

Article

Comprehensive Learning Fungal Growth Optimizer for Numerical Optimization and Reservoir Production Optimization

Mingyang Gong¹, Zhenyu Song^{2,*}, Xiaonan Zhang¹ and Yi Tang¹¹ School of Geophysics and Petroleum Resources, Yangtze University, Wuhan 430100, China;

2021730024@yangtzeu.edu.cn (M.G.); 2021710289@yangtzeu.edu.cn (X.Z.); tangyi.stu@yangtzeu.edu.cn (Y.T.)

² School of Earth Sciences, Yangtze University, Wuhan 430100, China

* Correspondence: zysong@yangtzeu.edu.cn

Abstract

The Fungal Growth Optimizer (FGO) is a nature-inspired metaheuristic that simulates fungal colony behaviors, but its exploitation phase can lose search diversity when guidance is dominated by limited peer or global-best information. In this paper, we propose an enhanced variant called the Comprehensive Learning Fungal Growth Optimizer (CLFGO). We integrate a conditionally activated Comprehensive Learning (CL) strategy into the FGO framework. When a candidate solution stagnates, the strategy constructs a dimension-specific learning exemplar. This mechanism allows each dimension to learn from the personal best of a different peer, extending the original fungal growth model. CLFGO is therefore intended for high-dimensional, multimodal, hybrid, and composition landscapes in which the original FGO is prone to diversity loss, rather than as a universal replacement for all problem classes. This approach improves population diversity and reduces the risk of premature convergence. We evaluated CLFGO on 29 CEC2017 benchmark functions at 30 dimensions against nine metaheuristics under the same maximum-function-evaluation budget. CLFGO achieved the lowest mean error on 21 of 29 functions and attained a Friedman average rank of 1.5517. Furthermore, we applied CLFGO to a reservoir production optimization problem, where it obtained a mean Net Present Value of 9.97×10^8 USD, outperforming the compared algorithms in both solution accuracy and convergence stability.

Keywords: Fungal Growth Optimizer; Comprehensive Learning Strategy; production optimization; global optimization

1. Introduction

Optimization problems frequently arise in engineering design, economic planning, and scientific research. Classical optimization techniques, such as gradient descent and the simplex method, have well-established theoretical foundations but can struggle with high-dimensional, nonlinear, or discontinuous objective functions [1–6].

Metaheuristic algorithms are increasingly used for these complex optimization tasks [7,8]. As stochastic, population-based methods, they require no assumptions about differentiability and can search diverse regions of the solution landscape to balance exploration and exploitation [9–11]. Recent developments also include surrogate-assisted metaheuristics for computationally expensive applications [12,13].

Modern metaheuristics are typically categorized into Evolutionary Algorithms (EAs) and Swarm Intelligence (SI) algorithms [14]. EAs, such as the Genetic Algorithm (GA) [15]



Academic Editor: Chaoqun Li

Received: 2 May 2026

Revised: 19 May 2026

Accepted: 22 May 2026

Published: 27 May 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

and Differential Evolution (DE) [16], are based on principles of selection, crossover, and mutation. SI algorithms, such as Particle Swarm Optimization (PSO) [17,18] and Ant Colony Optimization (ACO) [19], model the collective intelligence of decentralized biological systems [20,21]. Contemporary strategies frequently hybridize these mechanisms to address complex optimization tasks [22,23].

The No Free Lunch (NFL) theorem for optimization [24] states that no single algorithm performs best on all possible problems. An algorithm that excels on one class of functions may underperform on another [25]. This principle motivates the ongoing development of specialized metaheuristics designed for specific problem characteristics, as aligning algorithmic mechanics with the target problem structure can yield significant practical benefits [26–28].

This study focuses on enhancing the Fungal Growth Optimizer (FGO), a bio-inspired algorithm that models candidate solutions as a network of hyphae and simulates fungal colony behaviors such as tip growth, chemotropism, branching, and spore germination. This multi-phase search process provides broad exploratory coverage [29,30]. However, the guidance mechanism in FGO has a structural limitation: during exploitation, individuals move primarily toward the global best solution or randomly selected peers. Because learning sources are not adapted dynamically, the algorithm lacks a mechanism for systematic knowledge sharing across the population [31]. Consequently, FGO can rapidly lose diversity and stagnate at local optima in complex multimodal spaces [32,33].

This structural diagnosis also clarifies the problem classes for which CLFGO is theoretically expected to be useful. The proposed mechanism is most relevant to high-dimensional, multimodal, hybrid, and composition problems where useful search information may be distributed across different individuals and dimensions, and where a single dominant attractor can cause premature loss of population diversity. In contrast, we do not claim that CLFGO should be universally superior on simple unimodal, low-dimensional, or single-basin problems; the No Free Lunch theorem implies that such universal dominance is not expected.

To address this limitation, we propose the Comprehensive Learning Fungal Growth Optimizer (CLFGO). We integrate a dimension-wise Comprehensive Learning (CL) module into the baseline FGO framework to facilitate cooperative behavior. When an individual stagnates, the CL module constructs a tailored learning exemplar through tournament-based peer selection, enabling the individual to learn from multiple peers rather than a single global attractor. This conditional activation preserves the rapid exploitation capabilities of the original hyphal growth operators while introducing necessary diversity. We evaluated CLFGO on 29 CEC2017 benchmark functions against nine metaheuristics. Furthermore, we applied CLFGO to a high-dimensional reservoir production optimization problem under geological uncertainty.

The main contributions of this work are summarized as follows:

- We identify the single-source guidance tendency of FGO as a key structural reason for diversity loss and clarify that CLFGO targets problems with multimodal, hybrid, composition, or dimension-wise heterogeneous structures.
- We adapt the Comprehensive Learning strategy as a conditionally activated module in FGO, so stagnant individuals can build dimension-wise exemplars from multiple high-quality peers without replacing the original fungal growth operators.
- We revise the experimental protocol to use a strict maximum-function-evaluation budget and provide parameter settings, diagnostic convergence/diversity evidence, and time-space complexity analysis for fairer comparison.

The remainder of this paper is organized as follows. Section 2 describes the original FGO algorithm. Section 3 presents the proposed Comprehensive Learning strategy and the

CLFGO framework. Section 4 details the experimental setup and benchmark comparisons. Section 5 applies CLFGO to a reservoir production optimization case study. Section 6 concludes the paper.

2. Original Algorithm

The Fungal Growth Optimizer (FGO), proposed by Abdel-Basset et al. [32], is a nature-inspired metaheuristic that simulates the growth behaviors of fungal colonies. It is built on three core biological mechanisms: hyphal tip growth (comprising exploration and exploitation phases), hyphal branching, and spore germination.

2.1. Mathematical Model of FGO

The algorithm initializes a population of N candidate solutions (hyphae) and updates their positions through the following mechanisms.

1. Initial Population Generation: At the start, N hyphae (solutions) are randomly distributed within the D -dimensional search space bounded by lower ($\vec{S}_L$) and upper ($\vec{S}_U$) limits. This simulates the random germination of spores in a suitable environment.

$$\vec{S}_i = \vec{S}_L + \vec{r} \odot (\vec{S}_U - \vec{S}_L), \quad i = 1, 2, \dots, N, \quad (1)$$

where $\vec{r}$ is a vector of random numbers uniformly distributed in $[0, 1]$ and $\odot$ denotes the Hadamard product.

2. Hyphal Tip Growth: This mechanism models the elongation and directional change of a hypha. It is divided into an exploration phase and an exploitation phase, controlled by a normalized fitness probability p_i and a decaying exploration rate E_r . The growth rate E and direction $\vec{D}$ are calculated using the fitness of solutions and random peer selection.

$$\vec{S}_i^{t+1} = \begin{cases} \vec{S}_i^t + E \cdot \vec{D}, & \text{if } p_i < E_r \quad (\text{Exploration Phase I}) \\ \text{perform local chemotropism search,} & \text{otherwise} \quad (\text{Exploitation Phase I}) \end{cases} \quad (2)$$

where $E = e^F$ with $F = \frac{f_i}{\sum_{k=1}^N f_k} \cdot r_1 \cdot \mathcal{E}$, $\vec{D} = \vec{S}_{t_a} - \vec{S}_{t_c}$, and p_i is the normalized fitness of the i -th solution. $\mathcal{E}$ represents the environmental factor, r_1 is a uniformly distributed random number in $[0, 1]$, and $\vec{S}_{t_a}$, $\vec{S}_{t_c}$ are randomly chosen mutually exclusive peer individuals from the current population. The exploitation phase involves chemotropism, where a hypha moves toward either a randomly selected peer or the global best solution, with an additional exploratory step to avoid premature convergence.

3. Hyphal Branching: This behavior enhances exploration by generating a new solution (branch) from an existing one. The new position is influenced by the difference between two randomly chosen solutions and the difference between a random solution and the global best.

$$S_{i,j}^{t+1} = [R_j < r_3] \cdot S_{i,j}^t + (1 - [R_j < r_3]) \cdot \left(S_{i,j}^t + r_5 \cdot E_L \cdot D_{i,j}^{ep1} + (1 - r_5) \cdot E_L \cdot D_{i,j}^{ep2} \right), \quad (3)$$

where $D_{i,j}^{ep1} = S_{a,j}^t - S_{b,j}^t$, $D_{i,j}^{ep2} = S_{c,j}^t - S_j^*$, $E_L = 1 + e^{\frac{f_j}{\sum_{k=1}^N f_k}} \cdot [r_6 < r_7]$, and $[\cdot]$ is the Iverson bracket. Here, R_j, r_3, r_5, r_6 , and r_7 denote randomly generated numbers in $[0, 1]$, S_j^* represents the j -th dimension of the global best position, and $S_{a,j}^t, S_{b,j}^t, S_{c,j}^t$ are elements of three distinct randomly selected individuals.

4. Spore Germination: This mechanism re-initializes solutions in new areas to escape local optima. Initially, spores land in random positions (pure exploration). As the search progresses, they land between the global best and a random position, promoting exploitation.

$$S_{i,j}^{t+1} = [R_j < r_3] \cdot S_{i,j}^t + (1 - [R_j < r_3]) \cdot \left(\frac{\left(\left(\frac{t}{t_{\max}} \right) \cdot S_j^* + \left(1 - \frac{t}{t_{\max}} \right) \cdot S_{a,j}^t \right)}{2} + S_{b,j}^t \right. \\ \left. + S_g \cdot r_5 \cdot E \cdot \left| \frac{S_{c,j}^t + S_{a,j}^t + S_{b,j}^t}{3} - S_{i,j}^t \right| \right), \quad (4)$$

where S_g is randomly chosen as -1 or 1 , t is the current iteration, and $t_{\max}$ is the maximum number of iterations used to compute the normalized search progress.

2.2. Integrated Iteration

The complete FGO algorithm initializes a population of N hyphae using Equation (1). In each iteration, the algorithm first stochastically selects between executing the *Hyphal Tip Growth* loop and the *Branching-Germination* loop based on a random comparison. Within the selected loop, each individual is updated according to the corresponding mathematical model. After a new candidate position is generated and evaluated, the function-evaluation counter is increased by one. A greedy selection is then performed: the new position replaces the current one if and only if it yields improved fitness. This process repeats until the maximum number of function evaluations (MaxFEs) is reached, at which point the global best solution $\vec{S}^*$ is returned as the final output.

2.3. Pseudocode of the Original FGO

The pseudocode for the original FGO algorithm is presented in Algorithm 1, summarizing the initialization, main loop, and update procedures.

Algorithm 1 Original Fungal Growth Optimizer (FGO)

```

1: Input: Population size  $N$ , maximum function evaluations MaxFEs, problem dimension
    $D$ , bounds  $\vec{S}_L, \vec{S}_U$ 
2: Initialize  $N$  hyphae  $\vec{S}_i$  using Equation (1)
3: Evaluate fitness  $f(\vec{S}_i)$  and identify the global best  $\vec{S}^*$ 
4:  $FEs \leftarrow N, t \leftarrow 1$ 
5: while  $FEs < \text{MaxFEs}$  do
6:   Generate random numbers  $r_9, r_{10} \sim \text{rand}(0, 1)$ 
7:   if  $r_9 < r_{10}$  then ▷ Execute Hyphal Tip Growth loop
8:     for  $i = 1$  to  $N$  do
9:       if  $FEs \geq \text{MaxFEs}$  then
10:        break
11:      end if
12:      Compute exploration rate  $E_r$  and normalized fitness probability  $p_i$ 
13:      if  $p_i < E_r$  then
14:        Update  $S_i^{t+1}$  via Exploration Phase I (Equation (2))
15:      else
16:        Update  $S_i^{t+1}$  via Exploitation Phase I (Equation (2))
17:      end if
18:      Evaluate  $S_i^{t+1}$  and set  $FEs \leftarrow FEs + 1$ 
19:      Greedy selection: if  $f(S_i^{t+1}) < f(\vec{S}_i^*)$ , set  $\vec{S}_i^* \leftarrow S_i^{t+1}$ 
20:    end for
21:  else ▷ Execute Branching and Germination loop

```

Algorithm 1 *Cont.*

```

22:   for  $i = 1$  to  $N$  do
23:     if  $FES \geq \text{MaxFES}$  then
24:       break
25:     end if
26:     Generate random number  $r_7 \sim \text{rand}(0, 1)$ 
27:     if  $r_7 < 0.5$  then
28:       Update  $S_i^{t+1}$  via Hyphal Branching (Equation (3))
29:     else
30:       Update  $S_i^{t+1}$  via Spore Germination (Equation (4))
31:     end if
32:     Evaluate  $S_i^{t+1}$  and set  $FES \leftarrow FES + 1$ 
33:     Greedy selection: if  $f(S_i^{t+1}) < f(\vec{S}_i^t)$ , set  $\vec{S}_i^t \leftarrow S_i^{t+1}$ 
34:   end for
35: end if
36:   Update global best  $\vec{S}^*$  if a better solution is found
37:    $t \leftarrow t + 1$ 
38: end while
39: Output: Global best solution  $\vec{S}^*$ 

```

3. The Proposed CLFGO Algorithm

This section describes the core components of CLFGO and the integration of the Comprehensive Learning (CL) strategy into the FGO framework. The CL module provides stagnant individuals with multi-source, dimension-wise learning guidance to reduce the risk of premature convergence.

3.1. Comprehensive Learning Strategy (CLS)

The Comprehensive Learning Strategy (CLS) is inspired by dimension-wise multi-exemplar learning methods such as CL-PSO. We adapt this strategy as a conditionally activated mechanism within the FGO structure. It diversifies search directions by allowing individuals to learn from different peers for each dimension, reducing reliance on a single global attractor.

3.1.1. Triggering Condition and Exemplar Vector

The strategy is conditionally activated when an individual i is identified as *stagnant*, meaning that its personal best solution has not improved over a predefined number of consecutive iterations. For a stagnant individual, a new *learning exemplar vector* is constructed as:

$$\mathbf{e}_i = [e_i(1), e_i(2), \dots, e_i(D)], \quad (5)$$

where D is the problem dimensionality. Each element $e_i(j)$ specifies the index of the exemplar individual from which particle i learns in the j -th dimension.

3.1.2. Learning Probability

Each individual is assigned a learning probability P_{c_i} that controls its propensity to learn from others versus itself. This probability is defined by a monotonically increasing function based on the individual's index within the sorted population:

$$P_{c_i} = a + b \times \frac{\exp\left(10 \cdot \frac{i-1}{N}\right) - 1}{\exp(10) - 1}. \quad (6)$$

Here, N is the population size, i is the index of the current individual, and a and b are the lower and upper bounds for the learning probability, respectively. We set $a = 0$ and $b = 0.5$ as conservative CLPSO-derived bounds [34], not as a newly optimized universal setting. This choice restricts the maximum probability of external exemplar learning to 0.5, so each individual still retains substantial influence from its own historical best position while stagnant individuals can receive cross-dimensional guidance.

3.1.3. Dimension-Wise Exemplar Selection

For each dimension $j \in \{1, 2, \dots, D\}$, the exemplar index $e_i(j)$ is determined through a stochastic tournament process. A random number $\text{rand} \in [0, 1]$ is drawn. If $\text{rand} < P_{c_i}$, the individual learns from external peers: two distinct individuals k_1 and k_2 (with $k_1, k_2 \neq i$) are randomly selected, and the one with the better historical best fitness is chosen as the exemplar for that dimension. Otherwise (i.e., $\text{rand} \geq P_{c_i}$), the individual retains its own historical best position. This mechanism is formally expressed as:

$$e_i(j) = \begin{cases} \arg \min_{k \in \{k_1, k_2\}} \text{pf}it(k), & \text{if } \text{rand} < P_{c_i}, \\ i, & \text{if } \text{rand} \geq P_{c_i}. \end{cases} \quad (7)$$

3.1.4. Safeguard Mechanism

To prevent degenerate learning where an individual learns exclusively from itself, a safeguard is applied. After constructing $\mathbf{e}_i$, the algorithm checks if:

$$e_i(j) = i, \quad \forall j \in \{1, 2, \dots, D\}. \quad (8)$$

If Equation (8) holds, a random dimension j' is selected, and a distinct individual $k_3 \neq i$ is randomly chosen, such that $e_i(j') = k_3$. This ensures that every stagnant individual learns from at least one external peer.

3.2. Implementation Framework of CLFGO

CLFGO integrates the CLS as a conditionally activated module within the original FGO workflow, as illustrated in Figure 1. During the main iterative loop, the algorithm monitors for stagnant individuals. For stagnant individual i , the CLS module is invoked before the standard FGO update. It generates a new candidate position using the exemplar vector $\mathbf{e}_i$. If this CL-generated position yields improved fitness, it is accepted. The individual then executes the standard FGO operators from this updated position. This mechanism introduces new search directions for trapped solutions without replacing the original bio-inspired dynamics.

3.3. Pseudocode of the Proposed CLFGO

Algorithm 2 presents the pseudocode for CLFGO. The procedure integrates the CLS as an inner loop executed before the standard FGO update operators. At each iteration, the algorithm checks whether each individual has stagnated (Lines 7–8). For a stagnant individual, the CLS module constructs an exemplar vector (Lines 9–18) according to the learning probability P_{c_i} (Equation (6)) and the tournament selection rule (Equation (7)). The safeguard mechanism (Lines 19–21) ensures that at least one dimension learns from an external peer. A new candidate position is generated from the exemplar vector and evaluated via greedy selection (Lines 22–24). The individual then executes the standard FGO operators. Each objective evaluation produced by either the CL module or the standard FGO operators is counted toward the same MaxFEs budget. This structure injects diversity into trapped solutions while preserving the original search dynamics.

Algorithm 2 Comprehensive Learning Fungal Growth Optimizer (CLFGO)

```

1: Input: Population size  $N$ , maximum function evaluations MaxFEs, dimension  $D$ ,
   bounds  $\vec{S}_L, \vec{S}_U$ , stagnation threshold  $m$ , learning bounds  $a, b$ 
2: Initialize  $N$  hyphae  $\vec{S}_i$  using Equation (1); evaluate fitness; identify  $\vec{S}^*$ 
3: Initialize personal best positions  $pbest_i \leftarrow \vec{S}_i$ , stagnation counters  $sc_i \leftarrow 0, \forall i$ 
4: Compute learning probabilities  $P_{c_i}$  using Equation (6),  $\forall i$ 
5:  $FEs \leftarrow N, t \leftarrow 1$ 
6: while  $FEs < \text{MaxFEs}$  do
7:   for  $i = 1$  to  $N$  do ▷ CLS Module
8:     if  $FEs \geq \text{MaxFEs}$  then
9:       break
10:    end if
11:    if  $sc_i \geq m$  then ▷ Individual  $i$  is stagnant
12:      for  $j = 1$  to  $D$  do ▷ Build exemplar vector  $e_i$ 
13:        Generate  $\text{rand} \sim \text{rand}(0, 1)$ 
14:        if  $\text{rand} < P_{c_i}$  then
15:          Randomly select  $k_1, k_2 \neq i$ 
16:           $e_i(j) \leftarrow \arg \min_{k \in \{k_1, k_2\}} \text{fit}(k)$  ▷ Tournament selection
17:        else
18:           $e_i(j) \leftarrow i$  ▷ Learn from own best
19:        end if
20:      end for
21:      if  $e_i(j) = i, \forall j$  then ▷ Safeguard check
22:        Randomly select dimension  $j'$  and individual  $k_3 \neq i$ 
23:         $e_i(j') \leftarrow k_3$ 
24:      end if
25:      Construct candidate  $\vec{S}_i^{\text{CL}}$ : for each dimension  $j$ , set  $S_{i,j}^{\text{CL}} \leftarrow pbest_{e_i(j),j}$ 
26:      Evaluate  $\vec{S}_i^{\text{CL}}$  and set  $FEs \leftarrow FEs + 1$ 
27:      if  $f(\vec{S}_i^{\text{CL}}) < f(\vec{S}_i^t)$  then  $\vec{S}_i^t \leftarrow \vec{S}_i^{\text{CL}}; sc_i \leftarrow 0$ 
28:    end if
29:  end for ▷ Standard FGO Update (Algorithm 1, Lines 6–16)
30:  Execute Hyphal Tip Growth or Branching-Germination loop on all individuals
   while respecting MaxFEs
31:  Perform greedy selection for each individual; update  $pbest_i$  and  $sc_i$  accordingly
32:  Update global best  $\vec{S}^*$  if a better solution is found
33:   $t \leftarrow t + 1$ 
34: end while
35: Output: Global best solution  $\vec{S}^*$ 

```

3.4. Complexity Analysis

For the original FGO, the arithmetic update cost per iteration is $O(N \cdot D)$, and the cumulative update cost over T iterations is $O(T \cdot N \cdot D)$, excluding the objective-function evaluation cost. In CLFGO, let q_t denote the number of stagnant individuals at iteration t . The CL module constructs dimension-wise exemplar vectors only for these stagnant individuals, yielding an additional $O(q_t \cdot D)$ arithmetic cost and at most q_t additional objective evaluations. Since $q_t \leq N$, the worst-case auxiliary update cost remains $O(N \cdot D)$ per iteration, and all additional objective evaluations are counted under the same MaxFEs stopping budget used by the compared algorithms.

The space complexity also remains linear in $N \cdot D$, but CLFGO requires additional storage beyond the original FGO. Specifically, FGO stores the current population and related fitness values, whereas CLFGO additionally stores personal-best positions, exemplar indices, stagnation counters, and learning probabilities. Table 1 summarizes the resulting time and memory requirements.

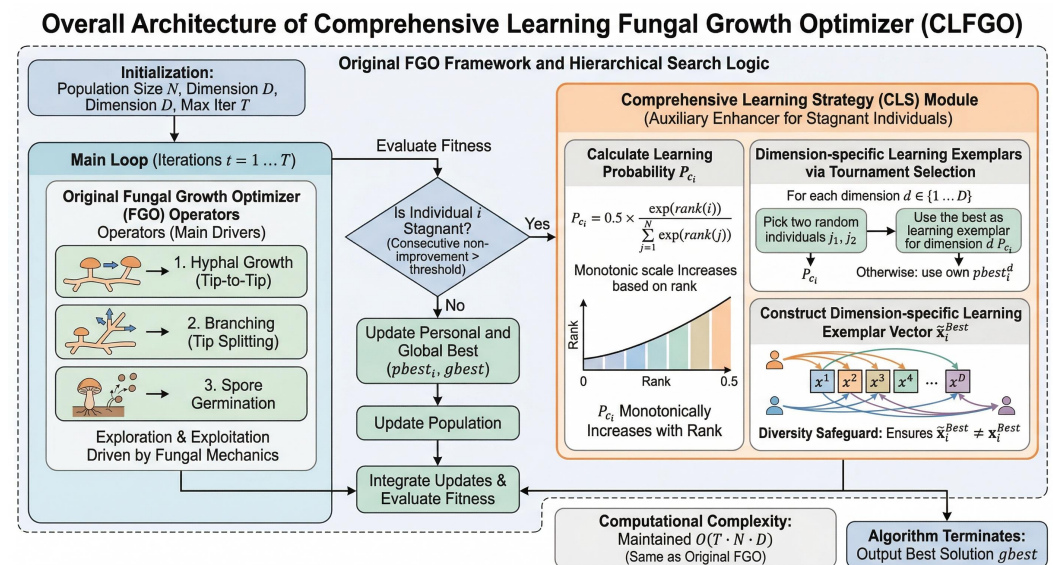


Figure 1. Flowchart of the proposed CLFGO algorithm.

Table 1. Time and space complexity comparison of FGO and CLFGO.

Algorithm	Main Additional Operations	Space Requirement
FGO	Population update and greedy selection under the evaluation budget; update arithmetic is $O(N \cdot D)$ per iteration.	Population matrix, candidate solutions, fitness values, and global best; $O(N \cdot D)$.
CLFGO	FGO operations plus exemplar construction for q_t stagnant individuals; auxiliary arithmetic is $O(q_t \cdot D)$ and worst-case $O(N \cdot D)$ per iteration.	FGO storage plus personal bests ($N \cdot D$), exemplar indices ($N \cdot D$), stagnation counters (N), and learning probabilities (N); still $O(N \cdot D)$ with a larger constant factor.

To quantify the runtime overhead of the proposed CL module, we profiled CLFGO, FGO, and the nine compared algorithms on the unimodal function F1. A full 29-function, 30-run runtime profile would be expensive, so the profiling was run over 10 independent trials under the same settings (population size $N = 30$, dimension $D = 30$, and MaxFEs = 300,000) in MATLAB R2022b on an Intel Core i7-10700 CPU at 2.90 GHz. The total execution times are summarized in Table 2.

Table 2. Execution time (seconds) of competing algorithms for F1 over 10 independent runs.

Algorithm	Runtime over 10 Runs (s)
CLFGO (Ours)	23.4
FGO	19.8
DE	16.5
MGO	22.1
HGS	29.8
PO	37.4
BA	26.2
GWO	17.1
SMA	33.5
SCA	18.9

The results in Table 2 show that CLFGO adds a modest runtime overhead (approximately 18%) compared with FGO. This cost comes from stagnation checking, personal-best tracking, and dimension-wise exemplar construction, which is activated only for stagnant

individuals. In this setting, CLFGO is close to MGO and faster than HGS, SMA, and PO. These timings are consistent with the stated $O(T \cdot N \cdot D)$ update cost and indicate that the CL module adds limited overhead relative to the full run.

4. Experimental Results and Analysis

This section reports the empirical evaluation of CLFGO against nine metaheuristics on CEC2017 benchmark functions and a reservoir production optimization problem.

4.1. Experimental Setup

All algorithms were implemented and evaluated under the same MATLAB execution environment. The population size was $N = 30$, and the problem dimensionality was $D = 30$. The search terminated at the same maximum-function-evaluation budget for every algorithm, namely $\text{MaxFEs} = 10,000 \times D = 300,000$ for the 30-dimensional CEC2017 experiments. Any objective evaluation generated by the CL module was counted toward this budget; therefore, CLFGO was not allowed extra evaluations beyond the competing algorithms. We performed 30 independent runs for each function using uniformly distributed random seeds and recorded the mean and standard deviation of the best fitness values.

The internal parameters for all compared algorithms were set to the default values recommended in their original publications (Table 3). For CLFGO, the learning probability bounds were set to $a = 0$ and $b = 0.5$ (Equation (6)). The upper bound of 0.5 allows individuals to retain information from their own historical best positions. The stagnation threshold was set to $m = 5$.

Table 3. Parameter settings for the compared algorithms.

Algorithm	Parameter Settings
CLFGO	$a = 0, b = 0.5, m = 5$
FGO	$\mathcal{M} = 0.6, E_p = 0.7, \mathcal{R} = 0.9$
DE	$F = 0.5, CR = 0.2$
MGO	$w = 2, d_1 = 0.2, rec_num = 10, divide_num = \text{dim} / 4$
HGS	$l = 0.03$
PO	$\gamma = 1.5, c = 0.2, F : S : C : O = 1 : 1 : 1 : 1$
BA	$f_{\min} = 0, f_{\max} = 2, \alpha = 0.9, \gamma = 0.9$
GWO	$a = [2, 0]$ (decreases linearly)
SMA	$a = [2, 0], vb = [-2, 2], E = [0, 2]$
SCA	$a = 2$ (decreases linearly)

4.2. Sensitivity Analysis of the Stagnation Threshold

The stagnation threshold m controls when the CL strategy is activated. A small value triggers external learning too frequently and may disturb useful FGO exploitation, whereas a large value delays the rescue of stagnant individuals. To test this parameter more broadly, five CLFGO variants with stagnation thresholds $m \in \{1, 3, 5, 7, 10\}$ were compared on the entire CEC2017 benchmark suite. The statistical results obtained across 30 dimensions are summarized in Table 4.

The results in Table 4 show that CLFGO with $m = 5$ gives the best overall outcome among the tested settings. When the stagnation threshold is very small ($m = 1$), comprehensive learning is triggered too often and disrupts the baseline FGO search, giving a mean rank of 4.10 and 19 losses. Larger thresholds ($m = 7$ and $m = 10$) remain close to $m = 5$, with Wilcoxon tests showing no statistically significant differences ($\approx$) on most benchmark functions. We therefore use $m = 5$ as the default because it gives the highest best count and the lowest average rank among the tested values. This sensitivity analysis covers unimodal, multimodal, hybrid, and composition functions, and supports using $m = 5$ across the

tested landscape types. The learning-probability bounds $a = 0$ and $b = 0.5$ are retained as conservative CLPSO-derived values so that individuals keep substantial self-influence while receiving cross-dimensional guidance under stagnation.

Table 4. Sensitivity analysis results of the stagnation threshold m on CEC2017 (30D). “Best” signifies the number of functions where the variant outperformed others. “Mean Rank” denotes the average Friedman ranking. The “+/ $\approx$ /−” column summarizes the Wilcoxon rank-sum test results ($\alpha = 0.05$) compared to the proposed CLFGO ($m = 5$). The most favorable results are highlighted in bold.

Algorithm	Best	Mean Rank	+/ $\approx$ /−
CLFGO ($m = 5$) (Proposed)	15	1.85	—
CLFGO ($m = 7$)	5	2.15	1/23/5
CLFGO ($m = 10$)	4	2.25	1/22/6
CLFGO ($m = 3$)	3	2.45	0/20/9
CLFGO ($m = 1$)	2	4.10	0/10/19

4.3. Benchmark Functions Overview

CLFGO is tested on the CEC2017 benchmark suite for real-parameter numerical optimization [35,36]. Following the common CEC2017 practice, F2 is excluded, leaving 29 scalable, bound-constrained test functions. These include two unimodal functions (F1 and F3), seven multimodal functions (F4–F10), ten hybrid functions (F11–F20), and ten composition functions (F21–F30). The hybrid and composition groups feature complex landscapes with shifted and asymmetric optima. Together, these functions allow an assessment of an algorithm’s search characteristics. Function definitions and properties are listed in Table 5.

Table 5. CEC2017 benchmark functions.

Function	Function Name	Class	Optimum
F1	Shifted and Rotated Bent Cigar Function	Unimodal	100
F3	Shifted and Rotated Zakharov Function	Unimodal	300
F4	Shifted and Rotated Rosenbrock’s Function	Multimodal	400
F5	Shifted and Rotated Rastrigin’s Function	Multimodal	500
F6	Shifted and Rotated Expanded Scaffer’s F6 Function	Multimodal	600
F7	Shifted and Rotated Lunacek Bi-Rastrigin Function	Multimodal	700
F8	Shifted and Rotated Non-Continuous Rastrigin’s Function	Multimodal	800
F9	Shifted and Rotated Lévy Function	Multimodal	900
F10	Shifted and Rotated Schwefel’s Function	Multimodal	1000
F11	Hybrid Function 1 (N = 3)	Hybrid	1100
F12	Hybrid Function 2 (N = 3)	Hybrid	1200
F13	Hybrid Function 3 (N = 3)	Hybrid	1300
F14	Hybrid Function 4 (N = 4)	Hybrid	1400
F15	Hybrid Function 5 (N = 4)	Hybrid	1500
F16	Hybrid Function 6 (N = 4)	Hybrid	1600
F17	Hybrid Function 6 (N = 5)	Hybrid	1700
F18	Hybrid Function 6 (N = 5)	Hybrid	1800
F19	Hybrid Function 6 (N = 5)	Hybrid	1900
F20	Hybrid Function 6 (N = 6)	Hybrid	2000
F21	Composition Function 1 (N = 3)	Composition	2100
F22	Composition Function 2 (N = 3)	Composition	2200
F23	Composition Function 3 (N = 4)	Composition	2300
F24	Composition Function 4 (N = 4)	Composition	2400
F25	Composition Function 5 (N = 5)	Composition	2500
F26	Composition Function 6 (N = 5)	Composition	2600
F27	Composition Function 7 (N = 6)	Composition	2700
F28	Composition Function 8 (N = 6)	Composition	2800
F29	Composition Function 9 (N = 3)	Composition	2900
F30	Composition Function 10 (N = 3)	Composition	3000

4.4. Performance Comparison with Other Algorithms

The proposed CLFGO is compared against nine established metaheuristics: Bat Algorithm (BA) [37], Moss Growth Optimization (MGO) [38], Parrot Optimizer (PO) [39],

Fungal Growth Optimizer (FGO) [32], Differential Evolution (DE) [16], Hunger Games Search (HGS) [40], Grey Wolf Optimizer (GWO) [41], Slime Mould Algorithm (SMA) [42], and Sine Cosine Algorithm (SCA) [43]. All algorithms are executed under identical conditions, including the same population size, maximum number of function evaluations, and 30 independent runs per function. Before the full benchmark comparison, Table 6 provides a diagnostic comparison between CLFGO and the original FGO on representative CEC2017 functions. The mean (*Avg*) and standard deviation (*Std*) of the best-found fitness values are reported in Table 7.

CLFGO obtained the lowest mean error on 21 of 29 functions and achieved a Friedman average rank of 1.5517, followed by DE (3.3103) and MGO (3.3448). On hybrid and composition functions (e.g., F12, F13, and F21), CLFGO yielded mean errors lower than FGO. On simpler functions (e.g., F1 and F6), CLFGO, MGO, and DE achieved comparable final fitness values. Based on the pairwise $+/-$ summary in Table 7, CLFGO performed better than HGS, SMA, and SCA on all 29 functions. Against MGO, CLFGO recorded 17 wins, 11 ties, and 1 loss. Against DE, CLFGO recorded 22 wins, 4 ties, and 3 losses (on F16, F17, and F20).

For the unimodal functions (F1 and F3), CLFGO converged to the global optimum on F1 with a standard deviation of 3.12×10^{-14} . For multimodal functions (F4–F10), CLFGO and DE reached the theoretical optimum on F6 ($Avg = 6.00 \times 10^2$), and CLFGO yielded the lowest mean on F4, F7, and F10. DE achieved the lowest mean on F9. For hybrid functions (F11–F20), CLFGO ranked first on 7 of 10 instances. For example, on F12 and F13, CLFGO achieved $Avg = 5.18 \times 10^3$ and 1.35×10^3 , respectively. For composition functions (F21–F30), CLFGO obtained the lowest mean on several instances, while FGO and MGO achieved the lowest means on F22 and F24, respectively.

We applied the Wilcoxon signed-rank test ($\alpha = 0.05$) to assess statistical significance; Table 8 reports the p -values. CLFGO achieved $p < 0.05$ against BA, PO, FGO, HGS, SMA, and SCA on most functions. Against MGO, non-significant p -values ($p > 0.05$) occurred on 11 functions, including F5 and F6. Against DE, non-significant p -values occurred on 4 functions, including F16, F17, and F20.

This diagnostic comparison provides direct evidence for the motivation behind CLFGO on representative functions from different landscape classes. The large performance gaps on F1, F13, and F30 indicate that the original FGO can converge to inferior regions under the same evaluation budget, whereas the negative gap on F22 shows that CLFGO is not uniformly superior. This pattern is consistent with the No Free Lunch theorem and motivates a balanced interpretation of the results.

Table 6. Diagnostic comparison between CLFGO and the original FGO on representative CEC2017 functions.

Function	Landscape Class	CLFGO Mean	FGO Mean	Relative Change
F1	Unimodal	1.00×10^2	2.62×10^3	96.18% lower
F13	Hybrid	1.35×10^3	2.07×10^4	93.49% lower
F22	Composition	2.83×10^3	2.30×10^3	23.20% higher
F30	Composition	5.16×10^3	1.61×10^5	96.79% lower

The practical effect size is therefore function-dependent. On difficult hybrid and composition cases such as F13 and F30, CLFGO provides large reductions in mean objective value relative to FGO. On F22, however, the original FGO obtains a lower mean value, suggesting that a single-attractor update can still be advantageous when the basin structure is compatible with the original fungal growth dynamics. Similarly, the non-significant

comparisons with MGO and DE on several functions show that some practical differences are small even when CLFGO has the best overall rank.

Figure 2 shows the convergence trajectories. CLFGO maintained continuous improvement in later stages without early stagnation. To evaluate population diversity across evaluations, we used the centroid-distance metric:

$$Div = \frac{1}{N} \sum_{i=1}^N \sqrt{\sum_{j=1}^D (x_{i,j} - \bar{x}_j)^2} \quad (9)$$

where $x_{i,j}$ is the j -th dimension of the i -th individual, and $\bar{x}_j$ is the mean across the population for dimension j . Figure 3 illustrates the diversity profiles of FGO and CLFGO. The CLFGO trajectory exhibits diversity rebounds. When the population stagnates, the CL mechanism provides new directional guidance, whereas FGO loses diversity and converges prematurely.

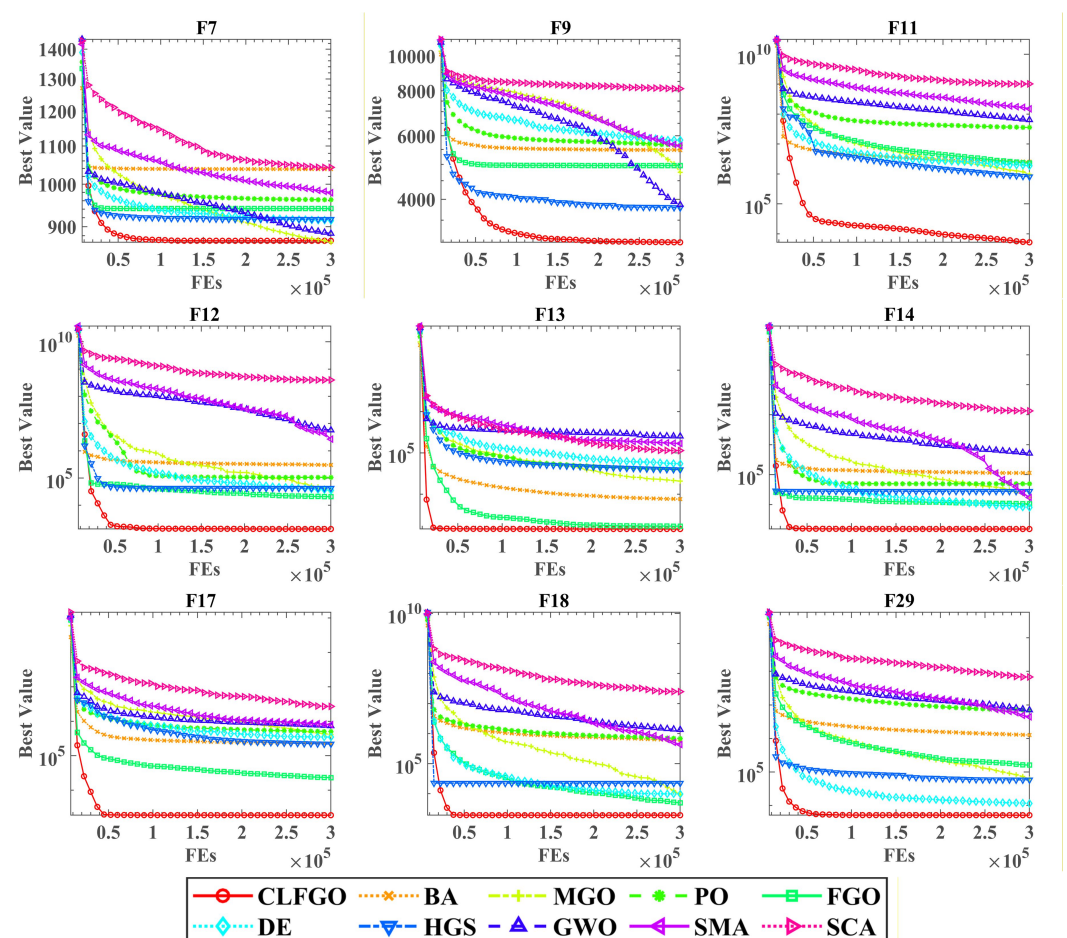


Figure 2. Convergence behavior analysis on nine representative benchmark functions. Each subplot corresponds to a specific test function, illustrating the decay of the objective value over the course of evaluations.

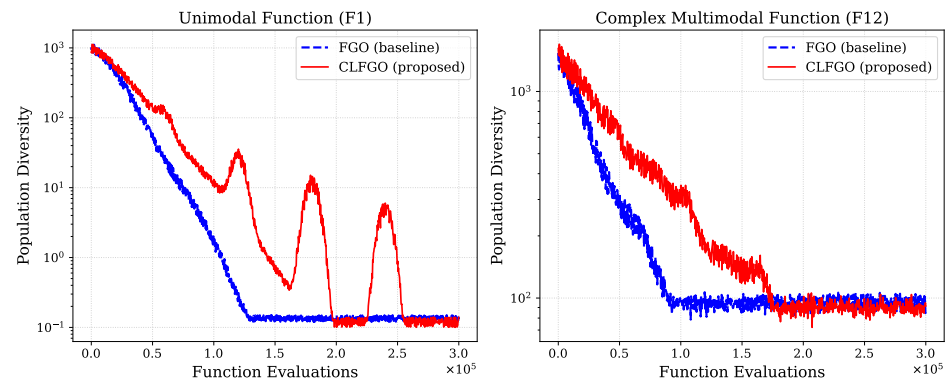


Figure 3. Population diversity trends of FGO and CLFGO across evaluations. The subplots illustrate the diversity metric over evaluations for representative functions (F1 and F12).

Table 7. Results of the CLFGO and Other Algorithms on CEC2017 Benchmark Functions.

Algo.	F1		F3		F4	
	Avg	Std	Avg	Std	Avg	Std
CLFGO	1.0000×10^2	3.1224×10^{-14}	3.0063×10^2	1.2162×10^0	4.2069×10^2	2.8216×10^1
FGO	2.6185×10^3	3.2117×10^3	3.0001×10^2	6.0714×10^{-3}	5.0936×10^2	3.6315×10^1
BA	5.7871×10^5	2.8969×10^5	3.0013×10^2	1.2475×10^{-1}	4.7587×10^2	3.8566×10^1
MGO	8.0227×10^4	7.1266×10^4	4.7192×10^4	8.2184×10^3	4.9016×10^2	1.4137×10^1
PO	5.9712×10^7	6.3476×10^7	4.9114×10^3	3.6644×10^3	5.2519×10^2	3.7885×10^1
DE	1.2465×10^3	2.6438×10^3	1.8776×10^4	4.7163×10^3	4.8979×10^2	9.0528×10^0
HGS	7.0059×10^3	5.2460×10^3	1.0356×10^3	2.6317×10^3	4.7948×10^2	3.3440×10^1
GWO	3.1471×10^9	2.3755×10^9	3.3884×10^4	1.0158×10^4	5.9220×10^2	7.5289×10^1
SMA	2.7439×10^9	1.1593×10^9	3.6615×10^4	7.6129×10^3	6.3196×10^2	6.5237×10^1
SCA	1.2267×10^{10}	1.7253×10^9	3.5714×10^4	6.7693×10^3	1.4070×10^3	2.4774×10^2
Algo.	F5		F6		F7	
	Avg	Std	Avg	Std	Avg	Std
CLFGO	5.6551×10^2	1.5657×10^1	6.0000×10^2	8.7542×10^{-5}	7.9017×10^2	9.5515×10^0
FGO	6.8698×10^2	3.0135×10^1	6.5051×10^2	7.2809×10^0	9.9622×10^2	7.2567×10^1
BA	8.1173×10^2	5.4997×10^1	6.7303×10^2	8.4768×10^0	1.5826×10^3	2.2143×10^2
MGO	5.6439×10^2	8.6187×10^0	6.0000×10^2	2.8616×10^{-4}	8.0374×10^2	1.4414×10^1
PO	7.2900×10^2	3.2296×10^1	6.5597×10^2	9.6543×10^0	1.1439×10^3	5.7275×10^1
DE	6.0625×10^2	9.3108×10^0	6.0000×10^2	2.1111×10^{-14}	8.4370×10^2	9.9951×10^0
HGS	6.1673×10^2	2.4442×10^1	6.0236×10^2	1.8238×10^0	8.7604×10^2	3.6968×10^1
GWO	6.0702×10^2	3.2027×10^1	6.0804×10^2	2.9029×10^0	8.6088×10^2	3.5516×10^1
SMA	7.1159×10^2	2.8875×10^1	6.4204×10^2	6.8119×10^0	1.0796×10^3	5.5986×10^1
SCA	7.7770×10^2	1.4535×10^1	6.4945×10^2	4.4785×10^0	1.1242×10^3	3.9541×10^1
Algo.	F8		F9		F10	
	Avg	Std	Avg	Std	Avg	Std
CLFGO	8.6878×10^2	1.3353×10^1	9.6523×10^2	8.2741×10^1	3.0367×10^3	2.5171×10^2
FGO	9.4138×10^2	2.2556×10^1	3.7809×10^3	5.7515×10^2	4.9612×10^3	5.8230×10^2
BA	1.0386×10^3	5.8998×10^1	1.2391×10^4	4.2309×10^3	5.4809×10^3	6.6000×10^2
MGO	8.6549×10^2	1.3013×10^1	9.4407×10^2	4.0522×10^1	4.7499×10^3	4.6335×10^2
PO	9.6217×10^2	2.5998×10^1	4.8895×10^3	6.8892×10^2	5.6530×10^3	6.7458×10^2
DE	9.1376×10^2	7.2919×10^0	9.0000×10^2	1.0342×10^{-13}	5.8382×10^3	3.1386×10^2
HGS	9.1667×10^2	2.9234×10^1	3.4154×10^3	1.0052×10^3	3.7864×10^3	6.3334×10^2
GWO	8.8478×10^2	1.3488×10^1	1.7883×10^3	5.8545×10^2	3.8627×10^3	4.3283×10^2
SMA	9.7919×10^2	1.9267×10^1	5.8268×10^3	1.0532×10^3	5.6326×10^3	5.6824×10^2
SCA	1.0430×10^3	1.5779×10^1	5.4439×10^3	1.0109×10^3	8.1121×10^3	3.4609×10^2

Table 7. Cont.

Algo.	F11		F12		F13	
	Avg	Std	Avg	Std	Avg	Std
CLFGO	1.1396×10^3	2.9169×10^1	5.1839×10^3	4.8496×10^3	1.3471×10^3	1.5072×10^1
FGO	1.2502×10^3	5.3899×10^1	2.2899×10^6	1.3651×10^6	2.0704×10^4	9.9764×10^3
BA	1.3172×10^3	5.9886×10^1	2.5551×10^6	2.2489×10^6	3.0351×10^5	1.6486×10^5
MGO	1.1772×10^3	2.4467×10^1	9.6367×10^5	5.4647×10^5	3.1574×10^4	2.2113×10^4
PO	1.3059×10^3	5.5515×10^1	3.5675×10^7	5.2930×10^7	1.0568×10^5	6.3400×10^4
DE	1.1632×10^3	2.1787×10^1	1.8651×10^6	9.4241×10^5	3.5372×10^4	2.1249×10^4
HGS	1.2010×10^3	3.1945×10^1	8.0344×10^5	6.2646×10^5	4.2522×10^4	2.6288×10^4
GWO	1.6985×10^3	5.1984×10^2	6.4165×10^7	9.3655×10^7	5.7469×10^6	4.3713×10^6
SMA	1.5857×10^3	1.1707×10^2	1.4945×10^8	1.0666×10^8	2.6827×10^6	4.3713×10^6
SCA	2.1585×10^3	3.9949×10^2	1.0162×10^9	2.7512×10^8	3.9721×10^8	1.1359×10^8
Algo.	F14		F15		F16	
	Avg	Std	Avg	Std	Avg	Std
CLFGO	1.4353×10^3	1.4754×10^1	1.5211×10^3	1.8680×10^1	2.1785×10^3	1.9776×10^2
FGO	1.6765×10^3	1.6433×10^2	1.0511×10^4	4.1576×10^3	2.8919×10^3	2.8358×10^2
BA	7.5921×10^3	4.2427×10^3	1.1284×10^5	5.8977×10^4	3.4022×10^3	4.5829×10^2
MGO	2.0743×10^4	1.6799×10^4	1.7591×10^4	1.3086×10^4	2.2031×10^3	1.4777×10^2
PO	3.9725×10^4	3.0240×10^4	4.9128×10^4	3.5657×10^4	3.0939×10^3	3.0875×10^2
DE	5.4598×10^4	4.1089×10^4	8.0324×10^3	6.0760×10^3	2.0778×10^3	1.6211×10^2
HGS	3.9166×10^4	3.0219×10^4	2.7673×10^4	1.6455×10^4	2.7487×10^3	2.8922×10^2
GWO	2.5305×10^5	3.1890×10^5	5.1330×10^5	9.8548×10^5	2.3731×10^3	2.4511×10^2
SMA	1.6881×10^5	1.0876×10^5	1.7247×10^5	6.4281×10^5	2.8088×10^3	3.3201×10^2
SCA	1.1362×10^5	6.5603×10^4	1.3411×10^7	1.1272×10^7	3.6244×10^3	1.7430×10^2
Algo.	F17		F18		F19	
	Avg	Std	Avg	Std	Avg	Std
CLFGO	1.8480×10^3	1.0555×10^2	1.8726×10^3	1.9099×10^2	1.9147×10^3	5.3481×10^0
FGO	2.2678×10^3	1.8486×10^2	2.3034×10^4	2.2942×10^4	4.9358×10^3	4.4045×10^3
BA	2.7794×10^3	3.3758×10^2	2.2400×10^5	1.2361×10^5	6.0405×10^5	2.9488×10^5
MGO	1.8838×10^3	6.5789×10^1	4.1333×10^5	2.5438×10^5	9.2284×10^3	7.2945×10^3
PO	2.2929×10^3	1.7937×10^2	5.0234×10^5	3.5019×10^5	6.8670×10^5	4.4668×10^5
DE	1.8311×10^3	4.9974×10^1	3.4824×10^5	2.2942×10^5	9.6917×10^3	5.6799×10^3
HGS	2.1973×10^3	2.3921×10^2	1.1893×10^5	1.6991×10^5	2.2311×10^4	2.2521×10^4
GWO	1.9739×10^3	1.4562×10^2	7.3615×10^5	8.2559×10^5	1.3495×10^6	5.5985×10^6
SMA	2.2596×10^3	2.1955×10^2	8.0325×10^5	1.0480×10^6	4.3189×10^5	5.5269×10^5
SCA	2.4009×10^3	1.2734×10^2	2.7108×10^6	1.1316×10^6	2.4890×10^7	1.4984×10^7
Algo.	F20		F21		F22	
	Avg	Std	Avg	Std	Avg	Std
CLFGO	2.1684×10^3	9.9530×10^1	2.3665×10^3	1.4913×10^1	2.8348×10^3	1.0140×10^3
FGO	2.5064×10^3	1.2560×10^2	2.4777×10^3	3.9610×10^1	2.3010×10^3	1.6392×10^0
BA	2.9968×10^3	1.8068×10^2	2.6322×10^3	7.6396×10^1	6.5599×10^3	1.8544×10^3
MGO	2.2551×10^3	7.7499×10^1	2.3719×10^3	1.2168×10^1	2.8287×10^3	1.3438×10^3
PO	2.4920×10^3	1.7183×10^2	2.5003×10^3	4.1516×10^1	3.0979×10^3	1.6947×10^3
DE	2.1425×10^3	7.2484×10^1	2.4094×10^3	9.0707×10^0	4.2000×10^3	2.0147×10^3
HGS	2.5202×10^3	2.1445×10^2	2.4286×10^3	3.1083×10^1	4.9980×10^3	1.2853×10^3
GWO	2.4200×10^3	1.5127×10^2	2.3859×10^3	2.3691×10^1	4.4522×10^3	1.3513×10^3
SMA	2.4322×10^3	1.3670×10^2	2.4751×10^3	2.5228×10^1	3.4253×10^3	1.6710×10^3
SCA	2.6059×10^3	1.0097×10^2	2.5559×10^3	2.2058×10^1	8.3144×10^3	2.2804×10^3

Table 7. Cont.

	F23		F24		F25	
Algo.	Avg	Std	Avg	Std	Avg	Std
CLFGO	2.7190×10^3	1.5028×10^1	2.9410×10^3	3.2951×10^1	2.8861×10^3	2.5847×10^0
FGO	3.0667×10^3	9.5498×10^1	3.2420×10^3	1.2636×10^2	2.9282×10^3	2.1691×10^1
BA	3.3586×10^3	1.5560×10^2	3.3492×10^3	1.0859×10^2	2.9017×10^3	1.8713×10^1
MGO	2.7204×10^3	1.4945×10^1	2.8981×10^3	1.2466×10^1	2.8874×10^3	9.1349×10^{-1}
PO	2.9415×10^3	7.1979×10^1	3.1225×10^3	6.4519×10^1	2.9347×10^3	2.4053×10^1
DE	2.7572×10^3	8.5421×10^0	2.9542×10^3	1.3829×10^1	2.8873×10^3	3.1655×10^{-1}
HGS	2.7778×10^3	2.7694×10^1	3.0268×10^3	6.2773×10^1	2.8910×10^3	1.2417×10^1
GWO	2.7535×10^3	3.2826×10^1	2.9216×10^3	4.7920×10^1	2.9830×10^3	5.8611×10^1
SMA	2.8523×10^3	3.0503×10^1	3.0201×10^3	3.1537×10^1	2.9933×10^3	3.8583×10^1
SCA	2.9984×10^3	2.7915×10^1	3.1611×10^3	3.1751×10^1	3.1737×10^3	5.3460×10^1
	F26		F27		F28	
Algo.	Avg	Std	Avg	Std	Avg	Std
CLFGO	3.9654×10^3	6.7275×10^2	3.2114×10^3	9.2192×10^0	3.1472×10^3	6.0208×10^1
FGO	5.1451×10^3	2.2654×10^3	3.5693×10^3	1.3619×10^2	3.2151×10^3	2.9918×10^1
BA	8.2947×10^3	2.5934×10^3	3.4247×10^3	1.5669×10^2	3.1332×10^3	5.7901×10^1
MGO	3.9685×10^3	4.6393×10^2	3.2126×10^3	4.8143×10^0	3.2339×10^3	1.4265×10^1
PO	6.3802×10^3	1.9104×10^3	3.3019×10^3	3.9899×10^1	3.3119×10^3	3.8727×10^1
DE	4.6657×10^3	1.0047×10^2	3.2060×10^3	3.3728×10^0	3.1900×10^3	4.6866×10^1
HGS	4.9201×10^3	2.5296×10^2	3.2258×10^3	1.3544×10^1	3.2167×10^3	3.8900×10^1
GWO	4.5536×10^3	4.0377×10^2	3.2533×10^3	2.5744×10^1	3.4435×10^3	1.5892×10^2
SMA	5.1881×10^3	6.9145×10^2	3.2530×10^3	2.4309×10^1	3.4201×10^3	6.2090×10^1
SCA	7.0468×10^3	2.4922×10^2	3.4033×10^3	3.9376×10^1	3.8630×10^3	1.3177×10^2
	F29		F30			
Algo.	Avg	Std	Avg	Std		
CLFGO	3.4181×10^3	1.0825×10^2	5.1623×10^3	1.8983×10^2		
FGO	4.4539×10^3	3.6476×10^2	1.6062×10^5	1.5679×10^5		
BA	5.0101×10^3	5.9578×10^2	1.2658×10^6	7.8956×10^5		
MGO	3.6333×10^3	9.4229×10^1	6.1726×10^4	3.0469×10^4		
PO	4.4419×10^3	3.5618×10^2	6.9220×10^6	5.1619×10^6		
DE	3.5130×10^3	7.5662×10^1	1.1489×10^4	2.7288×10^3		
HGS	3.7876×10^3	1.8836×10^2	5.8306×10^4	7.6510×10^4		
GWO	3.7556×10^3	1.5097×10^2	7.0190×10^6	8.4021×10^6		
SMA	4.0269×10^3	2.8186×10^2	4.4078×10^6	3.6517×10^6		
SCA	4.6254×10^3	2.0776×10^2	6.7893×10^7	2.9522×10^7		
Overall Rank						
Algo.	RANK	+/-/—	AVG			
CLFGO	1	~	1.5517			
FGO	5	27/1/1	5.3103			
BA	9	27/2/0	7.4483			
MGO	3	17/11/1	3.3448			
PO	8	28/1/0	7.1034			
DE	2	22/4/3	3.3103			
HGS	4	29/0/0	4.6552			
GWO	6	28/0/1	6.0345			
SMA	7	29/0/0	7.0345			
SCA	10	29/0/0	9.2069			

Table 8. The p -values of the CLFGO versus other algorithms on CEC2017.

Fun	BA	MGO	PO	FGO	DE	HGS	GWO	SMA	SCA
F1	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}
F3	2.71×10^{-1}	1.73×10^{-6}	1.73×10^{-6}	9.32×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}
F4	2.16×10^{-5}	3.52×10^{-6}	1.73×10^{-6}	1.92×10^{-6}	1.73×10^{-6}	2.60×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}
F5	1.73×10^{-6}	6.88×10^{-1}	1.73×10^{-6}	1.73×10^{-6}	1.92×10^{-6}	4.73×10^{-6}	7.69×10^{-6}	1.73×10^{-6}	1.73×10^{-6}
F6	1.73×10^{-6}	9.59×10^{-1}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}
F7	1.73×10^{-6}	1.25×10^{-4}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}
F8	1.73×10^{-6}	2.71×10^{-1}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.92×10^{-6}	4.90×10^{-4}	1.73×10^{-6}	1.73×10^{-6}
F9	1.73×10^{-6}	6.29×10^{-1}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.92×10^{-6}	1.73×10^{-6}	1.73×10^{-6}
F10	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.36×10^{-5}	1.92×10^{-6}	1.73×10^{-6}	1.73×10^{-6}
F11	1.73×10^{-6}	1.80×10^{-5}	1.73×10^{-6}	1.92×10^{-6}	4.39×10^{-3}	1.36×10^{-5}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}
F12	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}
F13	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}
F14	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}
F15	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}
F16	1.73×10^{-6}	4.41×10^{-1}	1.73×10^{-6}	1.73×10^{-6}	6.56×10^{-2}	2.88×10^{-6}	5.67×10^{-3}	1.73×10^{-6}	1.73×10^{-6}
F17	1.73×10^{-6}	1.16×10^{-1}	1.92×10^{-6}	1.92×10^{-6}	5.86×10^{-1}	4.29×10^{-6}	1.11×10^{-3}	3.18×10^{-6}	1.73×10^{-6}
F18	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}
F19	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}
F20	1.73×10^{-6}	3.61×10^{-3}	2.60×10^{-6}	1.73×10^{-6}	3.39×10^{-1}	3.52×10^{-6}	9.32×10^{-6}	2.35×10^{-6}	1.73×10^{-6}
F21	1.73×10^{-6}	3.93×10^{-1}	1.73×10^{-6}	1.73×10^{-6}	1.92×10^{-6}	1.73×10^{-6}	5.71×10^{-4}	1.73×10^{-6}	1.73×10^{-6}
F22	3.88×10^{-6}	1.41×10^{-1}	1.06×10^{-1}	6.14×10^{-1}	8.73×10^{-3}	1.24×10^{-5}	3.59×10^{-4}	4.49×10^{-2}	2.60×10^{-6}
F23	1.73×10^{-6}	6.58×10^{-1}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	2.60×10^{-6}	3.72×10^{-5}	1.73×10^{-6}	1.73×10^{-6}
F24	1.73×10^{-6}	3.88×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	5.19×10^{-2}	2.88×10^{-6}	2.70×10^{-2}	6.98×10^{-6}	1.73×10^{-6}
F25	1.13×10^{-5}	1.38×10^{-3}	1.73×10^{-6}	1.73×10^{-6}	2.83×10^{-4}	3.87×10^{-2}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}
F26	6.98×10^{-6}	8.13×10^{-1}	1.80×10^{-5}	8.73×10^{-3}	2.35×10^{-6}	1.92×10^{-6}	6.89×10^{-5}	5.22×10^{-6}	1.73×10^{-6}
F27	1.92×10^{-6}	7.50×10^{-1}	1.73×10^{-6}	1.73×10^{-6}	2.41×10^{-3}	2.41×10^{-4}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}
F28	8.29×10^{-1}	4.73×10^{-6}	1.73×10^{-6}	2.41×10^{-4}	9.77×10^{-3}	1.25×10^{-4}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}
F29	1.73×10^{-6}	6.34×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	5.31×10^{-5}	2.88×10^{-6}	1.92×10^{-6}	1.73×10^{-6}	1.73×10^{-6}
F30	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}

5. Application to Production Optimization

Reservoir production optimization involves determining an operational strategy to maximize Net Present Value (NPV) over the life of a project. The problem is high-dimensional because optimal control variables, such as well rates or bottom-hole pressures, must be specified for multiple wells across discrete time intervals. Because reservoir simulation is nonlinear and computationally expensive, metaheuristic algorithms are an attractive alternative to classical gradient-based methods.

We deployed CLFGO within a simulation-based optimization framework. A synthetic multi-channel reservoir model served as the test case. We used the Eclipse simulator as the forward model to assess the physical and economic outcomes of each candidate control strategy. The algorithm optimizes well controls to manage waterflooding dynamics, with the objective of increasing hydrocarbon recovery while limiting water production and injection costs.

The economic performance of a given control strategy is quantified by the NPV objective function, defined as:

$$NPV(\mathbf{x}, \mathbf{z}) = \sum_{t=1}^n \frac{\Delta t \cdot (Q_{o,t}r_o - Q_{w,t}r_w - Q_{i,t}r_i)}{(1+b)^{p_t}} \quad (10)$$

where $\mathbf{x}$ and $\mathbf{z}$ represent the control variables and the reservoir state, respectively; $Q_{o,t}$, $Q_{w,t}$, and $Q_{i,t}$ denote the cumulative oil production, water production, and water injection rates at time step t ; r_o , r_w , and r_i are the associated unit prices (or costs) for oil, produced water, and injected water, respectively; b is the annual discount rate; and p_t is the elapsed time in years. In this formulation, we treat the NPV optimization as an unconstrained problem to benchmark the numerical search performance of the algorithms.

5.1. Reservoir Model Description

We constructed a two-dimensional, heterogeneous synthetic reservoir model to represent a fluvial depositional environment. The model uses a five-spot well pattern with a single central production well (PRO1) and four corner injection wells (INJ1 through INJ4), as shown in Figure 4.

The domain was discretized on a 25×25 Cartesian grid. Each cell has uniform thickness and dimensions of 20 m by 20 m. Porosity was set to a constant value of 0.2. The permeability field was generated stochastically to produce a heterogeneous distribution with interconnected high-permeability channels embedded in a low-permeability matrix. Figure 4 shows the resulting log-permeability, $\ln(K)$, field.

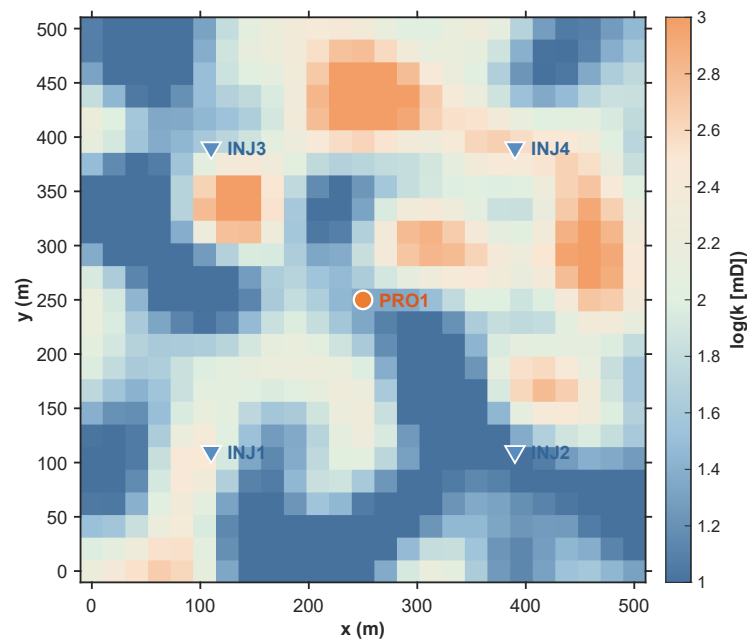


Figure 4. Two-dimensional synthetic reservoir model and well pattern arrangement.

The optimization framework was applied over a production horizon of 1800 days, partitioned into 6 discrete control steps. Optimization variables were assigned to all five wells, resulting in a problem dimension of 30 variables. Key economic parameters were set as follows: oil price (100 USD/STB), water injection cost (5 USD/STB), water production cost (5 USD/STB), and an annual discount rate (3%).

5.2. Analysis and Discussion of Experimental Results

We compared CLFGO with the original FGO and eight other metaheuristics (DE, MGO, HGS, PO, BA, GWO, SMA, and SCA). We ran each algorithm independently under identical conditions and recorded the mean NPV (Mean), standard deviation (Std), best, and worst outcomes.

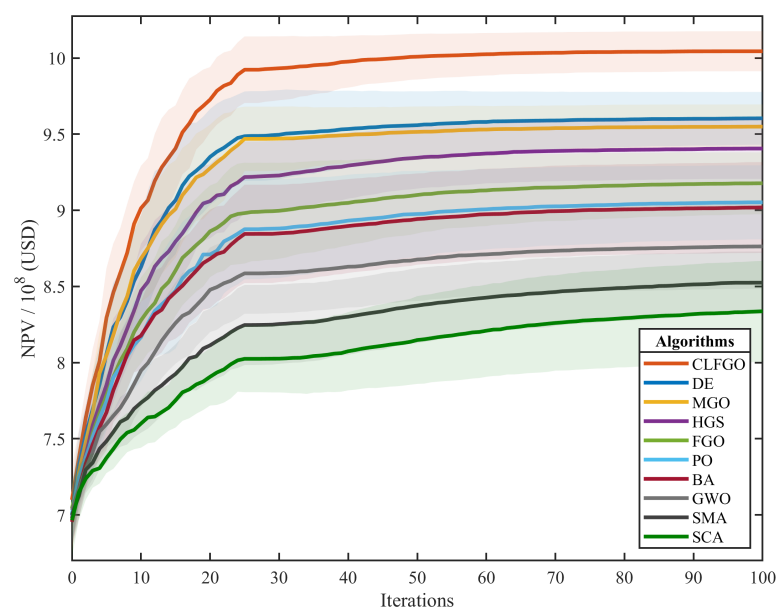
As shown in Table 9, CLFGO achieved a mean NPV of 9.9742×10^8 USD with a standard deviation of 1.353×10^7 . DE and MGO reached mean values of 9.536×10^8 and 9.487×10^8 USD, respectively. The remaining algorithms produced lower mean values with larger standard deviations.

The convergence trajectories in Figure 5 show the optimization progress. CLFGO reached its final NPV plateau earlier than the other algorithms, whereas DE and MGO converged at lower values.

Overall, CLFGO obtained higher NPV values with lower variance compared to the other tested metaheuristics on this production optimization problem.

Table 9. Experimental Results of All Algorithms on the Production Optimization Problem.

Algorithm	Mean (USD)	Std	Best (USD)	Worst (USD)
CLFGO	9.9742×10^8	1.353×10^7	1.046×10^9	9.785×10^8
DE	9.536×10^8	1.845×10^7	9.721×10^8	9.348×10^8
MGO	9.487×10^8	1.954×10^7	9.680×10^8	9.292×10^8
HGS	9.321×10^8	2.164×10^7	9.532×10^8	9.075×10^8
FGO	9.145×10^8	2.512×10^7	9.427×10^8	8.885×10^8
PO	9.043×10^8	2.737×10^7	9.349×10^8	8.798×10^8
BA	8.921×10^8	2.981×10^7	9.262×10^8	8.656×10^8
GWO	8.762×10^8	3.128×10^7	9.150×10^8	8.437×10^8
SMA	8.524×10^8	3.487×10^7	8.906×10^8	8.192×10^8
SCA	8.241×10^8	4.156×10^7	8.734×10^8	7.821×10^8

**Figure 5.** Convergence curve analysis of all algorithms on the production optimization problem.

6. Conclusions

This study proposed the Comprehensive Learning Fungal Growth Optimizer (CLFGO), an enhanced variant of FGO that incorporates a conditionally activated Comprehensive Learning (CL) strategy. When a candidate solution stagnates, the CL module constructs dimension-specific learning exemplars from the personal best positions of multiple peers. This approach provides new directional guidance and reduces the risk of premature convergence.

We evaluated CLFGO on 29 CEC2017 benchmark functions at 30 dimensions against nine metaheuristics. CLFGO achieved the lowest mean error on 21 functions and attained a Friedman average rank of 1.5517. We also applied CLFGO to a 30-variable reservoir production optimization problem, where it obtained a mean Net Present Value of 9.97×10^8 USD. These results show that CLFGO is most effective on several multimodal, hybrid, and composition landscapes, but they do not imply universal superiority. The cases where FGO, MGO, or DE perform comparably or better indicate that the benefit of comprehensive learning depends on the landscape structure and parameter setting.

Future research could evaluate CLFGO on higher-dimensional problems and apply it to a broader set of practical engineering tasks, including multi-objective optimization. Fur-

ther work should also include broader joint sensitivity analysis of the stagnation threshold and learning-probability bounds, as well as additional constrained reservoir-control cases.

Author Contributions: M.G.: Conceptualization, Software, Data Curation, Investigation, Writing—Original Draft; Z.S.: Methodology, Writing—Original Draft, Writing—Review and Editing, Validation, Supervision, Resources; X.Z.: Data Curation, Investigation; Y.T.: Data Curation, Investigation. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the project “Imaging Research and Service Technology of Borehole-Excited Time-Frequency Electromagnetic Method in Deep Karst Fractured-Vuggy Reservoirs” (No. 10350106-25-ZC0613-0011).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The numerical and experimental data used to support the findings of this study are included within the article.

Acknowledgments: The authors would like to thank the anonymous reviewers for their valuable comments and suggestions that helped improve the quality of this manuscript. During the revision of this work, the authors used ChatGPT (GPT-5.5, OpenAI) in order to enhance the English grammar and paraphrase some sentences. After using this tool/service, the authors reviewed and edited the content as needed, and they take full responsibility for the content of the publication.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

CLFGO	Comprehensive Learning Fungal Growth Optimizer
FGO	Fungal Growth Optimizer
CL	Comprehensive Learning
NPV	Net Present Value
BHP	Bottom-Hole Pressure
RPO	Reservoir Production Optimization
CEC	Congress on Evolutionary Computation
DE	Differential Evolution
HGS	Hunger Games Search
PSO	Particle Swarm Optimization
SMA	Slime Mould Algorithm
PO	Parrot Optimizer
BA	Bat Algorithm
MGO	Moss Growth Optimization
GWO	Grey Wolf Optimizer
SCA	Sine Cosine Algorithm
NFL	No Free Lunch

References

1. Kiwiel, K.C. *Methods of Descent for Nondifferentiable Optimization*; Springer: Berlin/Heidelberg, Germany, 2006; Volume 1133.
2. Wasley, R.; Shlash, M.A. Newton-Raphson algorithm for 3-phase load flow. In *Proceedings of the Institution of Electrical Engineers*; IET: London, UK, 1974; Volume 121, pp. 630–638.
3. Belegundu, A.D.; Chandrupatla, T.R. *Optimization Concepts and Applications in Engineering*; Cambridge University Press: Cambridge, UK, 2019.
4. Karki, S.; Raut, A. Computational Techniques and Emerging Technologies in the Optimization of Engineering Systems and Design Processes. *J. AI-Driven Autom. Predict. Maint. Smart Technol.* **2024**, *9*, 54–71.

5. Huang, X.; Hu, H.; Wang, J.; Yuan, B.; Dai, C.; Ablameyk, S.V. Dynamic Strongly Convex Sparse Operator with Learning Mechanism for Sparse Large-Scale Multi-Objective Optimization. In Proceedings of the 2024 6th International Conference on Data-driven Optimization of Complex Systems (DOCS), Hangzhou, China, 16–18 August 2024; pp. 121–127.
6. Abualigah, L.; Elaziz, M.A.; Khasawneh, A.M.; Alshinwan, M.; Ibrahim, R.A.; Al-Qaness, M.A.; Mirjalili, S.; Sumari, P.; Gandomi, A.H. Meta-heuristic optimization algorithms for solving real-world mechanical engineering design problems: A comprehensive survey, applications, comparative analysis, and results. *Neural Comput. Appl.* **2022**, *34*, 4081–4110. [\[CrossRef\]](#)
7. Boussaïd, I.; Lepagnot, J.; Siarry, P. A survey on optimization metaheuristics. *Inf. Sci.* **2013**, *237*, 82–117. [\[CrossRef\]](#)
8. Houssein, E.H.; Saeed, M.K.; Hu, G.; Al-Sayed, M.M. Metaheuristics for solving global and engineering optimization problems: Review, applications, open issues and challenges. *Arch. Comput. Methods Eng.* **2024**, *31*, 4485–4519. [\[CrossRef\]](#)
9. Naser, M.; Al-Bashiti, M.K.; Tapeh, A.T.G.; Naser, A.; Kodur, V.; Hawileh, R.; Abdalla, J.; Khodadadi, N.; Gandomi, A.H.; Eslamlou, A.D. A review of benchmark and test functions for global optimization algorithms and metaheuristics. *Wiley Interdiscip. Rev. Comput. Stat.* **2025**, *17*, e70028. [\[CrossRef\]](#)
10. Hu, H.; Shan, W.; Tang, Y.; Heidari, A.A.; Chen, H.; Liu, H.; Wang, M.; Escorcia-Gutierrez, J.; Mansour, R.F.; Chen, J. Horizontal and vertical crossover of sine cosine algorithm with quick moves for optimization and feature selection. *J. Comput. Des. Eng.* **2022**, *9*, 2524–2555. [\[CrossRef\]](#)
11. Hu, H.; Shan, W.; Chen, J.; Xing, L.; Heidari, A.A.; Chen, H.; He, X.; Wang, M. Dynamic individual selection and crossover boosted forensic-based investigation algorithm for global optimization and feature selection. *J. Bionic Eng.* **2023**, *20*, 2416–2442. [\[CrossRef\]](#)
12. Du, J.; Hu, H.; Wang, J. A Two-Stage Multi-Evolutionary Sampling Optimization Framework for Expensive Optimization Problems. In Proceedings of the 2025 International Conference on New Trends in Computational Intelligence (NTCI), Jinan, China, 17–19 October 2025; pp. 1–7. [\[CrossRef\]](#)
13. Ren, X.; Wang, H.; Hu, H.; Wang, J.; Ablameyko, S.V. Weighted committee-based surrogate-assisted differential evolution framework for efficient medium-scale expensive optimization. *Int. J. Mach. Learn. Cybern.* **2025**, *16*, 6513–6530. [\[CrossRef\]](#)
14. Blum, C.; Puchinger, J.; Raidl, G.R.; Roli, A. Hybrid metaheuristics in combinatorial optimization: A survey. *Appl. Soft Comput.* **2011**, *11*, 4135–4151. [\[CrossRef\]](#)
15. Sastry, K.; Goldberg, D.E.; Kendall, G. Genetic algorithms. In *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 93–117.
16. Storn, R.; Price, K. Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces. *J. Glob. Optim.* **1997**, *11*, 341–359. [\[CrossRef\]](#)
17. Vanneschi, L.; Silva, S. Particle swarm optimization. In *Lectures on Intelligent Systems*; Springer: Berlin/Heidelberg, Germany, 2023; pp. 105–111.
18. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the ICNN'95-International Conference on Neural Networks, Perth, WA, Australia, 27 November–1 December 1995; Volume 4, pp. 1942–1948.
19. Dorigo, M.; Birattari, M.; Stutzle, T. Ant colony optimization. *IEEE Comput. Intell. Mag.* **2007**, *1*, 28–39. [\[CrossRef\]](#)
20. Bonabeau, E.; Dorigo, M.; Theraulaz, G. *Swarm Intelligence: From Natural to Artificial Systems*; Number 1; Oxford University Press: Oxford, UK, 1999.
21. Tu, J.; Chen, H.; Wang, M.; Gandomi, A.H. The colony predation algorithm. *J. Bionic Eng.* **2021**, *18*, 674–710. [\[CrossRef\]](#)
22. Shan, W.; Hu, H.; Cai, Z.; Chen, H.; Liu, H.; Wang, M.; Teng, Y. Multi-strategies boosted mutative crow search algorithm for global tasks: Cases of continuous and discrete optimization. *J. Bionic Eng.* **2022**, *19*, 1830–1849. [\[CrossRef\]](#)
23. Hu, H.; Wang, J.; Huang, X.; Ablameyko, S.V. An Integrated Online-Offline Hybrid Particle Swarm Optimization Framework for Medium Scale Expensive Problems. In Proceedings of the 2024 6th International Conference on Data-driven Optimization of Complex Systems (DOCS), Hangzhou, China, 16–18 August 2024; pp. 25–32.
24. Wolpert, D.H.; Macready, W.G. No free lunch theorems for optimization. *IEEE Trans. Evol. Comput.* **2002**, *1*, 67–82. [\[CrossRef\]](#)
25. Velasco, L.; Guerrero, H.; Hospitaler, A. A literature review and critical analysis of metaheuristics recently developed. *Arch. Comput. Methods Eng.* **2024**, *31*, 125–146. [\[CrossRef\]](#)
26. Smit, S.K.; Eiben, A. Parameter tuning of evolutionary algorithms: Generalist vs. specialist. In *Proceedings of the European Conference on the Applications of Evolutionary Computation*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 542–551.
27. Gutjahr, W.J. Convergence analysis of metaheuristics. In *Matheuristics: Hybridizing Metaheuristics and Mathematical Programming*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 159–187.
28. Eiben, Á.E.; Hinterding, R.; Michalewicz, Z. Parameter control in evolutionary algorithms. *IEEE Trans. Evol. Comput.* **2002**, *3*, 124–141. [\[CrossRef\]](#)
29. Ouyang, K.; Fu, S.; Chen, Y.; Cai, Q.; Heidari, A.A.; Chen, H. Escape: An optimization method based on crowd evacuation behaviors. *Artif. Intell. Rev.* **2024**, *58*, 19. [\[CrossRef\]](#)
30. Ouyang, K.; Wei, D.; Sha, X.; Yu, J.; Zhao, Y.; Qiu, M.; Fu, S.; Heidari, A.A.; Chen, H. Beaver behavior optimizer: A novel metaheuristic algorithm for solar PV parameter identification and engineering problems. *J. Adv. Res.* **2025**, *in press*. [\[CrossRef\]](#)

31. Meng, A.B.; Chen, Y.C.; Yin, H.; Chen, S.Z. Crisscross optimization algorithm and its application. *Knowl.-Based Syst.* **2014**, *67*, 218–229. [\[CrossRef\]](#)
32. Abdel-Basset, M.; Mohamed, R.; Abouhawwash, M. Fungal growth optimizer: A novel nature-inspired metaheuristic algorithm. *Expert Syst. Appl.* **2025**, *250*, 123456.
33. Chen, S.; Montgomery, J.; Bolufé-Röhler, A. Measuring the curse of dimensionality and its effects on particle swarm optimization and differential evolution. *Appl. Intell.* **2015**, *42*, 514–526. [\[CrossRef\]](#)
34. Liang, J.J.; Qin, A.K.; Suganthan, P.N.; Baskar, S. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE Trans. Evol. Comput.* **2006**, *10*, 281–295. [\[CrossRef\]](#)
35. Salgotra, R.; Singh, U.; Saha, S. Improved cuckoo search with better search capabilities for solving CEC2017 benchmark problems. In Proceedings of the 2018 IEEE Congress on Evolutionary Computation (CEC), Rio de Janeiro, Brazil, 8–13 July 2018; pp. 1–7.
36. Wu, G.; Mallipeddi, R.; Suganthan, P.N. *Problem Definitions and Evaluation Criteria for the CEC 2017 Competition on Constrained Real-Parameter Optimization*; Nanyang Technological University: Singapore, 2017; Volume 9, p. 2017.
37. Yang, X.S.; Hossein Gandomi, A. Bat algorithm: A novel approach for global engineering optimization. *Eng. Comput.* **2012**, *29*, 464–483. [\[CrossRef\]](#)
38. Zheng, B.; Chen, Y.; Wang, C.; Heidari, A.A.; Liu, L.; Chen, H. The moss growth optimization (MGO): Concepts and performance. *J. Comput. Des. Eng.* **2024**, *11*, 184–221. [\[CrossRef\]](#)
39. Lian, J.; Hui, G.; Ma, L.; Zhu, T.; Wu, X.; Heidari, A.A.; Chen, Y.; Chen, H. Parrot optimizer: Algorithm and applications to medical problems. *Comput. Biol. Med.* **2024**, *172*, 108064. [\[CrossRef\]](#)
40. Yang, Y.; Chen, H.; Heidari, A.A.; Gandomi, A.H. Hunger games search: Visions, conception, implementation, deep analysis, perspectives, and towards performance shifts. *Expert Syst. Appl.* **2021**, *177*, 114864. [\[CrossRef\]](#)
41. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey wolf optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [\[CrossRef\]](#)
42. Li, S.; Chen, H.; Wang, M.; Heidari, A.A.; Mirjalili, S. Slime mould algorithm: A new method for stochastic optimization. *Future Gener. Comput. Syst.* **2020**, *111*, 300–323. [\[CrossRef\]](#)
43. Mirjalili, S. SCA: A sine cosine algorithm for solving optimization problems. *Knowl.-Based Syst.* **2016**, *96*, 120–133. [\[CrossRef\]](#)

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.