Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

An adaptive dung beetle optimizer based on a resilient annealing mechanism and its application to numerical problems and optimization of Reed–Muller logic circuits*#

Lixin MIAO[1], Zhenxue HE‡[1], Xiaojun ZHAO[1], Yijin WANG[1], Xiaodan ZHANG[1], Kui YU[1], Limin XIAO[2], Zhisheng HUO[3]

[1]*Key Laboratory of Agricultural Big Data of Hebei Province, Hebei Agricultural University, Baoding 071001, China*

[2]*School of Computer Science and Engineering, Beihang University, Beijing 100191, China*

[3]*School of Electronic and Information Engineering, Beihang University, Beijing 100191, China*

E-mail: 20232060116@pgs.hebau.edu.cn; hezhenxue@buaa.edu.cn; xxzhxj@hebau.edu.cn; wangyijin0409@sina.com; 531123@hebau.edu.cn; yukui@hebau.edu.cn; xiaolm@buaa.edu.cn; huozhisheng1122@126.com

Received Nov. 1, 2024; Revision accepted May 26, 2025; Crosschecked

Unedited

Abstract: The dung beetle optimizer (DBO) is a metaheuristic algorithm with fast convergence and powerful search capabilities, which has shown excellent performance in solving various optimization problems. However, it suffers from the problems of easily falling into local optimal solutions and poor convergence accuracy when dealing with large-scale complex optimization problems. Therefore, we propose an adaptive dung beetle optimizer (ADBO) based on an elastic annealing mechanism to address these issues. First, the convergence factor is adjusted in a nonlinear decreasing manner to balance the requirements of global exploration and local exploitation to improve convergence speed and search quality; second, a greedy difference optimization strategy is introduced to increase population diversity, improve the global search capability, and avoid premature convergence; finally, the elastic annealing mechanism is used to perturb the randomly selected individuals, which helps the algorithm escape the local optimum to improve its solution quality and stability. Experimental results based on the CEC 2017 and CEC 2022 benchmark function sets and MCNC benchmark circuits verify the effectiveness, superiority, and universality of the ADBO.

Key words: Metaheuristic algorithms; Dung beetle optimizer; Convergence factor; Greedy difference optimization strategy; Elastic annealing mechanism

https://doi.org/10.1631/FITEE.2400967 **CLC number:**

1 Introduction

Optimization problems are prevalent in real life and engineering applications that are widely used in areas such as resource allocation, financial portfolio optimization, intelligent transportation (Han et al., 2024), drone path planning (Amores et al., 2024), and supply chain management (Li et al., 2023). Metaheuristic algorithms, inspired by natural phenomena (e.g., biological evolution and physical dynamics), have emerged as mainstream solutions for complex optimization tasks, particularly in high-dimensional or uncertain problem spaces (Ali et al., 2015; Ikram et al., 2023).

‡ Corresponding author

* Project supported by the National Natural Science Foundation of China (No. 62102130), Central Government Guides Local Science and Technology Development Fund Project, China (No. 226Z0201G), Natural Science Foundation of Hebei Province, China (Nos. F2020204003 and F2024204001), Hebei Youth Talents Support Project, China (No. BJ2019008), Science and Technology Research Projects of Higher Education Institutions in Hebei Province, China (No. QN2024138)

Electronic supplementary materials: The online version of this article includes supplementary materials that are accessible to authorized users

ORCID: Lixin MIAO, https://orcid.org/0000-0002-2558-3885; Zhenxue HE, https://orcid.org/0000-0001-7041-8582

© Zhejiang University Press 2025

These algorithms simulate natural processes (e.g., evolutionary mechanisms or collective behaviors) to approximate global optimal solutions. Their core strengths include robust global search to avoid local optima, flexibility without strict mathematical formulations, and inherent parallelism for scalability. Metaheuristic algorithms are broadly classified into four categories based on design principles:

1. Evolutionary Algorithms (e.g., Genetic Algorithm [GA] (Grefenstette, 1993), Differential Evolution [DE] (Storn and Price, 1997), Cultural Algorithm [CA] (Reynolds and Peng, 2004)), which mimic biological evolution using mechanisms such as selection, mutation, and social learning;

2. Swarm-based Algorithms (e.g., Artificial Rabbits Optimization [ARO] (Wang et al., 2022), Tunicate Swarm Algorithm [TSA] (Kaur et al., 2020), Mountain Gazelle Optimizer [MGO] (Abdollahzadeh et al., 2022)), which are inspired by collective animal behaviors and address multimodal or high-dimensional optimization;

3. Physics-based Algorithms (e.g., Simulated Annealing [SA] (Kirkpatrick et al., 1983), Thermal Exchange Optimization [TEO] (Kaveh and Dadras, 2017), Gravitational Search Algorithm [GSA] (Rashedi et al., 2009)), which leverage natural phenomena such as thermodynamics and gravitational interactions for global search;

4. Human-based Algorithms (e.g., Society Civilization Algorithm [SCA] (Ray and Liew, 2003), Anarchic Society Optimization [ASO] (Ahmadi-Javid, 2011), Teamwork Optimization Algorithm [TOA] (Dehghani and Trojovský, 2021)) model societal collaboration, cultural dynamics, or group coordination to solve multi-objective or complex engineering problems.

While distinct in their inspiration, all categories inherently balance exploration–exploitation tradeoffs and operate with minimal prior knowledge of problem structures.

The rapid advancement of digital circuit technology has positioned performance enhancement as a central research focus. Reed–Muller (RM) logic offers significant advantages over traditional Boolean logic in optimizing area (He et al., 2021; Shao et al., 2021), speed, and power consumption for specialized circuits like arithmetic units and parity-check systems. RM circuits are classified as fixed-polarity (FPRM, XNOR/OR-based) or mixed-polarity (MPRM, XOR/AND-based), and polarity optimization is critical to performance, given the direct impact of polarity combinations on efficiency.

Identifying optimal polarities within exponentially large search spaces is a key challenge: Exhaustive search suffices for small-to-medium circuits but is infeasible for large-scale designs due to computational complexity. Intelligent algorithms (e.g., GA, DE) improve search efficiency in this combinatorial task. This paper addresses the underexplored area of optimization in FPRM (Sun et al., 2013) and MPRM (Wang et al., 2010) circuits by improving intelligent algorithms to find optimal polarities with minimal logic gates, aiming to boost circuit performance and promote RM logic adoption.

This paper presents an adaptive dung beetle optimizer (ADBO) that features the following three innovations:

1. Dynamic Decay Factor Strategy: An adaptive convergence factor adjustment balances global exploration and local exploitation by modulating decay rates across optimization stages, ensuring sustained efficacy.

2. Greedy Difference Optimization Strategy: A hybrid crossover mechanism for mutant/stealing beetles enhances population diversity and global search, while greedy selection retains elite solutions to accelerate convergence.

3. Elastic Annealing Mechanism: Dual-phase temperature scheduling—rapid cooling in early stages and gradual reduction later—perturbs random individuals in the final iterations to refine solutions, enhancing robustness and convergence reliability.

Comparative experiments using the CEC 2017 and CEC 2022 benchmark function sets and MCNC benchmark circuits demonstrate the superior performance of the ADBO algorithm, achieving higher convergence precision and global exploration capability than conventional and state-of-the-art swarm intelligence algorithms.

2 Related Works

The Dung Beetle Optimizer (DBO), a swarm intelligence algorithm inspired by dung beetle behaviors (Xue and Shen, 2023), offers robust global

Unedited

search, rapid convergence, and simplicity, with applications in robotic path planning and UAV trajectory optimization. Subsequent improvements aim to enhance its performance:

Zhu et al. (2024) integrated quantum computing and multi-strategy fusion, using good point set initialization for uniform population distribution, adaptive convergence factors to balance exploration–exploitation, and quantum-based perturbation to avoid local optima.

Shen et al. (2023) proposed a multi-strategy DBO with Beta-distribution-based dynamic inverse learning for solution space exploration, Levy distribution-based boundary handling, and dual crossover operators to balance search capabilities.

Jiachen and Li-hui (2024) improved DBO via Chebyshev chaos mapping for homogeneous initial populations, a curve-adaptive golden sine strategy for faster convergence, and Cauchy-t variation with Levy flights to balance global and local search.

Li et al. (2023) introduced stochastic inverse learning for diversity, fitness-distance balancing, spiral foraging for position updating, and Gaussian variation to escape local optima.

While these advancements enhance convergence and accuracy, challenges remain. The excessive focus on individual optimization during the stealing phase reduces population diversity, and over-reliance on current best solutions in later iterations risks trapping the algorithm in local optima, limiting its ability to discover superior solutions in high-dimensional spaces.

3 Dung Beetle Optimizer

The DBO categorizes the population as four types of search agents: rolling dung beetles, breeding dung beetles, foraging dung beetles, and stealing dung beetles.

3.1 Rolling dung beetle

When rolling a dung ball, a dung beetle relies on the sun or other celestial cues to navigate along a straight path. The selected rolling mode includes both an unobstructed mode and an obstructed mode.

3.1.1 Accessibility Model

Assuming that light source intensity also affects the dung beetle's path, the position of the rolling dung beetle is updated during the rolling process, as expressed by Eq. (1).

$$\begin{aligned} x_i(t+1) &= x_i(t) + \alpha \cdot k \cdot x_i(t-1) + b \cdot \Delta x \\ \Delta x &= \left| x_i(t) - X^w \right| \end{aligned}, \quad (1)$$

where t denotes the current iteration number, $x_i(t)$ denotes the location information of the i-th rolling dung beetle at the t-th iteration, and $k \in (0,0.2]$ denotes the constant value of the deflection coefficient. b indicates a constant value belonging to (0,1), where the value of α (assigned as 1 or −1 with a specified probability) indicates whether the dung beetle deviates from its path (1 = no deviation, −1 = deviation). X^w denotes the global worst position, and Δx simulates changes in illumination, such that greater illumination corresponds to the dung beetle being farther from the light source.

3.1.2 Obstacle Mode

The tangent function calculates a new rolling direction and mimics the beetle's dancing behavior, with the position update formula for the rolling dung beetle becoming Eq. (2).

$$x_i(t+1) = x_i(t) + | x_i(t) - x_i(t-1) | \cdot \tan(\theta), \quad (2)$$

where θ is the angle of deflection associated with [0, π]. The position of the rolling dung beetle is not updated when $\theta = 0$, $\pi/2$, and π.

3.2 Breeding dung beetle

A boundary selection strategy was proposed to simulate dung beetles' behavior in choosing a safe region to lay their eggs, with the optimal spawning area defined by Eq. (3).

$$\begin{aligned} Lb^* &= \max(X^* \cdot (1-R), Lb), \\ Ub^* &= \min(X^* \cdot (1+R), Ub) \end{aligned} \quad (3)$$

where Lb^* and Ub^* represent the lower and upper bounds of the optimal spawning region, X^* denotes the current local optimal position, and $R = 1-t/T_{max}$. T_{max} is the maximum number of iterations. Lb and Ub are the lower and upper bounds of the search space, respectively. Specifically, the position update of the breeding dung beetle is described by Eq. (4).

$$B_i(t+1) = X^* + b_1 \cdot (B_i(t) - Lb^*) + b_2 \cdot (B_i(t) - Ub^*), \quad (4)$$

where $B_i(t)$ is the positional information of the i-th breeding ball at the t-th iteration, b_1 and b_2 are two

Unedited

independent random vectors of size 1×D, and D is the dimension of the optimization problem.

3.3 Foraging dung beetle

In nature, dung beetles select safe foraging areas; the boundaries of the optimal foraging area are defined by Eq. (5).

$$\begin{aligned} Lb^b &= \max(X^b \cdot (1-R), Lb), \\ Ub^b &= \min(X^b \cdot (1+R), Ub) \end{aligned} \tag{5}$$

where Lb^b and Ub^b represent the lower and upper bounds of the optimal foraging area, and X^b represents the global optimal position. Therefore, the location update information of the foraging dung beetle is shown in Eq. (6).

$$x_i(t+1) = x_i(t) + C_1 \cdot (x_i(t) - Lb^b) + C_2 \cdot (x_i(t) - Ub^b), \tag{6}$$

where $x_i(t)$ denotes the location information of the i-th foraging dung beetle at the t-th iteration, C_1 denotes a random number obeying a normal distribution, and C_2 denotes a random vector belonging to (0,1).

3.4 Stealing dung beetle

In nature, some dung beetles choose to steal dung balls from other dung beetles, and the location update of a stealing dung beetle is given by Eq. (7).

$$x_i(t+1) = X^b + S \cdot g \cdot (|x_i(t) - X^*| + |x_i(t) - X^b|), \tag{7}$$

where $x_i(t)$ denotes the location information of the i-th stealing dung beetle at the t-th iteration, g is a random vector of size 1×D following a normal distribution, and S denotes a constant value.

4 Adaptive Dung Beetle Optimizer

This paper proposes three improvement strategies to further improve DBO's search performance.

4.1 Dynamic decay factor

The linear decay of the convergence factor with iterations may imbalance global exploration and local exploitation, risking premature local optima and reducing solution accuracy. To address this, a non-linear decay method dynamically adjusts the convergence factor's decay rate, prioritizing global exploration in early iterations and local exploitation in later stages to balance optimization speed and quality. This paper introduces a dynamic decay factor strategy defined by Eq. (8).

$$\begin{aligned} &R_g = R \cdot \exp(-G \cdot t) \\ &R_l = R \cdot \exp(-L \cdot (t - P \cdot T_{\max})) \\ &R_g \in [0.1, 0.9] \\ &R_l \in [0.1, 0.9 \cdot \exp(-L \cdot P \cdot T_{\max})] \end{aligned}, \tag{8}$$

where R is the convergence factor for nonlinear descent, G and L are the decay rates for the global and local search phases, respectively, and L = 0.1. T_{max} is the maximum number of iterations, t is the current number of iterations, and P is the threshold between the global and local search phases. R_g and R_l are the convergence factors for the global and local search phases, respectively. If $t < P \cdot T_{max}$, ADBO operates in the global search phase, and the scope of the spawning and foraging areas is determined by R_g; otherwise, it is governed by R_l.

4.2 Greedy difference optimization

In the algorithm's stealing phase, position updates based on current optimal states accelerate global convergence but reduce population diversity, risking premature clustering at local optima and degrading solution quality. To address this, a greedy difference optimization strategy enhances diversity by maintaining efficient search and improving global exploration to avoid premature convergence.

Inspired by DE and its variation/crossover operators, the greedy difference optimization strategy (Fig. 1) enhances search flexibility via adaptive variation operators. During the stealing phase, three random individuals undergo mutation to generate mutants, which crossover with stealing dung beetles to form hybrid individuals. Fitness values from the differential optimization strategy (hybrids) are compared with those from the original position update, and the greedy rule retains individuals with superior fitness, as defined by Eq. (9).

$$X_i(t+1) = X_\alpha(t) + M \cdot (X_\beta(t) - X_\eta(t)), \tag{9}$$

where $X_i(t+1)$ is the location information of the stealing dung beetle i-th after mutation, M is the mutation rate, α, β, and η are population-wide random numbers, and $X_\alpha(t)$, $X_\beta(t)$, and $X_\eta(t)$ represent three individuals randomly selected from the population at the t-th iteration, respectively.

After mutation, a variable-wise dimensional crossover is performed on the target vector $objective_i(t+1)$: if $rand() \le C$ (where C is the crossover rate) or $j = randi(D)$ (where D is the decision vector di-

Unedited

mension), the j-th dimension of $objective_i(t+1, j)$ is replaced with the corresponding dimension from the mutated vector $X_i(t+1, j)$, forming the differential optimization-based objective vector. Subsequently, the fitness values of individuals generated by the differential optimization and original position update strategies are compared, and the greedy rule retains the individuals with better fitness.

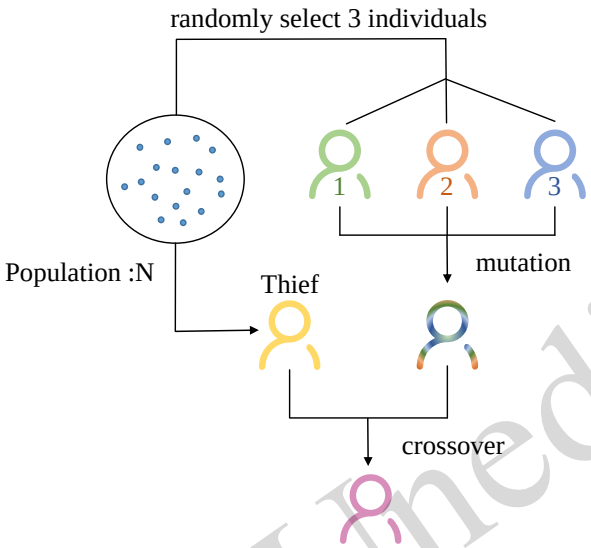


Fig. 1 Greedy difference optimization

4.3 Elastic annealing mechanism

To prevent premature local convergence, this paper introduces an elastic annealing mechanism inspired by SA, mimicking metallic material annealing (heating, holding, and controlled cooling) to balance global exploration and local exploitation via adaptive cooling rates. Unlike SA, it dynamically adjusts cooling speed: early-stage fast cooling allows for the acceptance of suboptimal solutions to avoid local traps, while late-stage slow cooling prioritizes refining optimal solutions. Fig. 2 simulates the metal annealing process—heating increases grain disorder and internal energy, and slow cooling reorders grains to minimize energy and stabilize the material in its ground state at room temperature.

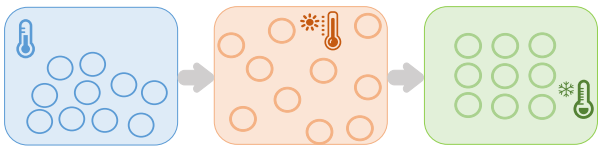


Fig. 2 Metal annealing model

The simulated annealing mechanism begins with a high initial temperature and probabilistically accepts suboptimal solutions as the temperature decreases, enabling random search in the solution space to escape local optima and converge to the global optimum. This can be illustrated by an analogy (Fig. 3): a drunk rabbit (representing the algorithm) randomly explores the mountain (solution space), occasionally moving to higher elevations (better solutions) or local optima (flatter areas), but as it sobers up (temperature drops), it prioritizes higher ground, eventually reaching the mountain top (global optimum). This highlights how the algorithm balances exploration via probabilistic acceptance at high temperatures with exploitation of better solutions during cooling to ensure global convergence.

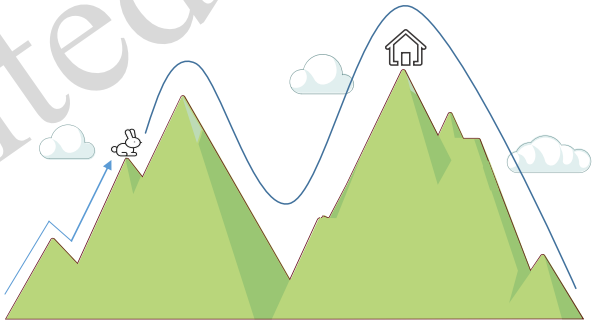


Fig. 3 Rabbit mountain climbing model

The elastic annealing mechanism dynamically balances exploration and exploitation by using faster temperature cooling in the early stages for global search and slower cooling in the later stages for local refinement. At each iteration's end, three randomly selected individuals undergo the strategy, with their fitness compared to annealed individuals; a greedy rule retains those with better fitness, as specified in Eq. (10).

$$\begin{aligned} \tau &= I(t)/T_{\max} \\ I(t+1) &= I(t)\cdot(1-\tau) \end{aligned}, \tag{10}$$

where t and T represent the current and maximum number of iterations of the annealing stage, respectively. The term τ denotes the adaptive annealing rate, while $I(t)$ represents the annealing temperature at the t-th iteration.

4.4 Algorithm description

The ADBO pseudo-code (Algorithm 1) initializes parameters and randomly initializes individual

Unedited

positions, updating dung beetle positions via steps 2–28. Key steps include a dynamic decay factor in step 13 to balance global/local search, a greedy difference optimization strategy in step 22 to boost diversity, and an elastic annealing mechanism in step 25 to avoid local optima. The algorithm outputs the best position X^b and fitness value f_b after optimization.

Algorithm 1 Pseudo-code of ADBO algorithm

Input: Maximum iterations T_{max}, number of search agents N, dimension D.

Output: Optimal position X^b, optimal fitness value f_b.

1: Initialize the positions of the dung beetles
2: **While** $t \le T_{max}$ **do**
3: Calculate the current best position and its fitness
4: **For** i = 1:N **do**
5: **if** i == Rolling dung beetles **then**
6: $\alpha = rand()$
7: **if** $\alpha \le 0.9$ **then**
8: Update the rolling dung beetles' position by Eq.(1)
9: **else**
10: Update position by Eq.(2)
11: **end if**
12: **end if**
13: Nonlinear convergence factor is calculated by Eq.(8)
14: **if** i == Breeding dung beetles **then**
15: Update the breeding dung beetles' position by Eq.(3) and Eq.(4)
16: **end if**
17: **if** i == Foraging dung beetles **then**
18: Update the foraging dung beetles' position by Eq.(5) and Eq.(6)
19: **end if**
20: **if** i == Stealing dung beetles **then**
21: Update the stealing dung beetles' position by Eq.(7)
22: First, a differential optimization strategy is applied to the rogue beetle individuals, and then the greedy rule is used to compare them with the original individuals to retain the dominant individuals.
23: **end if**
24: **end For**
25: Three individuals were randomly selected from the population for the elastic annealing strategy, and greedy selection was used to retain the better individuals.
26: Update the best position and its fitness
27: $t = t+1$
28: **end While**
29: Return the optimal solution X^b and its fitness value f_b

Fig. 4 illustrates the main flow of ADBO, while Table 1 presents the specific value ranges of various parameters in the algorithm.

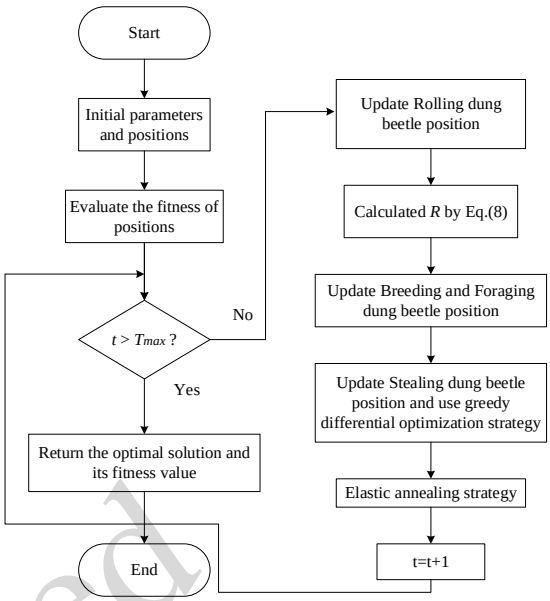


Fig. 4 The flow chart of ADBO

Unedited

4.5 Time complexity analysis

Assuming a population size of N (with N/5+1 stealing dung beetles), variable dimension D, and maximum iterations T, the DBO algorithm has a complexity of $O(N)$. For ADBO, time complexity analysis from its pseudo-code exhibits contributions from two main steps: 1) position updates via variation and crossover operators for stealing dung beetles, with complexity $O((N/5+1)\cdot T)$; and 2) annealing variation for three randomly selected individuals, with complexity $O(3T)$. Combining these, ADBO's overall complexity is $O(N)+O((N/5+1)\cdot T)$. Despite this, ADBO maintains an overall time complexity comparable to that of the original algorithm ($O(N)$), while demonstrating superior performance.

5 Experimental results and analysis

5.1 Configuration of the Experimental Environment

To ensure the experiment's rigor and fairness, the simulation experiment is configured using the following hardware and software environment:

Processor: Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz 1.19GHz.

Table 1 ADBO parameters setting

Parameter Type	Parameter	Description	Range
User-defined Parameters	N	Population Size	-
	T	Maximum Iterations	-
	D	Dimension	-
Custom Parameters	G	Global Convergence Factor	$G\in[0.02,0.04]$
	P	Transition Threshold	$P\in[0.55,0.75]$
	M	Mutation Frequency	$M\in[0.7,0.9]$
	C	Crossover Frequency	$C\in[0.8,1.1]$
	I	Initial Annealing Temperature	$I\in[2,9]$

Memory: 8 GB 3733MHz
Graphics: NVIDIA GeForce MX250
Simulation: MATLAB R2022b
Operating System: Windows 10 (64-bit)

5.2 Experimental Parameter Settings

To evaluate ADBO's performance, this paper compares ADBO with 12 algorithms on 41 test functions from the CEC 2017 and 2022 benchmark sets. The experiments use uniform parameter settings: a population size of 30, a maximum of 500 iterations, and 100 independent runs per function. The mean, standard deviation, and ranking of solution results for each test function are recorded, ensuring fair comparisons under identical initial conditions to enhance experimental reliability and repeatability.

Twelve comparison algorithms, including Particle Swarm Optimization (PSO) (Kennedy J and Eberhart R, 1995), Grey Wolf Optimizer (GWO) (Mirjalili et al., 2014), the Horned Lizard Optimization Algorithm (HLOA) (Peraza-Vázquez et al., 2024), the Whale Optimization Algorithm (WOA) (Mirjalili and Lewis, 2016), the Fireworks Algorithm (FWA) (Tan and Zhu, 2010), the Harris Hawk Optimization (HHO) (Heidari et al., 2019), the Coati Optimization Algorithm (COA) (Dehghani et al., 2023), the Ant Nesting Algorithm (ANA) (Hama Rashid et al., 2021), the Greylag Goose Optimization (GGO) (El-Kenawy et al., 2024), the Fire Hawk Optimizer (FHO) (Azizi et al., 2023), the GOOSE Algorithm (GOOSE; Hamad and Rashid, 2024), and DBO, are compared with ADBO. Parameter settings for the comparison algorithms are presented in Table 2. The specific configurations of benchmarking functions and comparison algorithms are as follows:

1. The comparison algorithms for the CEC 2017 benchmark functions include HLOA, WOA, FWA, HHO, COA, DBO, ANA, GGO, FHO, and GOOSE.

2. For the CEC 2022 benchmark functions, PSO, GWO, WOA, HHO, COA, DBO, ANA, GGO, FHO, and GOOSE are selected as comparison algorithms.

3. The Wilcoxon rank sum and Friedman test comparison algorithms comprise FWA, WOA, GWO, HHO, COA, DBO, HLOA, ANA, GGO, FHO, and GOOSE.

Table 2 provides details of the specific parameters for each algorithm, as described below.

1. In PSO, $C1$ and $C2$ are the individual and social learning factors, respectively.

2. HLOA uses a random number in (0,1) to model the stochastic behavior of horned lizard target attacks.

3. WOA's parameter α is a linear decreasing factor that defines the algorithm's search range.

4. FWA includes explosion radius (Er), explosion count (En), variable sparks (M), and explosion limit factors (a,b).

5. HHO's E (a random number in $[-1,1]$ represents prey escape strength, and Js is the jump strength.

6. COA's parameter I is randomly generated.

7. DBO's k (deflection coefficient constant, k = 0.1 and b (constant, b = 0.3) fall within (0,0.2] and (0,1), respectively, with S as a constant.

8. ANA's PH_d is the pheromone decay rate.

9. FHO's Ir is a randomly generated value influencing firehawk behavior (e.g., attractiveness, step size, movement control).

10. GGO's Z decays exponentially, a decreases linearly from 2 to 0, and A is the scaling factor for the

Unedited

leader's solution.

11. GOOSE's alpha parameter (0,2) decreases significantly with each iteration, and the stone weight (*S_W*) ranges from 5 to 25 grams.

To ensure fairness, all parameters of the above algorithms, except the random ones, are configured using default settings from the original literature.

Table 2 Parameter variable settings

Algorithm	Parameters
PSO	Inertia weight linear reduction from 0.9 to 0.1, $C1 = C2 = 2$.
GWO	The convergence factor is linearly decreasing from 2 to 0 during iterations.
HLOA	$\beta = rand()$
WOA	$\alpha = 2\cdot(1-t/T_{max})$
FWA	$M = 5$, $En = 6$, $Er = 5$, $a = 0.3$, $b = 0.6$
HHO	$E\in[-1,1]$, $Js = 2\cdot(1-rand)$
COA	$I = round(1+rand(1,1))$
DBO	$k = 0.1$, $b = 0.3$, $S = 0.5$
ANA	$PH_d = 0.95$
FHO	$Ir = unifrnd(0,1,1,2)$
GGO	$Z = 1-(t/T_{max})^2$, $A = 2a\cdot rand-a$
GOOSE	$alpha = (2-(loop/(Max_It/2)))$ $S_W = randi([5,25],1,1)$

5.3 Orthogonal Experiments

In this paper, the $L_{27}(3^6)$ orthogonal table was used to design orthogonal experiments for investigating the effects of the following six factors on response variables: global exploration decay rate (*G*), critical point (*P*), mutation rate (*M*), crossover rate (*C*), initial annealing temperature (*I*), and number of iterations in the annealing phase (*T*). Systematic factor settings enable detailed analysis of their impacts on response variables, facilitating the search for optimal factor combinations to achieve research objectives. The response variables represent experimental outputs influenced by these factors, with the aim of identifying the factor sets that significantly affect them to optimize experimental design for optimal results. A series of trials was conducted following the orthogonal table design, recording factor level combinations and corresponding response variables in each trial, as detailed in Table 3.

This orthogonal experiment was conducted on 12 CEC 2022 benchmark test functions, each running in 30 repeated loops. Optimal values from each run were averaged; these averages were further averaged to yield final experimental results. Table 3 shows that Group 8's experiments produced the best results; thus, the factor levels designed for Group 8 were adopted as the final parameter values in the table.

Table 3 Orthogonal experimental design table

Level	*G*	*P*	*M*	*C*	*I*	*T*	Average
1	0.02	0.55	0.7	0.8	2	10	1636.67
2	0.02	0.65	0.8	0.9	5	15	1631.82
3	0.02	0.75	0.9	1.1	9	20	1631.55
4	0.03	0.55	0.8	0.9	2	15	1632.53
5	0.03	0.65	0.9	1.1	5	20	1628.72
6	0.03	0.75	0.7	1.1	9	10	1630.60
7	0.04	0.55	0.9	1.1	5	15	1628.70
8	**0.04**	**0.65**	**0.7**	**0.8**	**9**	**20**	**1627.23**
9	0.04	0.75	0.8	0.9	2	10	1633.48
10	0.02	0.65	0.7	0.9	5	20	1629.28
11	0.02	0.75	0.8	1.1	9	10	1628.89
12	0.02	0.55	0.9	0.8	2	15	1650.09
13	0.03	0.75	0.7	0.9	9	20	1628.94
14	0.03	0.55	0.8	1.1	5	10	1631.64
15	0.03	0.65	0.9	0.8	9	15	1642.81
16	0.04	0.55	0.7	0.9	5	20	1630.60
17	0.04	0.65	0.8	1.1	9	15	1628.97
18	0.04	0.75	0.9	0.8	2	10	1650.30
19	0.02	0.55	0.8	1.1	9	15	1631.28
20	0.02	0.65	0.9	0.8	2	10	1643.18
21	0.02	0.75	0.7	0.9	5	20	1632.11
22	0.03	0.55	0.9	1.1	9	10	1633.18
23	0.03	0.65	0.7	0.8	2	15	1633.13
24	0.03	0.75	0.8	0.9	5	20	1632.06
25	0.04	0.55	0.7	1.1	9	15	1633.67
26	0.04	0.65	0.8	0.8	2	20	1635.33
27	0.04	0.75	0.9	1.1	5	10	1632.63

Unedited

5.4 Ablation Experiments

Ablation experiments are used to assess the impact of the dynamic decay factor strategy, greedy difference optimization strategy, and elastic annealing mechanism on the overall performance of DBO. This paper compares DBO with ADBO, incorporating these three strategies to verify their effectiveness and stability. The study involved conducting 30 independent trials on 12 functions from the CEC 2022 benchmark set, recording the best values achieved by each algorithm for each function, and calculating the mean and variance for statistical analysis and comparison. Additionally, convergence curves are plotted for all algorithms across different functions (Fig. 5, see Supplementary Materials) to enable comprehensive performance evaluation, with detailed results presented in Table 4.

The experimental results demonstrate that an algorithm combining the dynamic decay factor strategy (Algorithm A), greedy difference optimization strategy (Algorithm B), and elastic annealing mechanism (Algorithm C) outperforms DBO. Compared to DBO, this fused algorithm exhibits superior performance on specific metrics, with enhanced efficiency and stability. Furthermore, ADBO, which integrates all three strategies simultaneously, significantly outperforms algorithms that incorporate only one strategy. These ablation experimental results confirm the effectiveness and stability of the three proposed improvement strategies.

Table 4 Ablation experiment results

Fun	Mtr	DBO	A	B	C	ADBO
F1	Ave	2167.65	1193.74	479.76	314.99	300.02
	Std	1216.87	787.02	179.95	30.80	0.04
F2	Ave	413.61	410.35	408.26	404.54	403.40
	Std	18.58	13.59	3.21	1.42	1.72
F3	Ave	600.43	600.19	600.00	600.03	600.00
	Std	0.76	0.48	0.00	0.05	0.00
F4	Ave	833.26	829.15	829.06	815.87	814.13
	Std	7.09	8.86	9.46	4.06	4.00
F5	Ave	902.42	901.86	900.13	900.06	900.01
	Std	3.55	2.43	0.35	0.07	0.04
F6	Ave	6117.12	4947.05	2058.45	6018.70	1822.60
	Std	3673.38	2318.55	1122.88	2339.42	8.91
F7	Ave	2025.83	2024.84	2020.38	2021.53	2014.33
	Std	10.03	4.79	5.64	6.25	8.42
F8	Ave	2227.99	2226.70	2222.31	2222.70	2216.36
	Std	3.27	5.28	7.48	5.66	6.66
F9	Ave	2534.21	2534.20	2534.18	2492.98	2492.64
	Std	26.82	26.82	26.83	1.89	1.18
F10	Ave	2568.85	2562.51	2550.04	2548.81	2544.97
	Std	61.12	59.20	58.01	56.38	55.63
F11	Ave	2645.13	2643.77	2618.35	2600.00	2600.00
	Std	70.12	107.52	77.13	0.00	0.00
F12	Ave	2868.00	2865.62	2863.93	2856.56	2856.14
	Std	10.90	5.01	1.76	1.40	1.39

5.5 Analysis of Results for CEC 2017 Benchmark Functions

5.5.1 Analysis of CEC 2017 Data

We selected 29 single-objective test functions from the CEC 2017 test suite (excluding F2 due to uncontrollable factors) for experimental validation. To assess the comprehensive search capabilities of metaheuristic algorithms fairly, each experiment involved 100 independent runs to compare optimization performance and evaluate ADBO's scalability across dimensions (10, 30, 50, and 100). Performance was quantified using three metrics—mean, standard deviation, and average rank—to enable thorough analysis of each algorithm's behavior across different dimensional spaces. Experimental results for all algorithms on the CEC 2017 test set across these dimensions are documented in Tables 5, 6, 7, and 8 (see Supplementary Materials), with a detailed analysis and discussion of the findings provided below.

1. ADBO excels in solving the single-peak problem F1, consistently outperforming other algorithms across 10, 30, 50, and 100 dimensions by achieving minimal variance and optimal values in all cases.

2. ADBO generally outperforms other algorithms in the simple multi-modal problem experiments F3–F10. For F3, it converges more slowly in 30- and 100-dimensional cases (underperforming HLOA/COA) but remains competitive in 50 dimensions and achieves the global optimum in 10 dimensions. For F4–F8, ADBO significantly surpasses the others in convergence speed and accuracy, consistently obtaining optimal solutions. For F9/F10, GOOSE slightly outperforms ADBO in 100-dimensional speed, while ADBO dominates in low-to-medium dimensions.

3. In hybrid problem experiments F11–F20, ADBO outperforms other algorithms in 10-, 30-, and 100-dimensional tasks by rapidly converging to global optima with low variance and stable results. While slightly less effective than DBO for F13/F15 in 50 dimensions, ADBO remains robust in handling complex hybrid problems.

4. ADBO excels on composite functions F23–F29, rapidly converging to optimal results across four dimensions via its elastic annealing mechanism,

Unedited

which enhances escape from local optima. While slightly outperformed by GOOSE/DBO in 30-/100-dimensional F22 cases and marginally lagging DBO in 50-dimensional F30 experiments, ADBO remains highly competitive in most scenarios.

Overall, ADBO demonstrates robust performance in handling diverse complex problems, particularly multidimensional ones, affirming its potential as a robust optimization algorithm.

5.5.2 CEC 2017 Convergence Curve Analysis

Fig. 6 (see Supplementary Materials) shows the average ranking of all algorithms across 29 functions in 30- and 100-dimensional spaces, with ADBO achieving the lowest average ranking and thus demonstrating superior overall performance compared to other algorithms. Fig. 7 illustrates the convergence profiles of different algorithms for F1 and F3–F29 across 10-, 30-, 50-, and 100-dimensional problem spaces (F1, F3–F8 in 10D; F9–F15 in 30D; F16–F22 in 50D; F23–F29 in 100D). ADBO excels on single-peak and simple multi-peak functions, rapidly converging to near-global optima, highlighting its efficient solution-space exploration and ability to approach global optima quickly. While DBO and ADBO exhibit similar performance on F27 and F29 for composite functions, ADBO maintains superior competitiveness in most experiments. Overall, ADBO's fast convergence, efficient search, and adaptability underscore its competitiveness across diverse complex optimization problems.

5.6 Analysis of Results for CEC 2022 Benchmark Functions

5.6.1 Analysis of CEC 2022 Data

CEC 2022 includes 12 single-objective test functions. The experimental results of all algorithms on 10-dimensional and 20-dimensional CEC 2022 benchmark test sets are shown in Tables 9 and 10 (Supplementary Materials), with detailed analyses as

Unedited

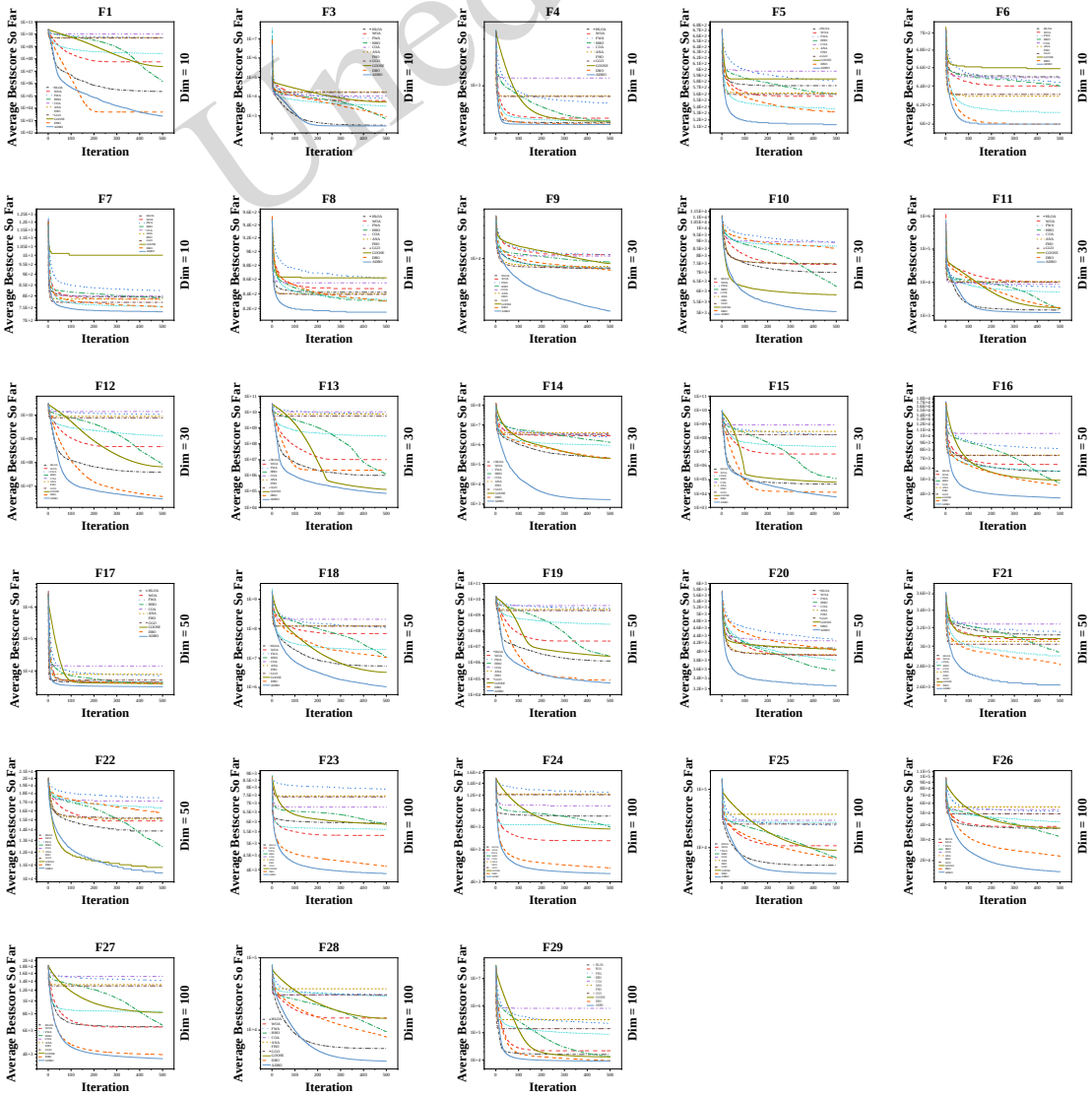


Fig. 7 CEC 2017 convergence curve

follows:

For the F1 single-peak function, ADBO significantly outperforms other algorithms in 10D, converging to the function's optimal value while others fail to match its accuracy. While PSO converges better than ADBO for F1 and F4 in 20D, ADBO is more efficient and accurate in lower-dimensional optimization. For basic functions F2–F5, ADBO demonstrates rapid convergence to optimal solutions, exhibiting excellent optimization performance.

ADBO excels in hybrid function experiments. For F6, it escapes local optima and converges to the optimal neighborhood in 10D, demonstrating robust global search capabilities. While GOOSE slightly outperforms ADBO in 20D optimal solution-finding, the margin is minimal, and ADBO still efficiently explores the solution space to identify high-quality solutions.

For composite functions F9–F12, ADBO exhibits optimization capability comparable to that of other algorithms but with significantly faster convergence speed and higher solution quality in global optimum achievement.

5.6.2 CEC 2022 Convergence Curve Analysis

Fig. 8 (Supplementary Materials) presents the average ranking of all algorithms on 10-dimensional and 20-dimensional CEC 2022 test sets. ADBO achieves the lowest average rank across all functions in 10D, demonstrating superior optimization performance. In 20D, ADBO maintains strong performance with excellent average rankings, highlighting its adaptability to multidimensional problems.

Fig. 9 presents the convergence curves of algorithms in 10D and 20D (F1–F6 in 10D; F7–F12 in 20D). All algorithms converge rapidly to global optima with high accuracy and robustness. On composite function F11, DBO and ADBO exhibit comparable convergence speeds and accuracies, indicating similar optimization capabilities. However, experimental data show that ADBO slightly outperforms DBO in search capability, enabling higher-quality solutions for certain composite functions and demonstrating superiority in complex optimization tasks.

Unedited

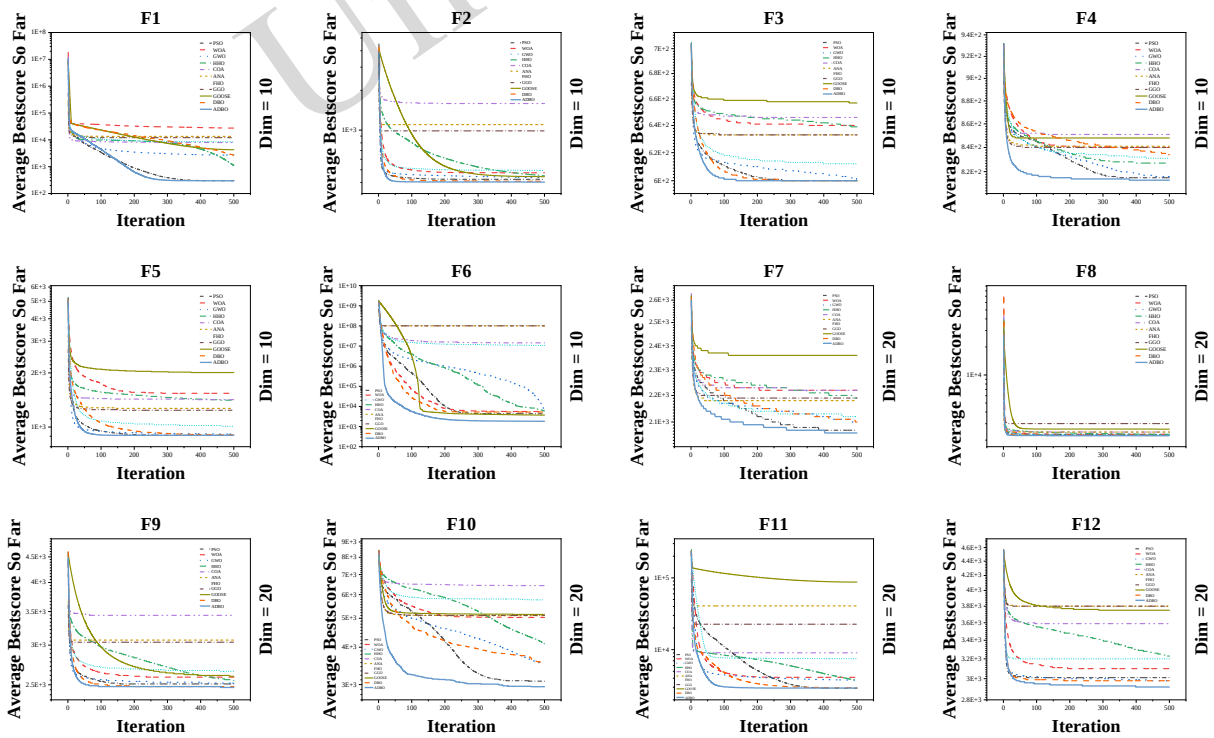


Fig. 9 CEC 2022 convergence curve

5.7 Wilcoxon Rank Sum Test

To validate ADBO's effectiveness, Wilcoxon Rank Sum tests assessed statistical differences between ADBO and other algorithms on 10/20-dimensional CEC 2022 and 30/100-dimensional CEC 2017 test sets (α = 0.05, 30 independent runs). Results report p-values, with "+/−/=" indicating ADBO's performance frequency relative to others. Most p-values < 0.05 confirm significant performance differences, validating ADBO's superior convergence. Detailed results are presented in Tables 11–14 (Supplementary Materials), with box plots in Figs. 10–11 visualizing performance distributions.

Figs. 10–11 display box plots for CEC 2022 (10D/20D, F1–F6/F7-F12) and CEC 2017 (30D/100D, F1–F12/F16–F26) test sets. Across all dimensions (10, 20, 30, 100), ADBO exhibits higher median values with narrower interquartile ranges, indicating reliable near-optimal solutions. Compared to other algorithms, ADBO shows lower variation, highlighting its superior stability and reduced performance variability. Fewer outliers demonstrate its ability to avoid suboptimal solutions, ensuring result reliability.

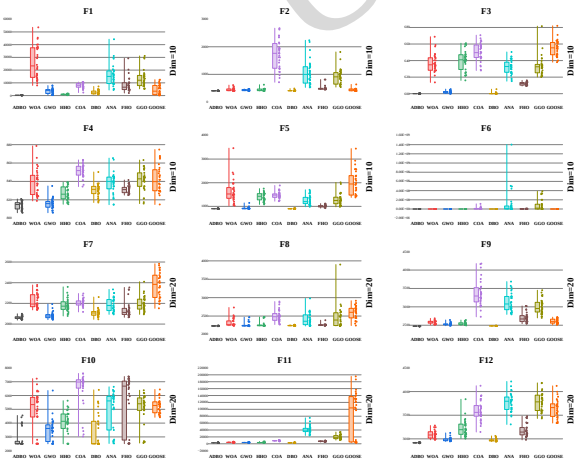


Fig. 10 CEC 2022 box plot

Tables 11 and 12 show that ADBO outperforms the compared algorithms in 10-dimensional and 20-dimensional experiments, with only minor differences from GWO on one specific function. Tables 13 and 14 further indicate that while DBO outperforms ADBO on select functions, ADBO leads across most functions. Emerging algorithms, such as HLOA, FHO, and GOOSE, outperform ADBO on specific functions, but these advantages are not universal. Conversely, ADBO exhibits consistent stability and superior global search capabilities across diverse conditions.

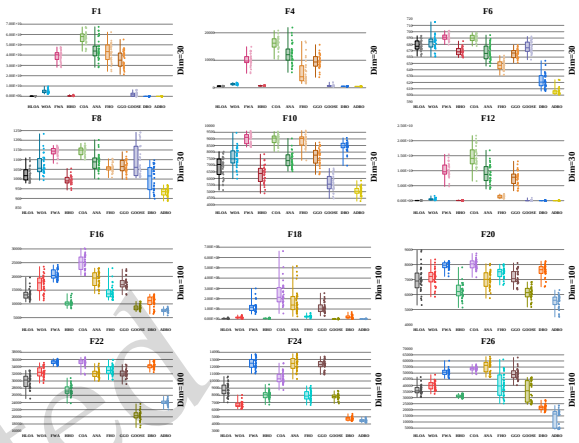


Fig. 11 CEC 2017 box plot

5.8 Friedman Test

The Friedman test, a non-parametric method for comparing multiple algorithms, was applied to evaluate ADBO against other algorithms on 30/100-dimensional CEC 2017 and 10/20-dimensional CEC 2022 test sets (Fig. 12). Results reveal that ADBO achieves the smallest average rank, outperforming the competitors and validating the improved algorithm's effectiveness.

The experimental results presented in Tables 15 and 16 (Supplementary Materials) show that ADBO achieves the lowest average rank in 10D/20D CEC 2022 and 30D CEC 2017 test sets, demonstrating its superior robustness. While HHO achieves the lowest average rank in CEC 2017 100D, ADBO maintains high rankings across test sets and dimensions, underscoring its overall superiority in multidimensional optimization.

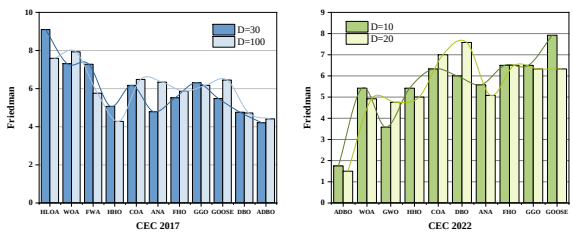


Fig. 12 Friedman test results

Unedited

Based on the experimental verification of ADBO's optimization performance, this paper further analyzes its theoretical complexity against 12 comparative algorithms to comprehensively evaluate the technical practicability. The algorithm's performance not only depends on the optimization ability, but also needs to take into account the computational efficiency, and the time complexity of ADBO is $O(N\cdot D\cdot T)$, which is consistent with most of the classical algorithms (where N = population size, D = dimension, T = number of iterations) such as DBO, PSO, GWO, HLOA, WOA, ANA, GGO, GOOSE, and so on, which indicates that the computational cost of ADBO is comparable to that of mainstream algorithms.

The complexity of some algorithms varies: the complexity of FWA is increased to $O(N\cdot(M+K)\cdot D\cdot T)$ due to the need to compute the exploding spark (M) and the mutating spark (K), which significantly increases the computational cost in high-dimensional scenarios; the complexity of HHO is $O(N\cdot(T+T\cdot D+1))$, that of COA is $O(N\cdot D\cdot(1+5T/2))$, and that of FHO is $O(3N\cdot D\cdot T)$. These complexities can be significantly higher than ADBO's when the number of iterations or the problem dimension is large.

In summary, ADBO exhibits better optimization performance while maintaining computational efficiency comparable to that of the classical algorithms, achieving a balance between efficiency and accuracy, and possessing stronger engineering practicality.

6 Optimization of Reed-Muller Logic Circuits

While most IC designs rely on Boolean logic with established automated methods, RM logic offers superior performance for specific circuits, significantly reducing their size, speed, and power consumption compared to Boolean logic (Romanov, 2024).

6.1 Background

Area optimization is critical in IC design, where XNOR/OR and XOR/AND RM logic yield more compact circuits than Boolean logic, reducing gate count and circuit area. The polarity of RM logic circuits significantly impacts complexity. Exhaustive search is feasible only for small-to-medium circuits due to the exponential growth of the polarity space in large designs. Swarm intelligence algorithms—such as GA, the Binary Adaptive Bacterial Foraging Algorithm (BABFA) (Zhou et al., 2021), and the Multi-strategy Wolf Pack Algorithm (MWPA) (Zhou et al., 2023)—address this by enabling global search for optimal polarities in high-dimensional spaces.

This paper experimentally evaluates eight algorithms (PSO, DE, GWO, GOOSE, ANA, GGO, FHO, ADBO) on MCNC benchmark circuits to assess their effectiveness in RM logic area optimization under fixed and mixed polarities. Experimental settings include a population size of 30, 10 independent runs, and a maximum of 50 iterations.

6.2 XNOR/OR-based FPRM Circuit Area Optimization

Any logic function with n variables corresponds to 2^n different polarity expansions. The FPRM circuit based on XNOR/OR expression is defined by Eq. (11).

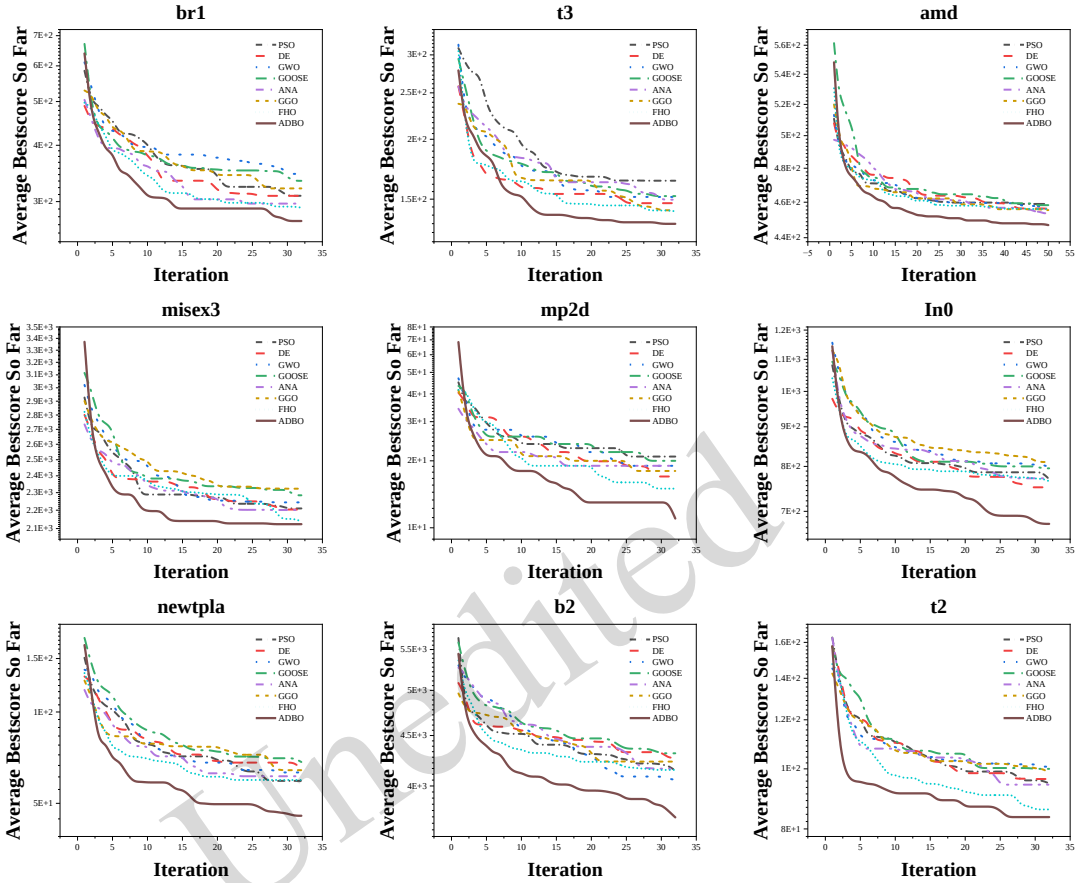
$$f=\odot\prod_{i=0}^{2^n-1}(d_i+s_i)\,, \tag{11}$$

where $\odot\prod$ denotes the same-or operation, s_i is the contingency term, and $d_i\in\{0,1\}$. When d_i is equal to 0, it means that the i-th or term in the expansion exists, and, vice versa, that the i-th or term does not exist. The fitness function of the FPRM circuit based on XNOR/OR area optimization can be expressed in terms of the circuit area model as shown in Eq. (12).

$$\begin{aligned}A_{FPRM}&=\sum_{i=0}^{2^n-1}((\overline{d_i})\sum_{i=0}^{n-1}\overline{i_k}-1)+\sum_{i=0}^{2^n-1}\overline{d_i}\\&=\sum_{i=0}^{2^n-1}\overline{d_i}\sum_{i=0}^{n-1}\overline{i_k}-1\end{aligned}\,, \tag{12}$$

Table 17 (Supplementary Materials) shows FPRM logic circuit area optimization results, with selected convergence curves in Fig. 13. ADBO identifies optimal area solutions for most circuits, while FHO matches performance only in br1 and br2. In br1, br2, t3, amd, and mp2d, ADBO achieves zero variance alongside optimal polarity discovery, demonstrating robust stability for engineering applications. Fig. 14 ranks ADBO first across all 11 XNOR/OR-optimized circuits, confirming its superior search capability, stability, and accuracy in complex optimization tasks.

Unedited



Unedited

Fig. 13 FPRM convergence curve

6.3 XOR /AND-based MPRM Circuit Area Optimization

The expansion of the MPRM circuit based on XOR/AND consists of multiple multi-input different-or terms and and terms. Any Boolean logic circuit with n input variables corresponds to 3^n different polarities, each corresponding to a unique MPRM logic expansion. The MPRM circuit based on XOR/AND expression is defined by Eq. (13).

$$f = \oplus\prod_{i=0}^{2^n-1} b_i \pi_i \ , \tag{13}$$

where $\oplus\prod$ denotes the different-or operation, π_i stands for the and term, b_i is the coefficient of the and term, and $b_i \in \{0,1\}$. If b_i is equal to 0, it means that the i-th and term in the expansion exists, and, vice versa, that the i-th and term does not exist.

The fitness function of the MPRM circuit based on XOR/AND area optimization can be expressed in

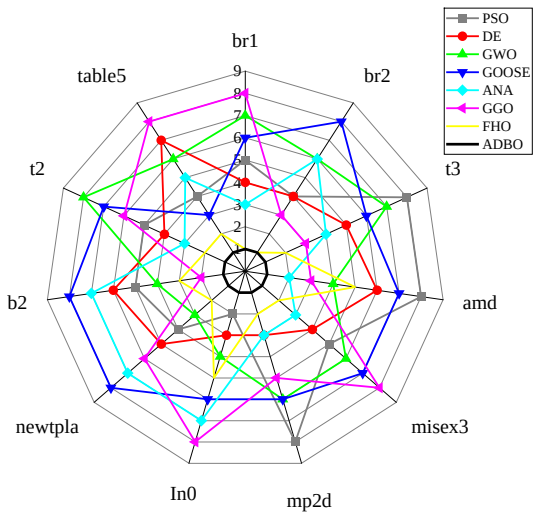


Fig. 14 FPRM ranking result

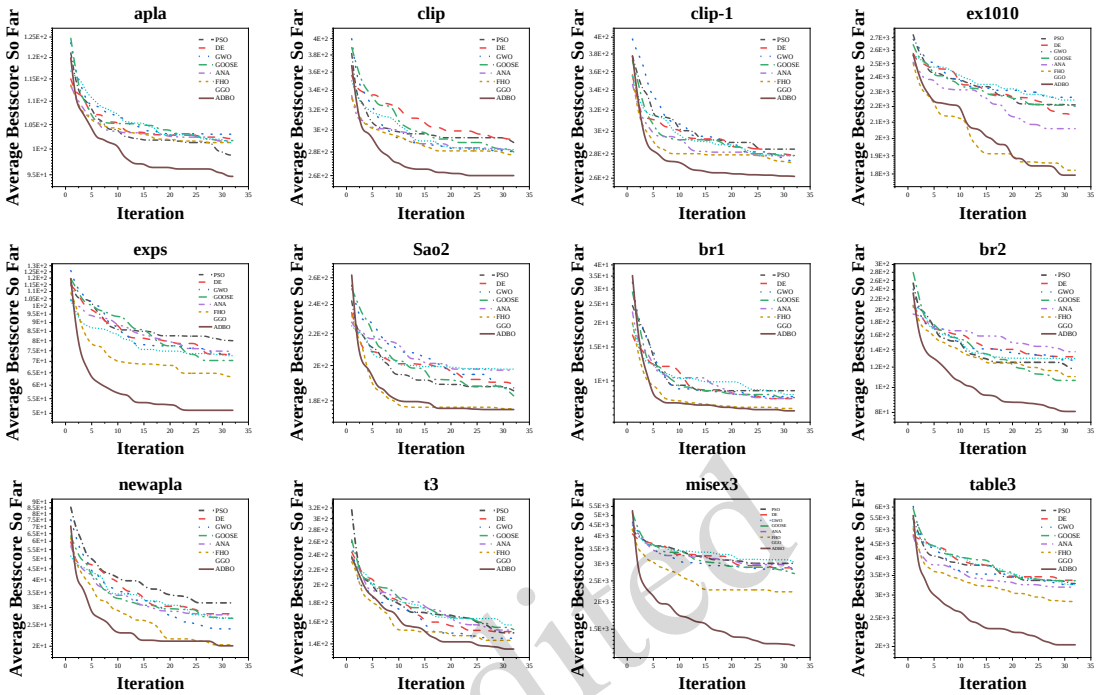


Fig. 15 MPRM convergence curve

Unedited

terms of the circuit area model as shown in Eq. (14).

$$A_{MPRM} = \lambda C_{AND} + \mu C_{XOR}. \quad (14)$$

where C_{AND} and C_{XOR} represent the costs of two-input terms and dissimilarity items, respectively, and λ and μ denote their respective quantities, with $C_{AND} = C_{XOR} = 1$.

Table 18 (Supplementary Materials) presents XOR/AND-based area optimization results for MPRM circuits, with selected convergence curves presented in Fig. 15. ADBO outperforms other algorithms in most circuits, identifying optimal polarity, besides the newapla test circuit, where GGO slightly surpasses it. Fig. 15 shows that ADBO demonstrates superior convergence to optimal values in apla, exps, br2, misex3, and table3 circuits. Fig. 16 presents the average ranking of ADBO against other algorithms across all circuits, with ADBO ranking first overall, validating its search efficiency and stability.

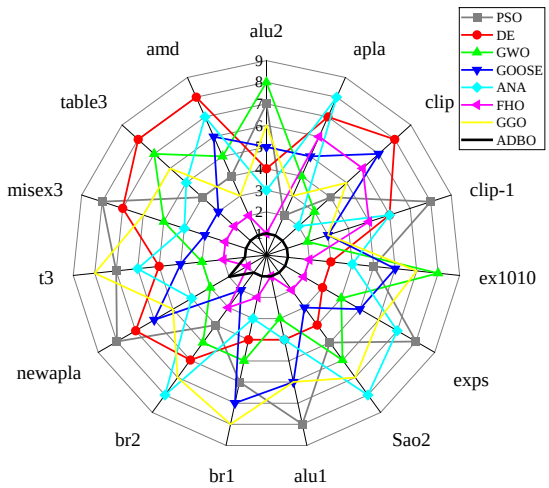


Fig. 16 MPRM ranking result

7 Conclusion

This paper presents ADBO, an improved algorithm with three key enhancements: dynamic adjustment of the convergence factor to balance global search and local exploitation; a greedy difference optimization strategy to enhance population diversity and global search while mitigating premature convergence; and an elastic annealing mechanism to

escape local optima via solution-position perturbation, boosting optimization stability.

ADBO's performance is evaluated on the CEC 2017 (29 functions across 10/30/50/100 dimensions) and CEC 2022 benchmark sets (12 functions in 10/30 dimensions), with results analyzed via Wilcoxon Rank Sum and Friedman tests. ADBO statistically outperforms 12 state-of-the-art algorithms—PSO, DE, FWA, WOA, HHO, COA, ANA, FHO, GGO, GOOSE, DBO, and GWO.

The experimental results demonstrate that ADBO effectively balances global and local search, avoids local optima, and delivers significant improvements across benchmark suites. Applied to area optimization of 11 FPRM and 15 MPRM circuits (MCNC benchmarks), ADBO rapidly identifies optimal polarities, outperforming traditional algorithms in terms of efficiency. Its robust global search and precise local exploitation confer high practical value for complex circuit design, offering efficient and reliable optimization solutions.

Contributors

Lixin MIAO, Zhenxue HE, and Xiaojun ZHAO conceptualized the research and set the objectives. Lixin MIAO collected data and drafted the manuscript, while Yijin WANG organized the paper's structure for clarity. Xiaodan ZHANG, Kui YU, and Zhisheng HUO provided critical feedback, enhancing the study's rigor. Limin XIAO contributed to final revisions and formatting. All authors engaged in discussions, ensuring a comprehensive and cohesive final product.

Conflict of interest

All the authors declare that they have no conflict of interest.

Data availability

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

References

Abdollahzadeh B, Gharehchopogh FS, Khodadadi N, et al., 2022. Mountain gazelle optimizer: a new nature-inspired metaheuristic algorithm for global optimization problems. *Adv Eng Softw*, 174:103282. https://doi.org/10.1016/j.advengsoft.2022.103282

Ahmadi-Javid A, 2011. Anarchic society optimization: a human-inspired method. IEEE Congress of Evolutionary Computation, p.2586-2592. https://doi.org/10.1109/CEC.2011.5949940

Ali JM, Hussain MA, Tade MO, et al., 2015. Artificial Intelligence techniques applied as estimator in chemical process systems–A literature survey. *Expert Syst Appl*, 42(14):5915-5931. https://doi.org/10.1016/j.eswa.2015.03.023

Amores D, Tanin E, Vasardani M, 2024. Flexible paths: A path planning approach to dynamic navigation. *IEEE Trans Intell Transp Syst*, 25(6):4795-4808. https://doi.org/10.1109/TITS.2023.3343490

Azizi M, Talatahari S, Gandomi AH, 2023. Fire Hawk Optimizer: A novel metaheuristic algorithm. *Artif Intell Rev*, 56(1):287-363. https://doi.org/10.1007/s10462-022-10173-w

Dehghani M, Trojovský P, 2021. Teamwork optimization algorithm: A new optimization approach for function minimization/maximization. *Sensors*, 21(13):4567. https://doi.org/10.3390/s21134567

Dehghani M, Montazeri Z, Trojovská E, et al., 2023. Coati Optimization Algorithm: A new bio-inspired metaheuristic algorithm for solving optimization problems. *Knowl-Based Syst*, 259:110011. https://doi.org/10.1016/j.knosys.2022.110011

El-Kenawy ESM, Khodadadi N, Mirjalili S, et al., 2024. Greylag Goose Optimization: nature-inspired optimization algorithm. *Expert Syst Appl*, 238:122147. https://doi.org/10.1016/j.eswa.2023.122147

Grefenstette JJ, 1993. Genetic algorithms and machine learning. Proc 6th Annual Conf on Computational Learning Theory, p.3-4. https://doi.org/10.1145/168304.168305

Hama Rashid DN, Rashid TA, Mirjalili S, 2021. ANA: Ant nesting algorithm for optimizing real-world problems. *Mathematics*, 9(23):3111. https://doi.org/10.3390/math9233111

Hamad RK, Rashid TA, 2024. GOOSE algorithm: a powerful optimization tool for real-world engineering challenges and beyond. *Evol Syst*, 15(4):1249-1274. https://doi.org/10.1007/s12530-023-09553-6

Han X, Meng ZL, Xia X, et al., 2024. Foundation intelligence for smart infrastructure services in transportation 5.0. *IEEE Trans Intell Veh*, 9(1):39-47. https://doi.org/10.1109/TIV.2023.3349324

He JC, Fu LH, 2024. Robot path planning based on improved dung beetle optimizer algorithm. *J Braz Soc Mech Sci Eng*, 46(4):235. https://doi.org/10.1007/s40430-024-04768-3

He ZX, Pan YH, Wang KJ, et al., 2021. Area optimization for MPRM logic circuits based on improved multiple disturbances fireworks algorithm. *Appl Math Comput*, 399:126008. https://doi.org/10.1016/j.amc.2021.126008

Heidari AA, Mirjalili S, Faris H, et al., 2019. Harris hawks optimization: Algorithm and applications. *Future Gener Comput Syst*, 97:849-872. https://doi.org/10.1016/j.future.2019.02.028

Ikram RMA, Dehrashid AA, Zhang BQ, et al., 2023. A novel swarm intelligence: cuckoo optimization algorithm (COA) and SailFish optimizer (SFO) in landslide susceptibility

Unedited

assessment. *Stoch Environ Res Risk Assess*, 37(5):1717-1743.
https://doi.org/10.1007/s00477-022-02361-5

Kaur S, Awasthi LK, Sangal AL, et al., 2020. Tunicate Swarm Algorithm: A new bio-inspired based metaheuristic paradigm for global optimization. *Eng Appl Artif Intell*, 90:103541.
https://doi.org/10.1016/j.engappai.2020.103541

Kaveh A, Dadras A, 2017. A novel meta-heuristic optimization algorithm: thermal exchange optimization. *Adv Eng Softw*, 110:69-84.
https://doi.org/10.1016/j.advengsoft.2017.03.014

Kennedy J, Eberhart R, 1995. Particle swarm optimization. Proc ICNN'95-Int Conf on Neural Networks, p.1942-1948.
https://doi.org/10.1109/ICNN.1995.488968

Kirkpatrick S, Gelatt CD Jr, Vecchi MP, 1983. Optimization by simulated annealing. *Science*, 220(4598):671-680.
https://doi.org/10.1126/science.220.4598.671

Li J, He Z, Wang SY, 2023. Advances in operation and finance in supply chains. *Int J Prod Econ*, 255:108707.
https://doi.org/10.1016/j.ijpe.2022.108707

Li LH, Liu LL, Shao YX, et al., 2023. Enhancing swarm intelligence for obstacle avoidance with multi-strategy and improved dung beetle optimization algorithm in mobile robot navigation. *Electronics*, 12(21):4462.
https://doi.org/10.3390/electronics12214462

Mirjalili S, Lewis A, 2016. The whale optimization algorithm. *Adv Eng Softw*, 95:51-67.
https://doi.org/10.1016/j.advengsoft.2016.01.008

Mirjalili S, Mirjalili SM, Lewis A, 2014. Grey wolf optimizer. *Adv Eng Softw*, 69:46-61.
https://doi.org/10.1016/j.advengsoft.2013.12.007

Peraza-Vázquez H, Peña-Delgado A, Merino-Treviño M, et al., 2024. A novel metaheuristic inspired by horned lizard defense tactics. *Artif Intell Rev*, 57(3):59.
https://doi.org/10.1007/s10462-023-10653-7

Rashedi E, Nezamabadi-Pour H, Saryazdi S, 2009. GSA: a gravitational search algorithm. *Inform Sci*, 179(13):2232-2248.
https://doi.org/10.1016/j.ins.2009.03.004

Ray T, Liew KM, 2003. Society and civilization: an optimization algorithm based on the simulation of social behavior. *IEEE Trans Evol Comput*, 7(4):386-396.
https://doi.org/10.1109/TEVC.2003.814902

Reynolds RG, Peng B, 2004. Cultural algorithms: modeling of how cultures learn to solve problems. 16th IEEE Int Conf on Tools with Artificial Intelligence, p.166-172.
https://doi.org/10.1109/ICTAI.2004.45

Romanov AM, 2024. Perfect mixed codes from generalized Reed–Muller codes. *Des Codes Cryptogr*, 92(6):1747-1759.
https://doi.org/10.1007/s10623-024-01364-3

Shao YX, He ZX, Zhou YH, et al., 2021. Area optimization of MPRM circuits based on M-AFSA. *J Beijing Univ Aeronaut Astronaut*, 49(3):693-701 (in Chinese).
https://doi.org/10.13700/j.bh.1001-5965.2021.0296

Shen QW, Zhang DM, Xie MS, et al., 2023. Multi-strategy enhanced dung beetle optimizer and its application in three-dimensional UAV path planning. *Symmetry*, 15(7):1432.
https://doi.org/10.3390/sym15071432

Storn R, Price K, 1997. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *J Glob Optim*, 11(4):341-359.
https://doi.org/10.1023/A:1008202821328

Sun F, Wang PJ, Yu HZ, 2013. Best polarity searching for ternary FPRM logic circuit area based on whole annealing genetic algorithm. IEEE 10th Int Conf on ASIC, p.1-4.
https://doi.org/10.1109/ASICON.2013.6812057

Tan Y, Zhu YC, 2010. Fireworks algorithm for optimization. 1st Int Conf on Advances in Swarm Intelligence, p.355-364.
https://doi.org/10.1007/978-3-642-13495-1_44

Wang LY, Cao QJ, Zhang ZX, et al., 2022. Artificial rabbits optimization: A new bio-inspired meta-heuristic algorithm for solving engineering optimization problems. *Eng Appl Artif Intell*, 114:105082.
https://doi.org/10.1016/j.engappai.2022.105082

Wang PJ, Li H, Wang ZH, 2010. MPRM expressions minimization based on simulated annealing genetic algorithm. IEEE Int Conf on Intelligent Systems and Knowledge Engineering, p.261-265.
https://doi.org/10.1109/ISKE.2010.5680868

Xue JK, Shen B, 2023. Dung beetle optimizer: A new meta-heuristic algorithm for global optimization. *J Supercomput*, 79(7):7305-7336.
https://doi.org/10.1007/s11227-022-04959-6

Zhou YH, He ZX, Liang XY, et al., 2021. Optimization of XNOR/OR circuit area based on BABFA. *J Beijing Univ Aeronaut Astronaut*, 48(10):2031-2039 (in Chinese).
https://doi.org/10.13700/j.bh.1001-5965.2021.0056

Zhou YH, He ZX, Zhang Y, et al., 2023a. Power optimization for mixed polarity Reed–Muller circuits based on multi-level adaptive memetic algorithm. *Int J Intell Syst*, 2023(1):3510001.
https://doi.org/10.1155/2023/3510001

Zhou YH, He ZX, Jiang JH, et al., 2023b. Fast area optimization approach for XNOR/OR-based fixed polarity Reed-Muller logic circuits based on multi-strategy wolf pack algorithm. *ACM Trans Des Autom Electron Syst*, 28(3):45.
https://doi.org/10.1145/3587818

Zhu F, Li GS, Tang H, et al., 2024. Dung beetle optimization algorithm based on quantum computing and multi-strategy fusion for solving engineering problems. *Expert Syst Appl*, 236:121219.
https://doi.org/10.1016/j.eswa.2023.121219

Unedited

List of supplementary materials

Fig. 5 Ablation experiment on CEC 2022 10-dimension
Fig. 6 CEC 2017 30-100 dimensions ranking results
Fig. 8 CEC 2022 10-20 dimensions ranking results
Table 5 CEC 2017 test results 10-dimension
Table 6 CEC 2017 test results 30-dimension
Table 7 CEC 2017 test results 50-dimension
Table 8 CEC 2017 test results 100-dimension
Table 9 CEC 2022 test results 10-dimension
Table 10 CEC 2022 test results 20-dimension
Table 11 CEC 2022 D = 10 Wilcoxon rank sum test
Table 12 CEC 2022 D = 20 Wilcoxon rank sum test
Table 13 CEC 2017 D = 30 Wilcoxon rank sum test
Table 14 CEC 2017 D = 100 Wilcoxon rank sum test
Table 15 Friedman test results on CEC 2022
Table 16 Friedman test results on CEC 2017
Table 17 Experimental results of FPRM on area optimization
Table 18 Experimental results of MPRM on area optimization

Unedited