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 an elastic 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 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 Aug. 28, 2025

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 DBO (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, thus improving the 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, helping the algorithm escape local optima and thereby improve solution quality and algorithm stability. The experimental results on the CEC 2017 and CEC 2022 benchmark function sets and MCNC benchmark circuits verify the effectiveness, superiority, and universality of ADBO.

Key words: Metaheuristic algorithm; Dung beetle optimizer; Convergence factor; Greedy difference optimization strategy; Elastic annealing mechanism
https://doi.org/10.1631/FITEE.2400967 **CLC number:** TP18

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 J 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

‡ Corresponding author
* Project supported by the National Natural Science Foundation of China (No. 62102130), the Central Government Guides Local Science and Technology Development Fund Project of China (No. 226Z0201G), the Natural Science Foundation of Hebei Province of China (Nos. F2020204003 and F2024204001), the Hebei Youth Talents Support Project of China (No. BJ2019008), the Science and Technology Research Projects of Higher Education Institutions in Hebei Province of China (No. QN2024138), the Basic Scientific Research Funds Research Project of Hebei Provincial Colleges and Universities of China (No. KY2022073), and the Hebei Province Higher Education Institution Scientific Research Project of China (No. QN2025192)
# Electronic supplementary materials: The online version of this article (https://doi.org/10.1631/FITEE.2400967) contains supplementary materials, which are available to authorized users
ORCID: Lixin MIAO, https://orcid.org/0009-0008-3213-6857; Zhenxue HE, https://orcid.org/0000-0001-7041-8582

uncertain problem spaces (Ali et al., 2015; Ikram et al., 2023).

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: evolutionary algorithms (e.g., genetic algorithm (GA) (Grefenstette, 1993), differential evolution (DE) (Storn and Price, 1997), and cultural algorithm (CA) (Reynolds and Peng, 2004)), which mimic biological evolution using mechanisms such as selection, mutation, and social learning; swarm-based algorithms (e.g., artificial rabbits optimization (ARO) (Wang LY et al., 2022), tunicate swarm algorithm (TSA) (Kaur et al., 2020), and mountain gazelle optimizer (MGO) (Abdollahzadeh et al., 2022)), which are inspired by collective animal behaviors and address multimodal or high-dimensional optimization; physics-based algorithms (e.g., simulated annealing (SA) mechanism (Kirkpatrick et al., 1983), thermal exchange optimization (TEO) (Kaveh and Dadras, 2017), and gravitational search algorithm (GSA) (Rashedi et al., 2009)), which leverage natural phenomena such as thermodynamics and gravitational interactions for global search; human-based algorithms (e.g., society civilization algorithm (SCA) (Ray and Liew, 2003), anarchic society optimization (ASO) (Ahmadi-Javid, 2011), and teamwork optimization algorithm (TOA) (Dehghani and Trojovský, 2021)), which 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 trade-offs 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 areas (He ZX 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 RM (FPRM, XNOR/OR-based) or mixed-polarity RM (MPRM, XOR/AND-based) circuits, 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 and 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 PJ 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: 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 the early stages followed by a 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 ADBO, which achieves higher convergence precision and global exploration capability than conventional and state-of-the-art swarm intelligence algorithms.

2 Related works

DBO, a swarm intelligence algorithm inspired by dung beetle behaviors (Xue and Shen, 2023), offers robust global search, rapid convergence, and simplicity, with applications in robotic path planning and unmanned aerial vehicle (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, Lévy distribution-based boundary handling, and dual crossover operators to balance search capabilities. He JC and Fu (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 Lévy flights to balance global and local search. Li LH 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 overreliance 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

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

3.1 Rolling dung beetles

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 the light source intensity affects the dung beetle path, the position of the rolling dung beetle is updated during the rolling process, which is expressed as

$$\begin{aligned}\boldsymbol{x}_i(t+1)&=\boldsymbol{x}_i(t)+A_1\cdot k\cdot\boldsymbol{x}_i(t-1)+A_2\cdot\Delta\boldsymbol{x},\\ \Delta\boldsymbol{x}&=\left|\boldsymbol{x}_i(t)-\boldsymbol{X}^{\mathrm{w}}\right|,\end{aligned}\tag{1}$$

where t denotes the current number of iterations, $\boldsymbol{x}_i(t)$ 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. A_2 represents a constant value within (0, 1), and the value of A_1 (assigned as 1 or −1 with a specified probability) indicates whether the dung beetle deviates from its path (A_1=1 means no deviation and A_1=−1 means deviation). $\boldsymbol{X}^{\mathrm{w}}$ denotes the global worst position, and $\Delta\boldsymbol{x}$ 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 dancing behavior, and the position update formula for the rolling dung beetle is given by

$$\boldsymbol{x}_i(t+1)=\boldsymbol{x}_i(t)+|\boldsymbol{x}_i(t)-\boldsymbol{x}_i(t-1)|\cdot\tan\theta,\tag{2}$$

where θ is the angle of deflection associated with [0, π], and $|\boldsymbol{x}_i(t)-\boldsymbol{x}_i(t+1)|$ represents the element-wise absolute difference of vectors $\boldsymbol{x}_i(t)$ and $\boldsymbol{x}_i(t+1)$. The position of the rolling dung beetle is not updated when θ= 0, π/2, or π.

3.2 Breeding dung beetles

A boundary selection strategy is proposed to simulate dung beetle behavior in choosing a safe region to lay eggs, with the optimal spawning area defined as

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

where $\mathbf{Lb}^*$ and $\mathbf{Ub}^*$ represent the lower and upper bounds of the optimal spawning region, respectively. $\boldsymbol{X}^*$ denotes the current local optimal position, and R (R=1−$t/T_{\max}$) is the nonlinear convergence factor with $T_{\max}$ 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 as

$$\boldsymbol{B}_i(t+1)=\boldsymbol{X}^*+\boldsymbol{b}_1\cdot\left(\boldsymbol{B}_i(t)-\mathbf{Lb}^*\right)+\boldsymbol{b}_2\cdot\left(\boldsymbol{B}_i(t)-\mathbf{Ub}^*\right),\tag{4}$$

where $\boldsymbol{B}_i(t)$ is the positional information of the i^{th} breeding ball at the t^{th} iteration, $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$ are two independent

random vectors of size $1\times D$, and D is the dimension of the optimization problem.

3.3 Foraging dung beetles

In nature, dung beetles select safe foraging areas; the boundaries of the optimal foraging area are defined as

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

where $\mathbf{Lb}^{\mathrm{b}}$ and $\mathbf{Ub}^{\mathrm{b}}$ represent the lower and upper bounds of the optimal foraging area, respectively, and $\boldsymbol{X}^{\mathrm{b}}$ represents the global optimal position. Therefore, the location update information of the foraging dung beetle is shown as

$$\boldsymbol{d}_i(t+1)=\boldsymbol{d}_i(t)+C_1\cdot\left(\boldsymbol{d}_i(t)-\mathbf{Lb}^{\mathrm{b}}\right)+\boldsymbol{C}_2\cdot\left(\boldsymbol{d}_i(t)-\mathbf{Ub}^{\mathrm{b}}\right),\tag{6}$$

where $\boldsymbol{d}_i(t)$ 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 $\boldsymbol{C}_2$ denotes a random vector, with all components of $\boldsymbol{C}_2$ belonging to (0, 1).

3.4 Stealing dung beetles

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

$$\boldsymbol{h}_i(t+1)=\boldsymbol{X}^{\mathrm{b}}+S\cdot\boldsymbol{g}\cdot\left(\left|\boldsymbol{h}_i(t)-\boldsymbol{X}^{*}\right|+\left|\boldsymbol{h}_i(t)-\boldsymbol{X}^{\mathrm{b}}\right|\right),\tag{7}$$

where $\boldsymbol{h}_i(t)$ denotes the location information of the i^{th} stealing dung beetle at the t^{th} iteration, $\boldsymbol{g}$ is a random vector of size $1\times D$ following a normal distribution, and S denotes a constant value.

4 Adaptive dung beetle optimizer

This paper proposes three improvement strategies to further improve the 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 nonlinear decay method dynamically adjusts the convergence factor's decay rate, prioritizing global exploration in early iterations and local exploitation in later stages to balance the optimization speed and quality. The dynamic decay factor strategy is defined as

$$\begin{cases}R_{\mathrm{g}}=R\cdot\exp(-G\cdot t),\\ R_{\mathrm{l}}=R\cdot\exp(-L\cdot(t-P\cdot T_{\max})),\\ R_{\mathrm{g}}\in[0.1,0.9],\\ R_{\mathrm{l}}\in[0.1,0.9\cdot\exp(-L\cdot P\cdot T_{\max})],\end{cases}\tag{8}$$

where G and L are the decay rates for the global and local search phases, respectively, and L=0.1. 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 the current optimal states accelerate global convergence but reduce population diversity, risking premature clustering at local optima and degrading the 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 cross over 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, which is defined as

$$\boldsymbol{X}_i(t+1)=\boldsymbol{X}_\alpha(t)+M\left(\boldsymbol{X}_\beta(t)-\boldsymbol{X}_\eta(t)\right),\tag{9}$$

where $\boldsymbol{X}_i(t+1)$ is the current mutation vector of the i^{th} stealing dung beetle, M is the mutation rate, α, β, and

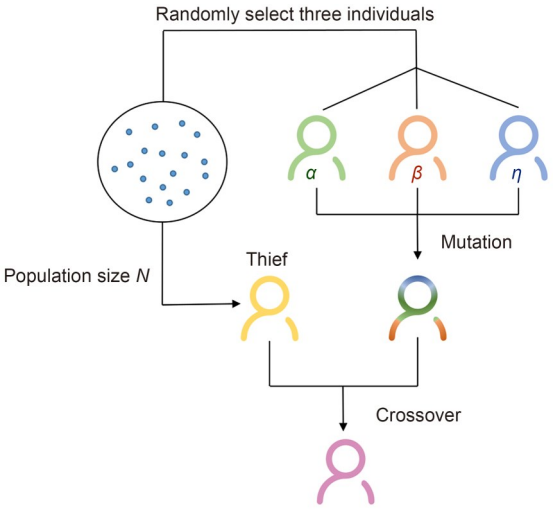

Fig. 1 Greedy difference optimization strategy

η are population-wide random numbers, and $\boldsymbol{X}_{\alpha}(t)$, $\boldsymbol{X}_{\beta}(t)$, and $\boldsymbol{X}_{\eta}(t)$ represent the location information of 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 $\mathbf{obj}_i(t+1)$: if rand(·)≤C (where C is the crossover rate) or j=randi(D), the j^{th} dimension (obj$_i$(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.

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 the cooling rate: 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 stabilizes the material in its ground state at room temperature.

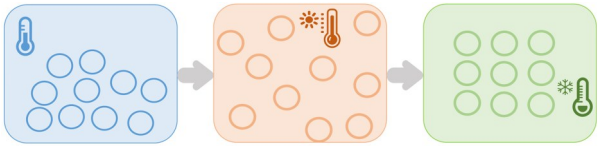

Fig. 2 Metal annealing model

The SA mechanism begins with a high initial temperature and probabilistically accepts suboptimal solutions as the temperature decreases, enabling a 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 the exploitation of better solutions during cooling to ensure global convergence.

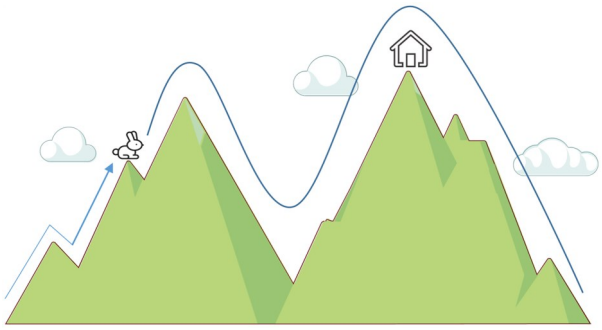

Fig. 3 Rabbit mountain climbing model

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 the end of each iteration, three randomly selected individuals undergo the strategy, with their fitness compared to that of the annealed individuals; a greedy rule retains those with better fitness, which is expressed as

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

where τ denotes the adaptive annealing rate and $I(t)$ represents the annealing temperature at the t^{th} iteration.

4.4 Algorithm description

ADBO initializes parameters and randomly initializes individual positions, updating dung beetle positions via steps 2–28, as shown in Algorithm 1. The key steps include step 13 with nonlinear convergence factors to balance global/local search, step 22 with a greedy difference optimization strategy to boost diversity, and step 25 with an elastic annealing mechanism to avoid local optima. The algorithm outputs the global optimal position $\boldsymbol{X}^{\text{b}}$ and optimal fitness value f_{b} after optimization.

Algorithm 1 ADBO

Input: Maximum number of iterations $T_{\max}$, population size N, and dimension D.

Output: Global optimal position $\boldsymbol{X}^{\text{b}}$ and optimal fitness value f_{b}.

1: Initialize the positions of the dung beetles
2: **While** $2 \leqslant t \leqslant T_{\max}$ **do**
3: Calculate the current best position and its fitness
4: **For** i=1: N **do**
5: **if** i is a rolling dung beetle **then**
6: α_1=rand(·)
7: **if** $\alpha_1 \leqslant 0.9$ **then**
8: Update the rolling dung beetle position by Eq. (1)
9: **else**
10: Update position by Eq. (2)
11: **end if**
12: **end if**
13: Nonlinear convergence factors are calculated by Eq. (8)
14: **if** i is a breeding dung beetle **then**
15: Update the breeding dung beetle position by Eqs. (3) and (4)
16: **end if**
17: **if** i is a foraging dung beetle **then**
18: Update the foraging dung beetle position by Eqs. (5) and (6)
19: **end if**
20: **if** i is a stealing dung beetle **then**
21: Update the stealing dung beetle 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 are randomly selected from the population for the elastic annealing mechanism, and greedy selection is used to retain the dominant individuals.
26: Update the optimal position and its fitness
27: t=t+1
28: **end while**
29: Return the global optimal position $\boldsymbol{X}^{\text{b}}$ and its fitness value f_{b}

Fig. 4 illustrates the flowchart of ADBO, and Table 1 presents the specific value ranges of various parameters in ADBO.

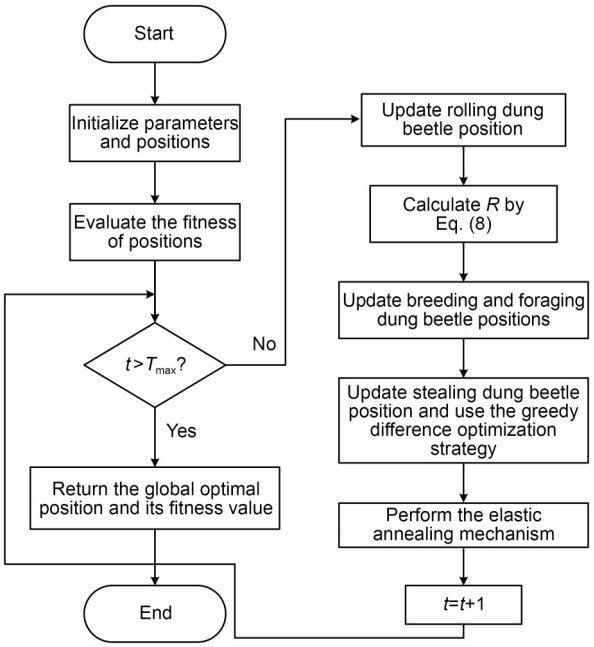

Fig. 4 Flowchart of ADBO

4.5 Time complexity analysis

Assuming a population size of N (with N/5+1 stealing dung beetles), a variable dimension D, and a maximum number of iterations $T_{\max}$, the DBO algorithm has a complexity of $O(N)$. For ADBO, time complexity analysis from its pseudocode results in contributions from two main steps: (1) position updates via variation and crossover operators for stealing dung beetles, with complexity $O((N/5+1)T_{\max})$; (2) annealing variation for three randomly selected individuals, with complexity $O(3T_{\max})$. Combining these, the overall complexity of ADBO is $O(N)+O((N/5+1)T_{\max})$. Despite this, ADBO maintains an overall time complexity comparable to that of the original algorithm ($O(N)$) while demonstrating superior performance.

Table 1 ADBO parameter settings

Parameter type	Parameter	Description	Range
User-defined parameters	N	Population size (typically a multiple of 5)	
	T_{max}	Maximum number of iterations	
	D	Dimension	
Custom parameters	G	Decay rate for the global search phase	$G\in[0.02, 0.04]$
	P	Threshold between the global and local search phases	$P\in[0.55, 0.75]$
	M	Mutation rate	$M\in[0.7, 0.9]$
	C	Crossover rate	$C\in[0.8, 1.1]$
	I	Initial annealing temperature	$I\in[2, 9]$

5 Experimental results and analysis

5.1 Configuration of the experimental environment

To ensure the rigor and fairness of the experiment, the following hardware and software environment was used:

Processor: Intel® Core™ i5-1035G1 CPU @1.00 GHz 1.19 GHz.

Memory: 8 GB 3733 MHz.

Graphics: NVIDIA GeForce MX250.

Simulation: MATLAB R2022b.

Operating system: Windows 10 (64-bit).

5.2 Experimental parameter settings

To evaluate the performance of ADBO, 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 number of 500 iterations, and 100 independent runs per function. The mean, standard deviation, and ranking of the solution results for each test function are recorded, ensuring fair comparisons under identical initial conditions to enhance the experimental reliability and repeatability.

A total of 12 algorithms are compared with ADBO, including the particle swarm optimization (PSO) (Kennedy and Eberhart, 1995), the 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 hawks 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. The parameter settings for these comparison algorithms are presented in Table 2. The specific configurations of benchmark 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 for comparison.

3. The Wilcoxon rank sum test and Friedman test are performed on FWA, WOA, GWO, HHO, COA, DBO, HLOA, ANA, GGO, FHO, and GOOSE.

Table 2 Parameter settings for 12 algorithms

Algorithm	Parameter(s)
PSO	The inertia weight reduces linearly from 0.9 to 0.1, and $n_1=n_2=2$
GWO	The convergence factor is linearly decreasing from 2 to 0 during iterations
HLOA	q=rand(·)
WOA	$Y=2(1-t/T_{max})$
FWA	Er=5, En=6, Mr=5, a=0.3, br=0.6
HHO	$E\in[-1, 1]$, Js=2(1−rand(·))
COA	I_0=round(1+rand(1, 1))
DBO	$k_0=0.1$, $A_2=0.3$, $S=0.5$
ANA	$PH_d=0.95$
FHO	Ir=unifrnd(0, 1, 1, 2)
GGO	$Z=1-(t/T_{max})^2$, $A=2v\cdot$rand(·)$-v$
GOOSE	alpha=$2-t/(T_{max}/2)$ SW=randi([5, 25], 1, 1)

1. In PSO, n_1 and n_2 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. In WOA, Y is a linearly decreasing factor that defines the algorithm's search range.

4. FWA includes the explosion radius (Er), explosion count (En), variable sparks (Mr), and explosion limit factors (a and br).

5. In HHO, E is a random number in [−1, 1], representing prey escape strength, and Js is the jump strength.

6. In COA, I_0 is randomly generated.

7. In DBO, k_0 is the deflection coefficient constant, which falls within (0, 0.2], A_2 represents a constant value within (0, 1), and S denotes a constant value.

8. In ANA, PH_d is the pheromone decay rate.

9. In FHO, Ir is a randomly generated value influencing Firehawk behavior (e.g., attractiveness, step size, and movement control).

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

11. In GOOSE, alpha falls within (0, 2) and decreases significantly with each iteration, and the stone weight SW ranges from 5 g to 25 g.

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

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: decay rate for the global search phase (G), threshold between the global and local search phases (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 the 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.

Table 3 Orthogonal experimental design table

Group	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

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

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. ADBO is compared with DBO, incorporating these three strategies to verify their effectiveness and stability. We conduct 30 independent trials on 12 functions from the CEC 2022 benchmark set, record the optimal values achieved by each algorithm for each function, and calculate the mean and variance for statistical analysis and comparison. Additionally, convergence curves are plotted for all algorithms across different functions (Fig. S1 in the supplementary materials) to enable comprehensive performance evaluation, with detailed results presented in Table 4.

Table 4 Ablation experimental results

Fun	Metric	Optimal value				
		DBO	AD	AG	AE	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

Fun: function; AD: the dynamic decay factor strategy; AG: greedy difference optimization strategy; AE: elastic annealing mechanism; Ave: average; Std: standard deviation

The experimental results demonstrate that algorithms combining the dynamic decay factor strategy, greedy difference optimization strategy, or elastic annealing mechanism outperform DBO. Compared to DBO, these fused algorithms exhibit 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.

5.5 Analysis of the results for the 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 validation. Each experiment involved 100 independent runs to compare the optimization performance and evaluate ADBO scalability in 10-dimensional (10D), 30-dimensional (30D), 50-dimensional (50D), and 100-dimensional (100D) spaces. The 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 shown in Tables S1–S4 in the supplementary materials.

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

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

In hybrid problem experiments (F11–F20), ADBO outperforms other algorithms in 10D, 30D, and 100D spaces by rapidly converging to global optima with low variance and stable results. While it is slightly less effective than DBO on F13/F15 in the 50D space,

ADBO remains robust in handling complex hybrid problems.

ADBO excels on composite functions (F23–F29), rapidly converging to optimal results in four different dimensional spaces via its elastic annealing mechanism, which enhances escape from local optima. Despite being slightly outperformed by GOOSE and DBO in 30D and 100D F22 cases and marginally lagging DBO in 50D F30 cases, 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. S2 in the supplementary materials shows the average rank of all algorithms on 30D and 100D CEC 2017 test sets, with ADBO achieving the best average rank and thus demonstrating superior overall performance compared to the other algorithms. Fig. 5 illustrates the convergence curves of different algorithms on F1 and F3–F29 in 10D, 30D, 50D, and 100D spaces. ADBO excels on single-peak and simple multipeak functions, rapidly converging to near-global optima and 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, the fast convergence, efficient search, and adaptability of ADBO underscore its competitiveness across diverse complex optimization problems.

5.6 Analysis of the results for the 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 10D and 20-dimensional (20D) CEC 2022 benchmark test sets are shown in Tables S5 and S6 in the supplementary materials.

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

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

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

5.6.2 CEC 2022 convergence curve analysis

Fig. S3 in the supplementary materials presents the average rank of all algorithms on 10D and 20D CEC 2022 test sets. ADBO achieves the best average rank across all functions in the 10D space, demonstrating superior optimization performance. In the 20D space, ADBO maintains strong performance with excellent average ranks, highlighting its adaptability to multidimensional problems.

Fig. 6 presents the convergence curves of different algorithms in 10D and 20D spaces. All algorithms converge rapidly to global optima with high accuracy and robustness. For the composite function F11, DBO and ADBO exhibit comparable convergence speeds and accuracies, indicating similar optimization capabilities. However, results show that ADBO slightly outperforms DBO in terms of search capability, enabling higher-quality solutions for certain composite functions and demonstrating superiority in complex optimization tasks.

5.7 Wilcoxon rank sum test

To validate ADBO's effectiveness, Wilcoxon rank sum tests were used to assess statistical differences between ADBO and other algorithms on 10D and 20D CEC 2022 test sets and 30D and 100D CEC 2017 test sets (significance level α_0=0.05, 30 independent runs). The results report p-values, with "+/−/="

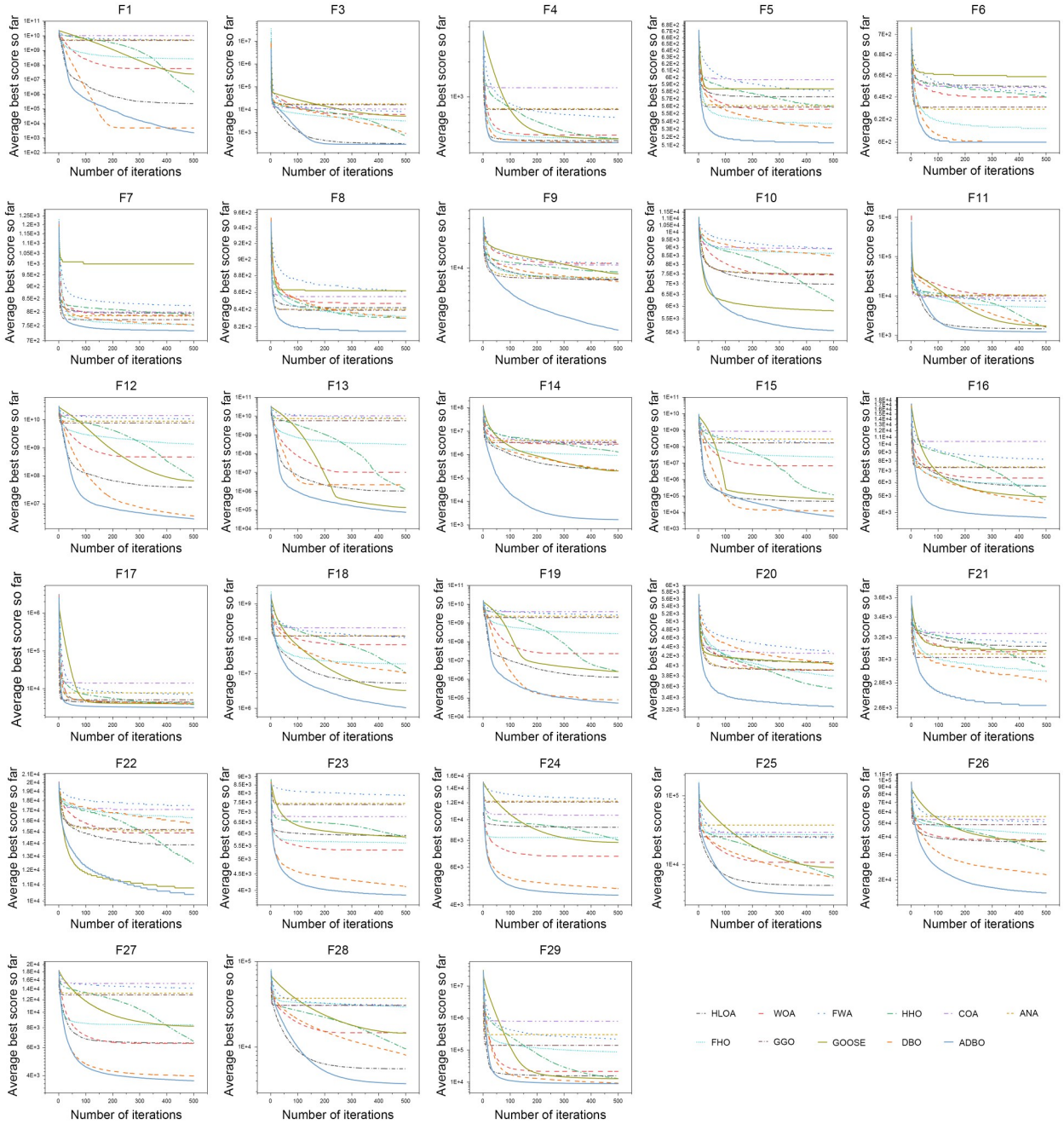

Fig. 5 Convergence curves of different algorithms on the CEC 2017 test set (F1 and F3–F29). F1 and F3–F8, F9–F15, F16–F22, and F23–F29 were conducted in 10D, 30D, 50D, and 100D spaces, respectively

indicating the ADBO performance frequency relative to the others. Most p-values <0.05 confirm significant performance differences, validating ADBO's superior convergence. The detailed results are presented in Tables S7–S10 in the supplementary materials.

Figs. 7 and 8 display box plots for the CEC 2022 (F1–F12) and CEC 2017 (12 functions) test sets, respectively. Across all dimensional spaces (10D, 20D, 30D, and 100D), ADBO exhibits higher median values with narrower interquartile ranges, indicating reliable near-optimal solutions. Compared to other algorithms, ADBO shows smaller variation, highlighting its superior stability and reduced performance variability. Fewer outliers demonstrate its ability to avoid suboptimal solutions, ensuring result reliability.

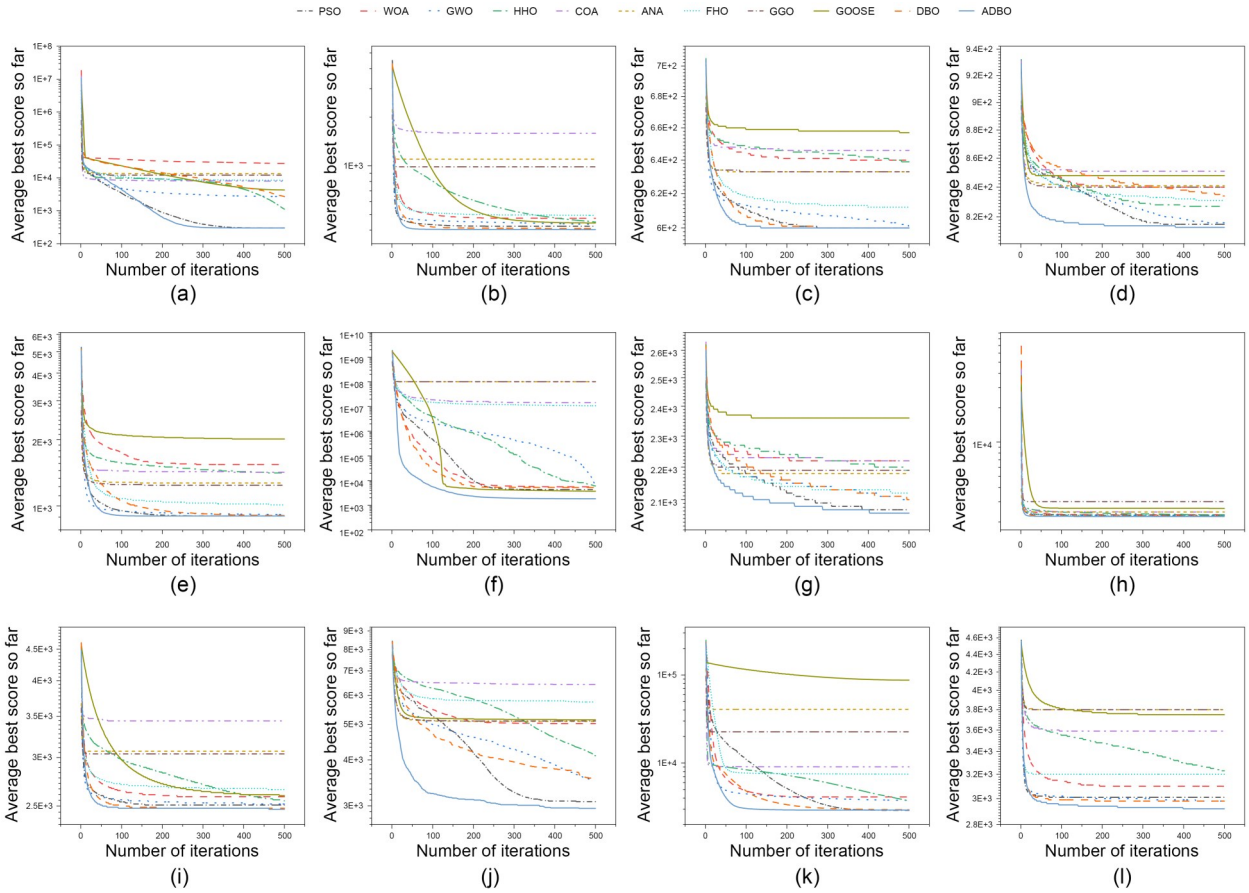

Fig. 6 Convergence curves of different algorithms on different CEC 2022 test sets: (a) F1; (b) F2; (c) F3; (d) F4; (e) F5; (f) F6; (g) F7; (h) F8; (i) F9; (j) F10; (k) F11; (l) F12. (a)–(f) and (g)–(l) were conducted in 10D and 20D spaces, respectively

Tables S7 and S8 in the supplementary materials show that ADBO outperforms the compared algorithms in 10D and 20D spaces, with only minor differences from GWO on one specific function.

Tables S9 and S10 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.

5.8 Friedman test

The Friedman test, a non-parametric method for comparing multiple algorithms, was applied to evaluate ADBO against other algorithms on 30D and 100D CEC 2017 test sets and 10D and 20D CEC 2022 test sets (Fig. 9). The results reveal that ADBO achieves the best average rank, outperforming the competitors and validating the improved algorithm's effectiveness.

The experimental results presented in Tables S11 and S12 in the supplementary materials show that ADBO achieves the best average rank in 10D/20D CEC 2022 and 30D CEC 2017 test sets, demonstrating its superior robustness. While HHO achieves the best average rank in CEC 2017 100D, ADBO maintains high rankings in test sets and dimensions, underscoring its overall superiority in multidimensional optimization.

Based on the experimental verification of ADBO's optimization performance, we further analyze 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

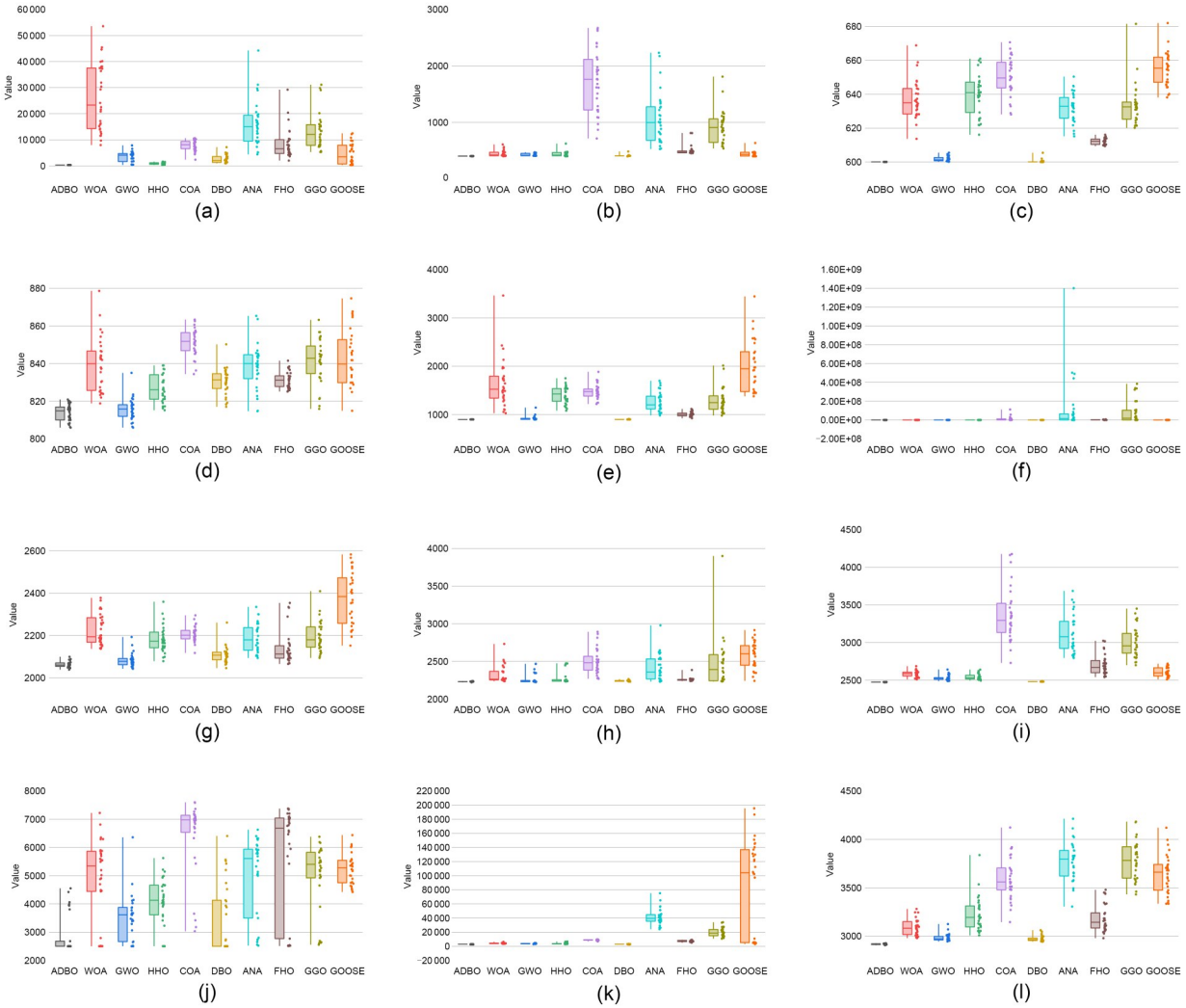

Fig. 7 Box plots for the 10D and 20D CEC 2022 test sets: (a) F1; (b) F2; (c) F3; (d) F4; (e) F5; (f) F6; (g) F7; (h) F8; (i) F9; (j) F10; (k) F11; (l) F12. (a)–(f) and (g)–(l) were conducted in 10D and 20D spaces, respectively. References to color refer to the online version of this figure

complexity of ADBO is $O(N{\cdot}D{\cdot}T)$, which is consistent with most of the classical algorithms, such as DBO, PSO, GWO, HLOA, WOA, ANA, GGO, and GOOSE, indicating 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 $O(N{\cdot}(M_c{+}K){\cdot}D{\cdot}T)$ due to the need to compute the exploding spark (M_c) 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 greater than those of ADBO 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 integrated circuit (IC) designs rely on Boolean logic with established automated methods, RM logic offers superior performance for specific circuits, significantly reducing their size and power consumption while shortening their runtime compared to Boolean logic (Romanov, 2024).

Fig. 8 Box plots for the 30D and 100D CEC 2017 test sets: (a) F1; (b) F4; (c) F6; (d) F8; (e) F10; (f) F12; (g) F16; (h) F18; (i) F20; (j) F22; (k) F24; (l) F26. (a)–(f) and (g)–(l) were conducted in 30D and 100D spaces, respectively. References to color refer to the online version of this figure

6.1 Background

Area optimization is critical in IC design, where XNOR/OR and XOR/AND RM logics yield more compact circuits than Boolean logic, reducing the 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 issue by enabling a 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. The experimental settings include a population size of 30, 10 independent runs, and a maximum number 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 the XNOR/OR expression is defined as

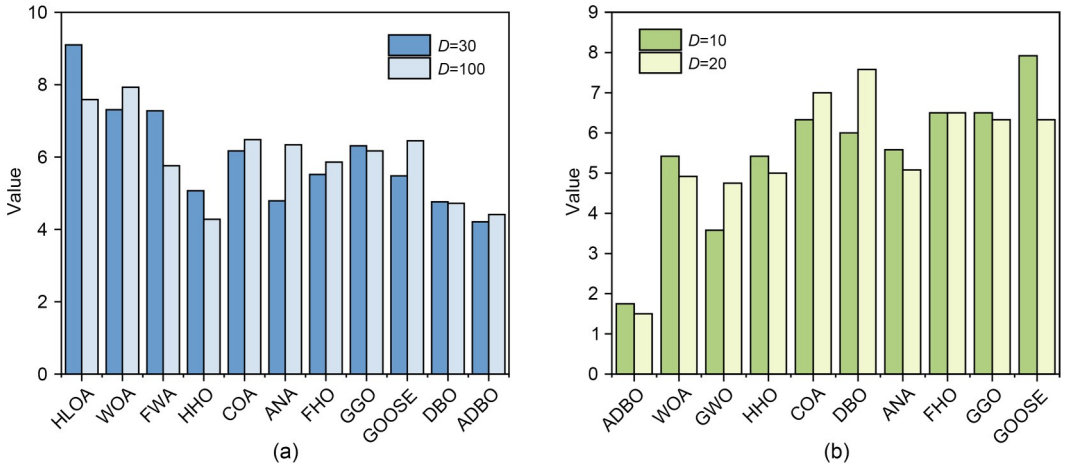

Fig. 9 Friedman test results of ADBO against other algorithms: (a) 30D and 100D CEC 2017 test sets; (b) 10D and 20D CEC 2022 test sets

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

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; otherwise, 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, which is shown as

$$\begin{aligned}A_{\text{FPRM}}&=\sum_{i=0}^{2^n-1}\left(\overline{d}_i\sum_{i=0}^{n-1}\overline{i}_k-1\right)+\sum_{i=0}^{2^n-1}\overline{d}_i\\&=\sum_{i=0}^{2^n-1}\overline{d}_i\sum_{i=0}^{n-1}\overline{i}_k-1.\end{aligned} \tag{12}$$

Table S13 in the supplementary materials shows the FPRM circuit area optimization results, with convergence curves in Fig. 10. 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. 11 ranks ADBO first across all 11 XNOR/OR-based FPRM circuits, confirming its superior search capability, stability, and accuracy in complex optimization tasks.

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 as

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

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; otherwise, 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 terms of the circuit area model, which is shown as

$$A_{\text{MPRM}}=\lambda C_{\text{AND}}+\mu C_{\text{XOR}}, \tag{14}$$

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 $C_{\text{AND}}=C_{\text{XOR}}=1$.

Table S14 in the supplementary materials presents XOR/AND-based area optimization results for MPRM circuits, with convergence curves presented in Fig. 12. ADBO outperforms other algorithms in most circuits, identifying the optimal polarity, besides the newapla circuit, where GGO slightly surpasses it. Fig. 12 shows that ADBO demonstrates superior convergence to optimal values in the apla, exps,

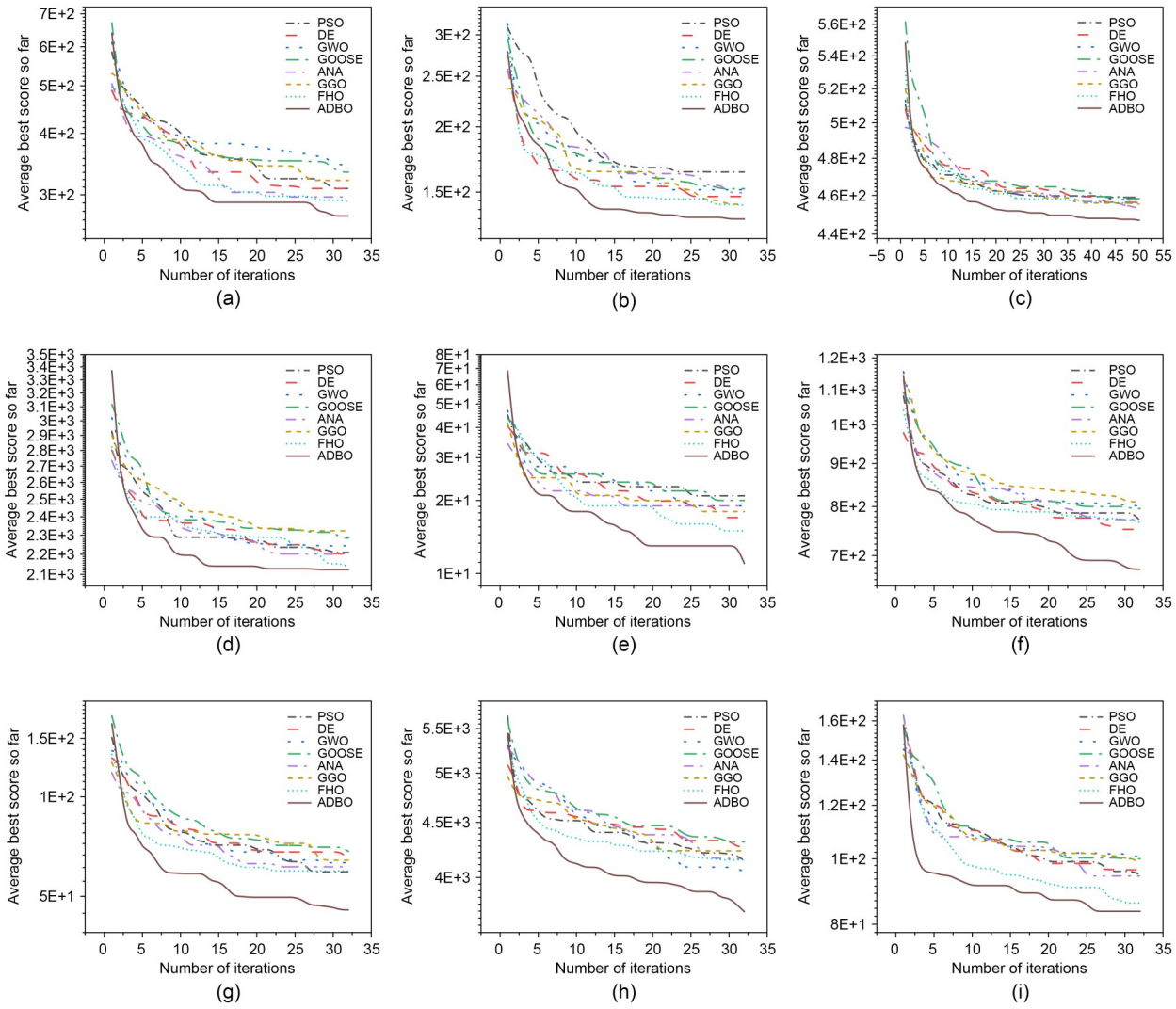

Fig. 10 Convergence curves of different XNOR/OR-based FPRM circuits: (a) br1; (b) t3; (c) amd; (d) misex3; (e) mp2d; (f) In0; (g) newtpla; (h) b2; (i) t2

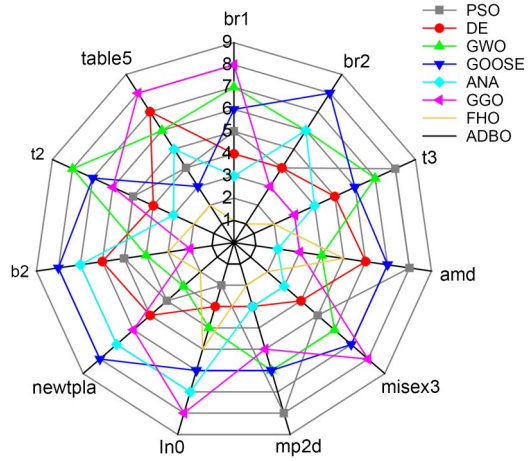

Fig. 11 Ranking result of eight algorithms in 11 XNOR/OR-based FPRM circuits

br2, misex3, and table3 circuits. Fig. 13 presents the average rank of ADBO against other algorithms across all circuits, with ADBO ranking first overall, validating its search efficiency and stability.

7 Conclusions

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 increase population diversity and global search while mitigating premature convergence, and an elastic annealing mechanism to escape

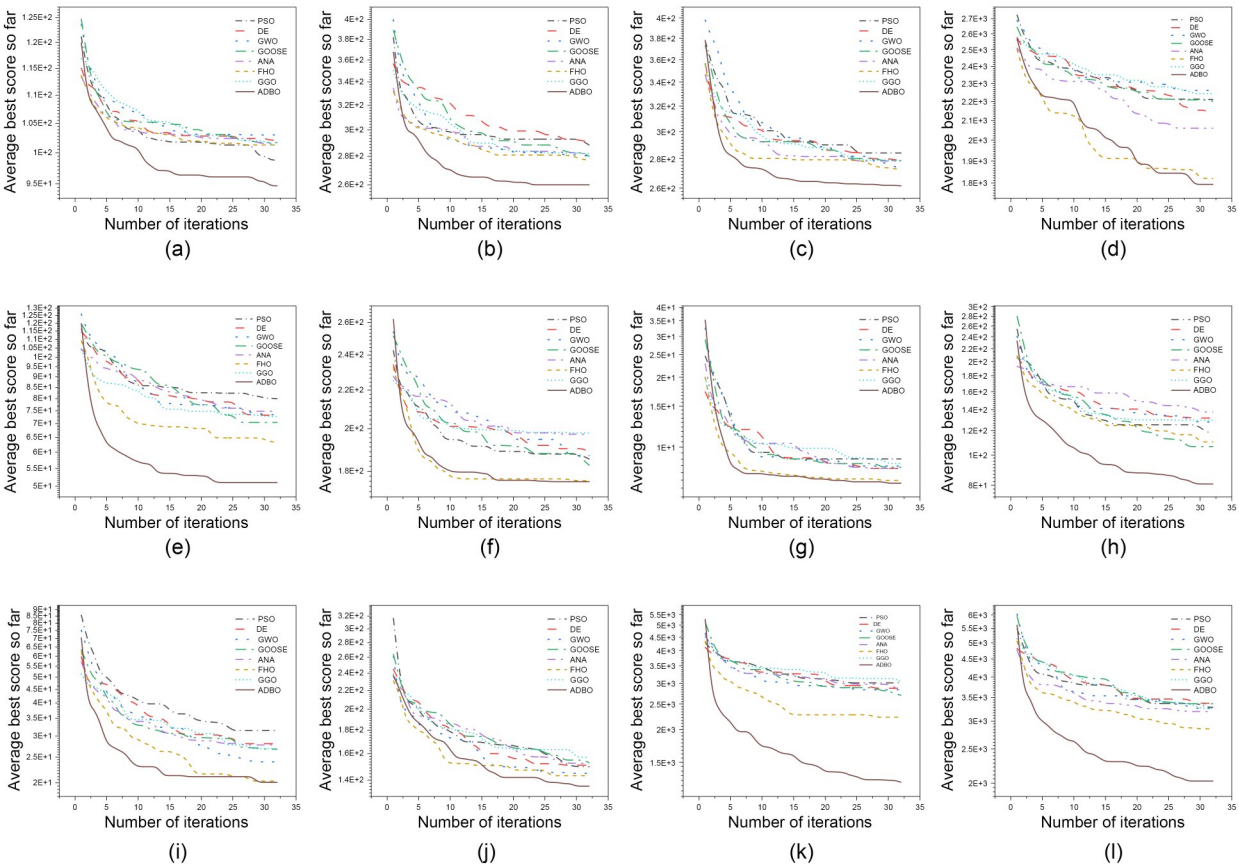

Fig. 12 Convergence curves of different XOR/AND-based MPRM circuits: (a) apla; (b) clip; (c) clip-1; (d) ex1010; (e) exps; (f) Sao2; (g) br1; (h) br2; (i) newapla; (j) t3; (k) misex3; (l) table3

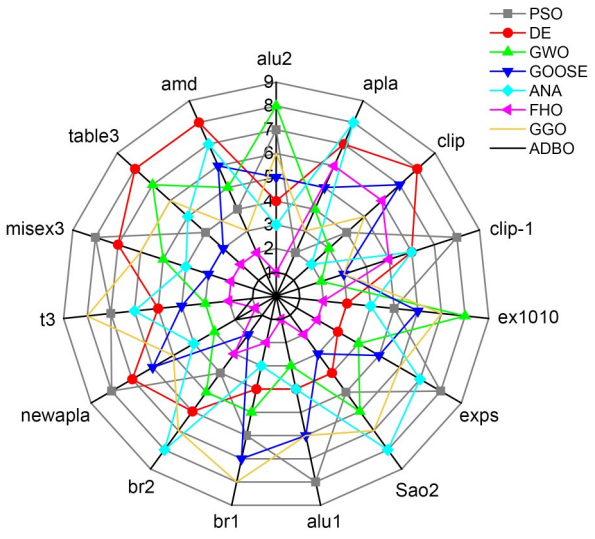

Fig. 13 Ranking result of eight algorithms in 15 XOR/AND-based MPRM circuits

local optima via solution–position perturbation, boosting optimization stability.

The performance of ADBO is evaluated on the CEC 2017 benchmark sets (29 functions in 10D/30D/50D/100D spaces) and the CEC 2022 benchmark sets (12 functions in 10D/20D spaces), with results analyzed via the Wilcoxon rank sum test and the Friedman test. ADBO statistically outperforms 12 state-of-the-art algorithms—PSO, HLOA, 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. When applied to the area optimization of 11 FPRM circuits 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 paper, while Yijin WANG organized the paper for clarity. Xiaodan ZHANG, Kui YU, and Zhisheng HUO provided critical feedback, enhancing the rigor of the study. Limin XIAO contributed to final revisions and formatting. All the authors engaged in discussions, ensuring the comprehensiveness and cohesiveness of the final product.

Conflict of interest

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

Data availability

Data sharing is not available for this paper as no new data were created or analyzed.

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. *Fut 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 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 metaheuristic 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 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 metaheuristic 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, Jiang JH, et al., 2023. 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

List of supplementary materials

Fig. S1 Ablation experimental results

Fig. S2 Average rank of all algorithms on 30D and 100D CEC 2017 test sets

Fig. S3 Average rank of all algorithms on 10D and 20D CEC 2022 test sets

Table S1 10D CEC 2017 test set results

Table S2 30D CEC 2017 test set results

Table S3 50D CEC 2017 test set results

Table S4 100D CEC 2017 test set results

Table S5 10D CEC 2022 test set results

Table S6 20D CEC 2022 test set results

Table S7 Wilcoxon rank sum test results on 10D CEC 2022 test sets

Table S8 Wilcoxon rank sum test results on 20D CEC 2022 test sets

Table S9 Wilcoxon rank sum test results on 30D CEC 2017 test sets

Table S10 Wilcoxon rank sum test results on 100D CEC 2017 test sets

Table S11 Friedman test results on CEC 2022 test sets

Table S12 Friedman test results on CEC 2017 test sets

Table S13 Experimental results of FPRM on area optimization

Table S14 Experimental results of MPRM on area optimization