phase
   FOR each cuckoo i DO
      // a) Lévy flight
      x_i_new = x_i + α ⊕ Lévy(λ)
      // b) Evaluate new solution
      F_i_new = f(x_i_new)

      // c) Random nest selection
      Randomly choose nest j
      // d) Greedy replacement (minimization)
      IF (F_i_new < F_j) THEN
         REPLACE x_j with x_i_new
      END IF
   END FOR

   // Abandonment phase
   ABANDON a fraction P_a of worst nests
   GENERATE new random solutions for abandoned nests

   // Re-evaluation
   EVALUATE fitness of new solutions

   // Update best solution
   RANK all nests by fitness
   IDENTIFY current best solution

   // Increment generation counter
   t = t + 1

END WHILE

// 5. Output
OUTPUT best solution found

END
```

3 Optimized CSA

3.1. Implementation of Enhanced Exploration in CSA

The original CSA, as implemented by Yang, relies on stochastic processes during both the exploration and exploitation phases. This stochastic nature helps prevent the algorithm from becoming trapped in local optima, particularly through the use of Equation (2), which generates new solutions based solely on the current population. However, while effective at maintaining diversity, this strategy can sometimes slow down convergence.

To address this limitation, our modified algorithm retains the core stochastic mechanisms but introduces additional features designed to accelerate convergence while maintaining global search

capability. Drawing inspiration from natural cuckoo behavior, where cuckoos engage in extensive exploratory movements before selecting a secure, resource-rich nesting site, our approach incorporates multiple exploratory phases.

The process begins with an initial exploration using Lévy flights (Figures 1 and 2) to identify promising regions in the search space. This is followed by random exploratory displacements originating from the initial points to further sample nearby regions. Finally, the algorithm performs directed movements toward the current best nest, which include both lateral displacements (left or right of the starting point) and gradient-inspired movements (upstream or downstream relative to the best nest location).

By combining stochastic and quasi-random exploration with guided exploitation, this enhanced strategy improves both the efficiency and reliability of the search process. The subsequent selection process for determining the optimal nesting sites proceeds as follows (Algorithm 2):

Algorithm 2- Optimized Cuckoo Nesting Algorithm with Lévy Flights and Asymmetric Local Moves.

BEGIN

FOR i = 1 TO number_of_cuckoo_nests DO
$x_i^{(1)}$ = Lévy_flight_move(x_i^0)
$x_i^{(2)}$ = downstream_move ($x_i^{(1)}$, best_nest)
$x_i^{(3)}$ = upstream_move($x_i^{(1)}$, best_nest)
$x_i^{(4)}$ = right_move($x_i^{(1)}$)
$x_i^{(5)}$ = left_move($x_i^{(1)}$)

x_i = BEST($x_i^{(1)}$, $x_i^{(2)}$, $x_i^{(3)}$, $x_i^{(4)}$, $x_i^{(5)}$)
END FOR
END

3.2. Implementation of Enhanced Exploitation Phase in CSA

In contrast to the standard CSA assumption described in [5], where each cuckoo lays only one egg, we introduce an intensified exploitation mechanism inspired by biological observations. In nature, certain parasitic cuckoo species lay multiple eggs in host nests, with only the best-adapted offspring surviving. We incorporate this behavior into our algorithm to enhance its exploitation capability.

This modification introduces several key features: a multi-egg deposition strategy, a competitive selection process, and an adaptive survival mechanism. Together, these elements allow the algorithm to generate multiple candidate solutions within each iteration, from which only the most promising are retained. The resulting enhanced exploitation phase operates as follows (Algorithm 3).

Algorithm 3. Competitive Multi-Egg Cuckoo Search Algorithm with Adaptive Survival Strategy.

BEGIN
 	
FOR k = 1 TO number_of_eggs_per_cuckoo DO
 	
// Abandon egg with probability Pa
IF random() ≥ Pa THEN
 	
// Randomly select another nest j
j = random_integer(1, Nnest)
 	
// Generate new solution via differential mutation
	xi'(k) = xi(k) + random() * (xj - xi)
	
	ELSE
	// Keep original egg (implicitly xi'(k) = xi(k))
	END IF
	
	END FOR
	
	// Keep best egg (for minimization)
	xi = ARGMIN{ f(xi'(k)) } // or BEST OF {xi'}
	
	END

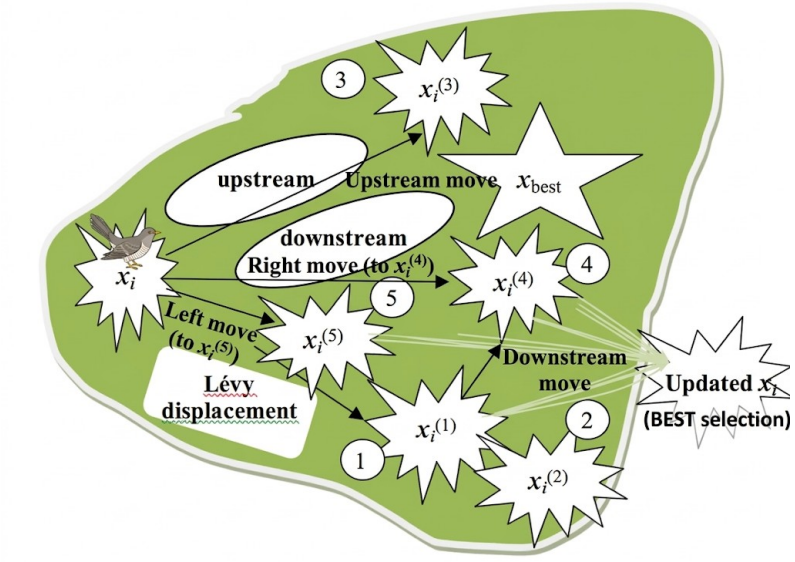


Fig. 1 Exploration the area by the cuckoo bird

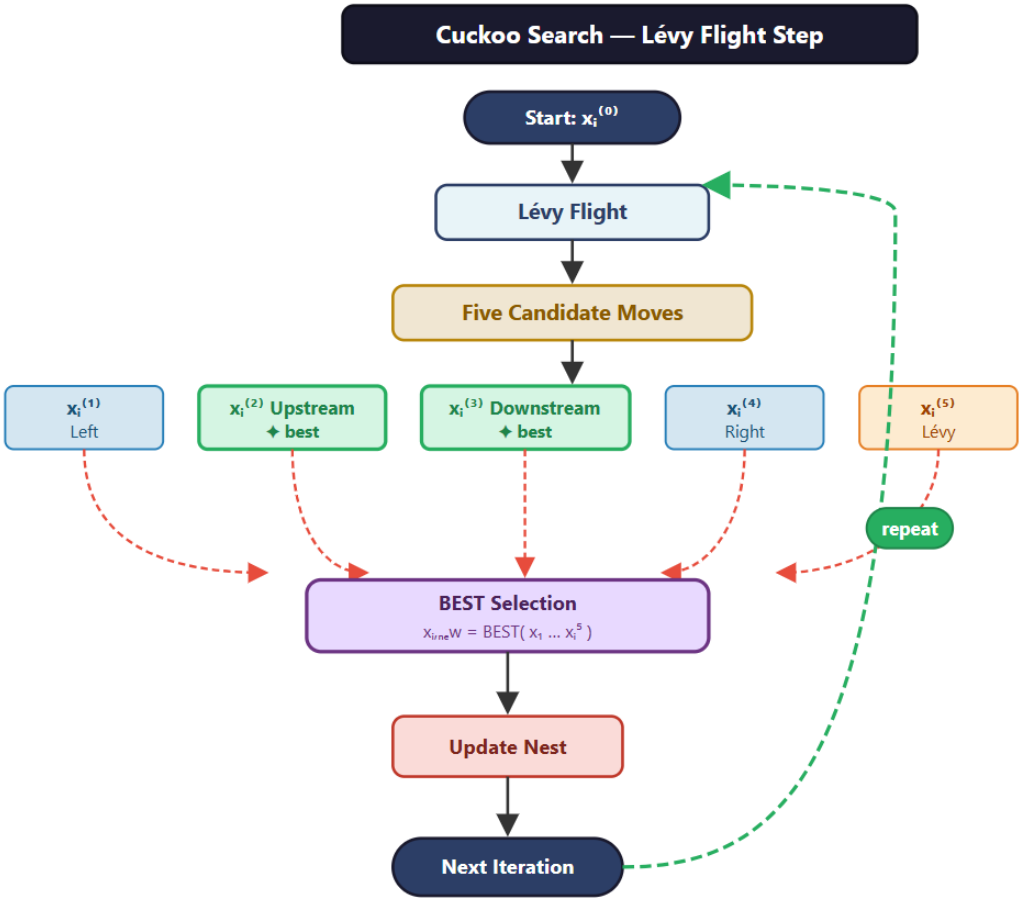


Fig. 2 Flowchart- OCSA Lévy Flight Step

4. Simulation Methodology

In our simulation study, we maintained consistent problem formulations while replicating most of the experimental applications reported by Zheng *et al.* [15]. Our validation approach consisted of two main stages to thoroughly assess the performance of the proposed algorithm.

First, we conducted comprehensive benchmark testing that included both constrained and unconstrained optimization problems drawn from references [22–24]. In addition, we evaluated the algorithm on a set of challenging global optimization problems reported in [25], which further tested its robustness and versatility.

Second, we carried out specific performance evaluations focusing on test function F9, as illustrated in Figure 4. These tests included augmenting the dimensions to large scales to examine the algorithm's scalability and computational efficiency under extreme problem sizes.

4.1 Results on Unconstrained Optimization Problems

In this study, we evaluated the performance of our proposed algorithm using 15 benchmark functions (F1–F15) under consistent test conditions [23]. For functions F1 to F13, the dimensionality was set to 30, while for F14 and F15, it was increased to 100. Each test involved 2,500 function evaluations and was repeated 50 times to ensure statistical significance.

The comparative results between the CCCS and OCSA algorithms, as summarized in Table 1, demonstrate the superior convergence behavior of OCSA toward the exact global optima. Notably, OCSA consistently attained optimal values for both the best and average solutions across most functions. In contrast, CCCS exhibited substantially inferior results in terms of solution quality metrics, indicating a significant performance gap.

Figures 3 and 4 further illustrate this advantage. For example, on function F8, OCSA reached convergence within 500 evaluations, whereas CCCS required approximately 1,000 evaluations. Similarly, for the high-dimensional function F15, OCSA converged before 500 evaluations, while CCCS needed nearly 1,500 evaluations.

Additionally, a comparative analysis involving other algorithms such as MECA and OEA (Table 2) confirms that OCSA maintains superior average solution quality. Across the majority of the test functions, OCSA consistently outperforms these methods, reinforcing its robustness and efficiency in global optimization.

Table 1: Comparison between CCCS and OCSA over 50 trials [15]

Function	fmin	CCCS Best Value	CCCS Mean Value	OCSA Best Value	OCSA Mean Value
F1	0	8.804e-242	2.011e-217	0	0
F2	0	4.271e-143	1.628e-130	0	0
F3	0	6.280e-192	5.464e-154	0	0
F4	0	1.085e-121	1.333e-098	0	0
F5	0	25.0366	25.4182	2.521e-29	1.525e-25
F6	0	0	0	0	0
F7	0	5.309e-05	3.080e-04	0	0
F8	-12569.5	-1.25694870e+04	-1.25694867e+04	-1.25694866e+04	-1.25694866e+04
F9	0	0	0	0	0
F10	0	1.655e-07	3.663e-05	-8.882e-16	-8.882e-16
F11	0	0	0	0	0
F12	0	6.845e-11	9.518e-07	1.571e-32	1.571e-32
F13	0	2.073e-13	1.723e-07	1.350e-32	1.350e-32
F14	-99.60	-96.6008	-83.7328	-96.4322	-94.3259
F15	-78.33236	-78.33233	-78.33232	-78.3323	-78.3323

Table 2: Comparison between MECA, OEA, CCCS, YCSA and OCSA over 50 trials for mean function values [15]

Function	**fmin**	**CCCS**	**MECA**	**OEA**	**OCSA**
F1	0	2.011e-217	4.228e-183	2.481e-30	0
F2	0	1.628e-130	1.845e-110	2.068e-13	0
F3	0	5.464e-154	3.274e-95	1.883e-09	0
F4	0	1.333e-098	5.124e-02	8.821e-02	0
F5	0	25.4182	7.973e-02	0.227	1.525e-25
F6	0	0	0	0	0
F7	0	3.080e-04	4.084e-04	3.297e-03	0
F8	-12569.5	-1.25694867e+04	-1.25694866e+04	-1.25694866e+04	-1.25694866e+04
F9	0	0	0	5.430e-17	0
F10	0	3.663e-05	0	5.336e-14	-8.882e-16
F11	0	0	3.844e-03	1.317e-02	0
F12	0	9.518e-07	1.571e-32	9.207e-30	1.571e-32
F13	0	1.723e-07	1.350e-32	4.323e-19	1.350e-32
F14	-99.60	-83.7328	-98.7095	-99.5024	-94.32596
F15	-78.33236	-78.33232	-78.33233	-78.33233	-78.33232

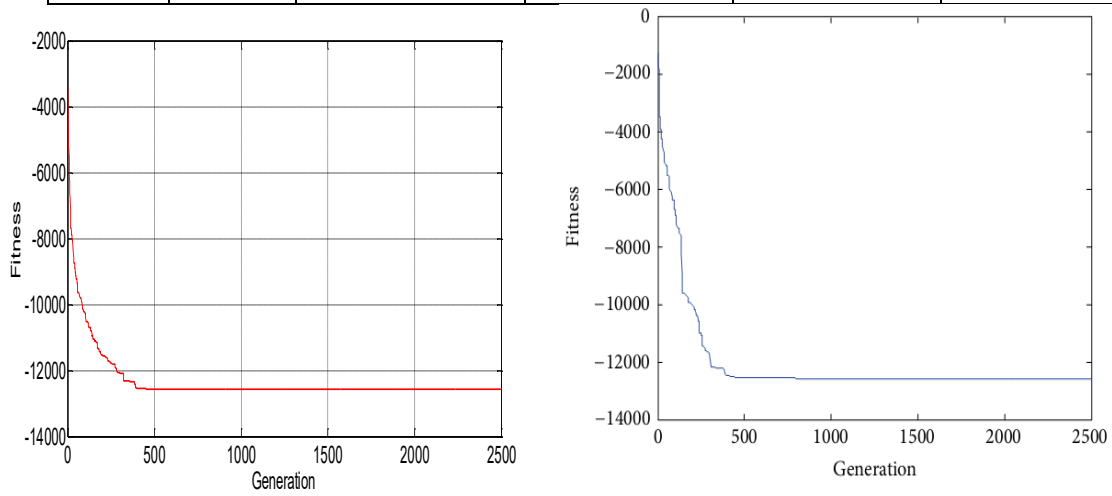


Figure 3: Convergence curves of our application (figure on the left) and of CCCS of F8

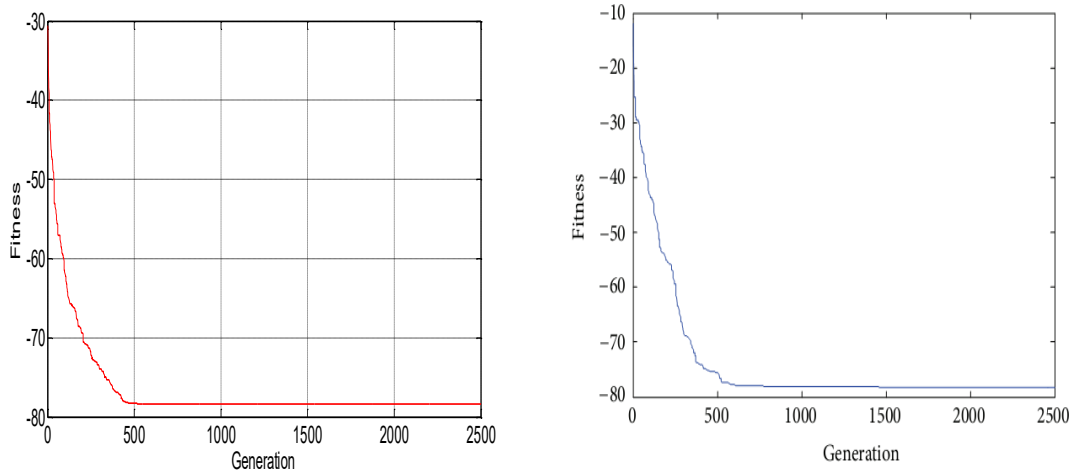


Figure 4: Convergence curves of our application (figure on the left) and of CCCS of F15

4.2 Results on Constrained Optimization Problem

This section evaluates the performance of the proposed algorithm on a diverse set of constrained optimization challenges. The test suite includes nine benchmark problems (G01 to G11), excluding G02 and G03 due to ambiguous or inconsistent results reported in the literature. Additionally, two engineering design problems from [15] are considered to assess practical applicability. Finally, an economic load dispatch problem that incorporates valve-point effects is included to demonstrate the algorithm's capability in handling complex real-world scenarios.

4.2.1 Benchmark Function Analysis

In accordance with the CCCS methodology, we conducted 2,500 function evaluations per test. It is worth noting that the comparative algorithm OEA utilized 24,000 evaluations per test. Table 3 summarizes the statistical results obtained from 50 trials for each function, although our implementation employed fewer trials in certain cases.

The key findings indicate that our algorithm consistently reached the global optimum for the majority of the tested functions. Furthermore, it demonstrated superior performance when compared to several benchmark methods, including CCCS, OEA CHp, OEA CHc, and EMS. Overall, these results clearly highlight the competitive advantage of our method in solving constrained optimization problems efficiently and reliably.

Table 3: Comparison between OEA, SMES, CCCS best function values and OCSA function values [15]

Function	fmin	CCCS	OEA CHp	OEA CHc	SMES	OCSA
G01	15	15	15	15	15	15
G04	-30665.539	-3.066554e+04	-30665.539	-30665.539	-30665.539	-30665.52411
G05	5126.498	5.126498e+03	5126.497	5126.532	5126.599	5126.498077
G06	-6961.814	-6.961814e+03	-6961.814	-6961.814	-6961.814	-6961.813875
G07	24.306	24.30621	24.308	24.307	24.327	24.306206
G08	-0.095825	-9.5825e-02	-0.095825	-0.095825	-0.095825	-0.105460
G09	680.630	680.630	680.630	680.630	680.632	680.630057
G10	7049.331	7.0492e+03	7052.236	7100.030	7051.903	7049.239
G11	0.750	0.7500	0.750	0.750	0.750	0.749950

4.2.2 Welded Beam Design Case Study

This section provides a comparative analysis of different optimization methods reported in the literature for the welded beam design problem. The comparison focuses on two main aspects: performance metrics across various evaluation budgets and the identification of solutions that violate problem constraints, which are marked as unreliable.

As shown in Table 4, our proposed method achieves the best-known optimal solution for the problem. Additionally, it demonstrates superior convergence efficiency by requiring the fewest iterations among all feasible solutions. These results highlight the effectiveness and reliability of our approach compared to existing methods.

Table 4: Welded beam problem: comparison of OCSA results with the literature [14].

Researcher / Study	Method	m	H	l	t	b	Cost / Objective	No. of Eval.	Constraint Violation
Coello	GA	0.2088	3.4205	8.9975	0.2100	1.7483	N.A.	N.A.	Yes
Leite & Topping	GA	0.2489	6.1097	8.2484	0.2485	2.4000	6273	No	No
Deb	GA	0.2489	6.1730	8.1789	0.2533	2.4331	320,080	No	No
Lemonge & Barbosa	GA	0.2443	6.2117	8.3015	0.2443	2.3816	320,000	No	No
Bernardino et al.	AISa-GA	0.2444	6.2183	8.2912	0.2444	2.3812	320,000	No	No
Atiqullah & Rao	SAb	0.2471	6.1451	8.2721	0.2495	2.4148	N.A.	No	No
Hedar & Fukushima	SA	0.2444	6.2175	8.2915	0.2444	2.3810	N.A.	No	No
Liu	SA-GA	0.2231	1.5815	12.8468	0.2245	2.2500	26,466	Yes	Yes
Hwang & He	SA-DSc	0.2444	6.2158	8.2939	0.2444	2.3811	56,243	No	No
Parsopoulos & Vrahatis	PSO	N.A.	N.A.	N.A.	N.A.	1.9220	100,000	-	-
He et al.	PSO	0.2444	6.2175	8.2915	0.2444	2.3810	30,000	No	No
Zhang et al.	EAd	0.2443	6.2201	8.2940	0.2444	2.3816	28,897	No	No
Coello	EA	N.A.	N.A.	N.A.	N.A.	1.8245	N.A.	??	??
Lee & Geem	HSe	0.2442	6.2231	8.2915	0.2443	2.3810	110,000	Yes	Yes
Mahdavi et al.	HS	0.2057	3.4705	9.0366	0.2057	1.7248	200,000	Yes	Yes
Fesanghary et al.	HS-SQP	0.2057	3.4706	9.0368	0.2057	1.7248	90,000	Yes	Yes
Siddall	RAg	0.2444	6.2819	8.2915	0.2444	2.3815	N.A.	No	No
Akhtar et al.	SBMh	0.2407	6.4851	8.2399	0.2497	2.4426	19,259	No	No
Ray & Liew	SCAi	0.2444	6.2380	8.2886	0.2446	2.3854	33,095	No	No
Montes & Oca	BFOj	0.2536	7.1410	7.1044	0.2536	2.3398	N.A.	Yes	Yes
Zhang et al.	DEk	0.2444	6.2175	8.2915	0.2444	2.3810	24,000	No	No
Gandomi et al.	FA	0.2015	3.5620	9.0414	0.2057	1.7312	50,000	Yes	Yes
Hongqing Zheng	CCCS	0.3312	10.000	2.4095	0.3456	2.1730	25,000	Yes	Yes
Present study	OCSA	0.20573	7.0924	9.03663	0.20573	2.21815	35 (0.9 sec)	No	No

4.2.3 Case of pressure design

Table 5, which presents the results obtained for the pressure design problem, also demonstrates the superiority of our algorithm. It achieves the best optimal solution with fewer function evaluations compared to various solutions reported in the literature.

Table 5: Pressure vessel design: comparison of OCSA results with the literature [15].

Method	R	L	Ts	Th	Cost	Iter. Max / Time
CCCS	40.3196	200.0000	0.7782	0.3846	5885.3	1000
ALPSO	41.3500	200.0000	0.7980	0.3950	6234.0	7590
PSOA	N.A.	N.A.	N.A.	N.A.	6292.0	6506
PSOSTR	N.A.	N.A.	N.A.	N.A.	6272.0	3723
He et al.	N.A.	N.A.	N.A.	N.A.	6290.0	30000
Akhtar et al.	N.A.	N.A.	N.A.	N.A.	6335.0	12630
OCSA	40.31962	200.0000	0.77817	0.38304	5880.67	55 (0.7 sec)

4.2.4 Case of economic dispatch problem with valve point effect

In this subsection, the economic dispatch problem with three generating units was re-examined, incorporating the valve-point loading effect of steam admission [22]. Consequently, the cost function exhibits a non-smooth, rippling shape, as illustrated in Figure 6 for the case of two generating units.

Table 6 summarizes the results, including the errors associated with the nonlinear equality constraint for each optimization method. The proposed algorithm demonstrated superior performance by achieving the lowest total cost, followed by the Artificial Bee Colony (ABC) algorithm, the Particle Swarm Optimization (PSO) method, the Lambda Iteration method, and the Genetic Algorithm (GA). In contrast, the Binary Bat Algorithm (BBT) produced an inaccurate solution, with a deviation of 20 MW in the power balance equation.

Table 6: Economic dispatch with valve point effect results [25].

Load demand	Parameter	Lambda	GA	PSO	ABC	BBT	**OCSA**
850 MW	P1 MW	382.258	382.2552	394.5243	300.266	296.3495	**300.265979611306**
	P2 MW	127.419	127.4184	200.000	149.733	199.6002	**149.733100113962**
	P3 MW	340.323	340.3202	255.4756	400.000	334.0503	**399.999999999999**
	Error	0	6.2e-003	1e-005	1e-003	-20	**9.2e-004**
Total cost Rs/h		8575.68	8575.64	8280.81	8253.10	8121.8568	**8253.08824756006**

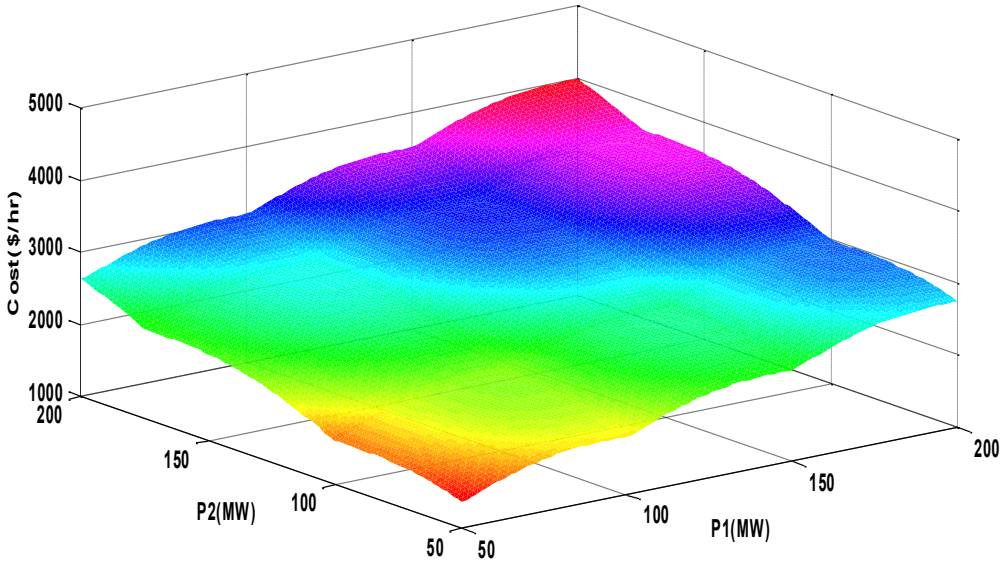


Figure 5: Illustration cost function for two production units

4.3 Simulation of difficult global optimization problems

In this section, for comparison with the optimization solvers Ipopt, LOQO, MINOS, PGAPack, PSwarm, and Couenne, we conducted the same set of simulations as Vanaret *et al.* [26]. In their study, a hybrid algorithm combining an Evolutionary Algorithm with interval-based methods (numerically certified using Charibde) was applied to a set of challenging optimization problems, including Michalewicz, Sine Envelope, Shekel, Egg Holder, Rana, and Keane functions. However, the Shekel function was excluded from our tests due to ambiguity in its definition, as multiple variants of the Shekel function exist and the specific version used was not clearly identified by the authors.

The simulation results are summarized in Table 7, presenting the obtained minimum values, along with the number of function evaluations and/or computation time when available. Based on these results, it is evident that our proposed OCSA algorithm outperforms several of the compared solvers, both in terms of global optimality and computational efficiency.

Table 7: State-of-the-art solvers against OCSA on test functions [26]

Method	Metric	Michalewicz (d=50)	Sine Envelope (d=5)	Egg Holder (d=5)	Rana (d=5)	Keane (d=4)
Ipopt	Value	-19.773742	-5.8351843	-3586.3131827	-75.512076	-0.2010427
	Eval	167	24	8	16	8
LOQO	Value	-0.0048572	-5.8351843	25.5609238	-69.5206	-0.0983083
	Eval	88	17	5924	138	50
MINOS	Value	0	-5.87878	-3586.313183	-233.592	-0.2347459
	Eval	3	38	3	1	3

PGAPack	Value	-37.60465	-5.569544	-3010.073	-2091.068	—
	Eval	9582	9615	9626	9622	—
PSwarm	Value	-24.38158	-5.835182	-2840.799	-1595.056	—
	Eval	2035	2049	2040	2046	—
Couenne	Value	-49.619042	-5.9660007	-3719.7287498	-2046.8320657	-0.6222999
	Time	265s	0.4s	20.6s	20.3s	2s
Charibde	Value	-49.6248323	-5.9659811	-3719.7248363	-2046.8320657	-0.6222810
	Time	8.4s	219s	0.8s	17.8s	14860s
	Eval	717100	227911400	742600	19186150	6351693297
OCSA	Value	-49.6248323040744	-5.966048150900620	-3719.724835658383	-2046.832065725112	-0.6222810255678127
	Time	223.3s	0.8s	13.1s	16.4s	12.9s
	Eval	8872	108	1702	2000	1000

4.4 Convergence Analysis

To further assess the convergence capabilities of the proposed Optimized Cuckoo Search Algorithm (OCSA), we conducted a series of rigorous experiments using the benchmark function F9, as illustrated in Figure 5. These tests were intentionally designed under extreme conditions to challenge the algorithm's robustness and efficiency. Specifically, we imposed a highly stringent termination criterion, requiring the relative error to fall below $\varepsilon < 10^{-400}$ with respect to the known global optimum. Additionally, we progressively increased the problem dimensionality, ranging from 10^1 to 10^6, to evaluate scalability. For comparison, we used the YCSA algorithm as a baseline, which has been reported to outperform established methods such as Particle Swarm Optimization (PSO) and Simulated Annealing (SA) [1, 3].

The results, summarized in Table 8, reveal several key insights. First, OCSA consistently outperforms YCSA across all tested dimensions, demonstrating superior convergence behavior and reliability in reaching the global optimum. Notably, while YCSA is recognized for its high performance, OCSA achieves even better results, indicating its robustness and enhanced optimization capacity. Furthermore, the algorithm shows remarkable scalability, maintaining its performance advantage even as the dimensionality increases to extremely high levels.

Convergence metrics, presented in Figure 7, further reinforce these findings. OCSA successfully reaches the global minimum with a final function value of fmin=0. Even at the highest tested dimensionality ($d=10^6$), the algorithm maintains efficient performance, with a total computation time of approximately 130 seconds. This result underscores the algorithm's practicality and effectiveness for solving large-scale, complex optimization problems.

Table 8: Comparison between YCSA [3] and OCSA of variation dimension of F9

Fun.	Meth.	d	10	10^2	10^3	10^4	10^5	10^6
F9	OCSA	*t*(sec)	0.031809	0.040294	0.103599	0.656107	13.066879	134.508003
		f_{min}	0	0	0	0	0	0
F9	YCSA	*t*(sec)	1.640971	*failed*	*failed*	*failed*	*failed*	*failed*
		f_{min}	0	*failed*	*failed*	*failed*	*failed*	*failed*
F9	SA	*t*(sec)	*failed*	*failed*	*failed*	*failed*	*failed*	*failed*
		f_{min}	*failed*	*failed*	*failed*	*failed*	*failed*	*failed*

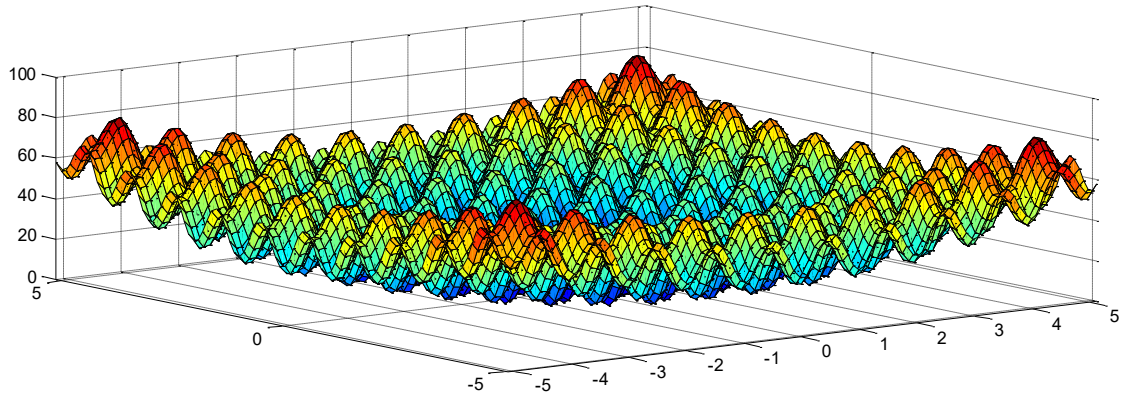


Figure 6: The landscape of F9's 2D function

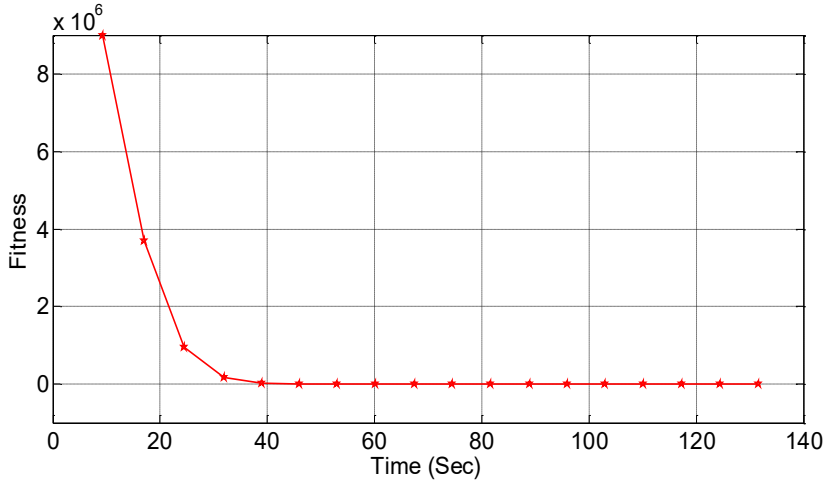


Figure 7: Convergence curve for the dimension d=10^6 of F9

5. Conclusion

This study presents an enhanced version of the Cuckoo Search Algorithm (CSA) with several improvements to address common limitations in metaheuristic optimization. First, a hybrid exploration strategy combines random and quasi-random search patterns to improve coverage of the solution space and maintain diversity among candidate solutions. Second, the algorithm strengthens exploitation while preserving global search capability. Carefully designed refinement phases enhance promising solutions without causing premature convergence. Additionally, the proposed method improves

computational efficiency through faster convergence, lower memory usage, and better performance on large-scale problems, making it suitable for high-dimensional applications.

The main contribution is the bidirectional integration of exploration and exploitation. Exploitation mechanisms operate during exploration, while exploration continues during exploitation, ensuring a balanced and adaptive search process. As a result, the algorithm effectively targets high-quality solutions, avoids suboptimal regions, prevents local optima stagnation, and maintains consistent performance across challenging scenarios. Comparative evaluations demonstrate its superiority over conventional algorithms and specialized solvers, with improved speed, accuracy, constraint handling, and strong potential for integration into other population-based metaheuristic frameworks.

Competing interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Funding

Not applicable.

References

[1] Yang XS, *Cuckoo Search and Firefly Algorithm: Theory and Applications*, Studies in Computational Intelligence, vol. 516, Springer, Warsaw, Poland, 2014.

[2] Tuba M, Subotic M, Stanarevic N, Modified cuckoo search algorithm for unconstrained optimization problems, in: Proc. 5th European Conf. on European Computing Conference (ECC'11), 2011, pp. 263–268.

[3] Yang XS, Deb S, Engineering optimization by cuckoo search, *Int. J. Math. Model. Numer. Optim.*, 1 (2010) 330–343.

[4] Bindu AH, Reddy MD, Economic load dispatch using cuckoo search algorithm, *Int. J. Eng. Res. Appl.*, 3 (2013) 498–502.

[5] Alireza G, Rozbeh J, A new approach for solving the unit commitment problem by cuckoo search algorithm, *Indian J. Sci. Technol.*, 6 (2013) 5235–5242.

[6] Chandrasekaran K, Simon SP, Multi-objective scheduling problem: Hybrid approach using fuzzy assisted cuckoo search algorithm, *Swarm Evol. Comput.*, 5 (2012) 1–16.

[7] Rao MR Mohana, Naresh Babu AV, Optimal power flow using cuckoo optimization algorithm, *Int. J. Adv. Res. Electr. Electron. Instrum. Eng.*, 2 (2013) 4213–4218.

[8] Yildiz AR, Cuckoo search algorithm for the selection of optimal machining parameters in milling operations, *Int. J. Adv. Manuf. Technol.*, 64 (2013) 55–61.

[9] Tawfik AS, Badr AA, Abdel-Rahman IF, One rank cuckoo search algorithm with application to algorithmic trading systems optimization, *Int. J. Comput. Appl.*, 64 (2013) 30–37.

[10] Agrawal S, Panda R, Bhuyan S, Panigrahi BK, Tsallis entropy based optimal multilevel thresholding using cuckoo search algorithm, Swarm Evol. Comput., 11 (2013) 16–30.

[11] Naumann DS, Evans B, Walton S, Hassan O, A novel implementation of computational aerodynamic shape optimisation using modified cuckoo search, *Appl. Math. Model.*, 40 (2016) 4543–4559.

[12] Farahani SM, Abshouri AA, Nasiri B, Meybodi MR, A Gaussian firefly algorithm, *Int. J. Mach. Learn. Comput.*, 1 (2011) 448–454.

[13] Ouyang, C., Liu, X., Zhu, D. et al. Hierarchical adaptive cuckoo search algorithm for global optimization. Cluster Comput 28, 321 (2025). https://doi.org/10.1007/s10586-024-04924-3.

[14] Mlakar U, Fister I Jr, Fister I, Hybrid self-adaptive cuckoo search for global optimization, *Swarm Evol. Comput.*, 11 (2016) 1–26.

[15] Zheng H, Zhou Y, A cooperative coevolutionary cuckoo search algorithm for optimization problem, *J. Appl. Math.*, 2013 (2013), Article ID 912056, 9 pages.

[16] Valian E, Mohanna S, Tavakoli S, Improved cuckoo search algorithm for global optimization, *Int. J. Commun. Inf. Technol.*, 1 (2011) 31–44.

[17] Tuba M, Cuckoo search optimization metaheuristic adjustment, in: *Recent Advances in Knowledge Engineering and Systems Science*, 2013, pp. 144–149.

[18] Shao-Qiang Ye, Azlan Mohd Zain, Yusliza Yusoff, Improved Cuckoo Search Algorithm for Engineering Optimization Problems, Computers, Materials and Continua, 87, 1, 2026, https://doi.org/10.32604/cmc.2025.073411.

[19] Makhadmeh, S.N., Awadallah, M.A., Kassaymeh, S. et al. Recent advances in Multi-objective Cuckoo Search Algorithm, its variants and applications. Arch Computat Methods Eng 32, 3213–3240 (2025). https://doi.org/10.1007/s11831-025-10240-9.

[20] Payne RB, Sorenson MD, Klitz K, *The Cuckoos*, Oxford University Press, Oxford, 2005.

[21] Yang XS, *Nature-Inspired Metaheuristic Algorithms*, 2nd ed., Luniver Press, UK, 2010.

[22] Yao X, Liu Y, Lin GM, Evolutionary programming made faster, *IEEE Trans. Evol. Comput.*, 3 (1999) 82–102.

[23] Leung YW, Wang YP, An orthogonal genetic algorithm with quantization for global numerical optimization, *IEEE Trans. Evol. Comput.*, 5 (2001) 41–53.

[24] Runarsson TP, Yao X, Stochastic ranking for constrained evolutionary optimization, *IEEE Trans. Evol. Comput.*, 4 (2000) 284–294.

[25] Mallikarjuna B, Reddy KH, Hemakesavulu O, Economic load dispatch problem with valve-point effect using a binary bat algorithm, *ACEEE Int. J. Electr. Power Eng.*, 4 (2013) 33–38.

[26] Vanaret C, Gottel JB, Durand N, Alliot JM, Certified global minima for a benchmark of difficult optimization problems, *HAL Id: hal-00996713*, 2014. Available: https://hal-enac.archives-ouvertes.fr/hal-00996713