" A-GGCO: A Hybrid Adaptive Metaheuristic for Optimization of Paillier Homomorphic Encryption Parameters"

Rekha Gaitond, Asst. Prof., *Computer Science and Engineering, PDA College of Engineering,* Kalburgi, India *Orcid Id: 0009-0008-5259-5444*
Dr. Gangadhar S. Biradar, Professor, *Electronics and Communication Engineering, PDA College of Engineering,* Kalburgi, India
Dr. Shubhangi D.C., Professor, *Visveswaraya Technological University, Kalburgi, India, Orcid Id:0000-0002-9221-166X*

Abstract

Bio-inspired optimization approaches are powerful tools for tackling high-dimensional and complex problems. The research proposes a Hybrid Adaptive Greylag Goose–Crayfish Optimization (A-GGCO) algorithm that introduces a diversity-driven switching mechanism that balances exploration and exploitation by combining the behaviors of geese and crayfish. Tested on classical benchmarks and CEC2017/2020/2022 suites against SHADE, L-SHADE, DE, CMA-ES, HO, JSO, and CSO-MA, it consistently achieved faster convergence, higher accuracy, and improved robustness. Statistical analyses confirmed significant improvements, with effect-size measures (Cohen's d) indicating large to very large gains, particularly in challenging cases such as Michalewicz ($d > 2.3$) and Griewank/Salomon ($d > 3.0$). Sensitivity, diversity, and ablation studies verified its adaptability and the advantages of hybridization. A practical case study on Paillier Homomorphic Encryption further highlighted reductions in computation time and enhanced efficiency in secure cloud environments, while its lightweight design proved effective for IoT healthcare by enabling energy-efficient, latency-sensitive optimization. Overall, A-GGCO emerges as a robust and versatile framework for both benchmark optimization and real-world applications in cryptography, IoT, and cloud security.

Keywords: Hybrid Optimization, Greylag Goose Optimization, Crayfish Optimization, Nature-Inspired Algorithms, Global–Local Search

1. INTRODUCTION

Optimization refers to the process of identifying the best solution from a set of feasible alternatives under specific objectives and constraints. In computational sciences, optimization methods are generally classified into deterministic and stochastic approaches. Deterministic algorithms yield the same solution for a given input, while stochastic methods incorporate randomness, allowing diverse exploration of the solution space and the possibility of escaping local optima.[1][2][3] Among stochastic approaches, metaheuristic algorithms have gained remarkable attention for solving nonlinear, multimodal, and high-dimensional optimization problems.[1][4][5]

Within this class, bio-inspired metaheuristics—which emulate natural, evolutionary, or ecological behaviors—have been widely recognized for their effectiveness in maintaining a balance between exploration (global search across the solution landscape) and exploitation (local refinement around promising regions) [7][8][13]. Despite their success, such algorithms often face two persistent challenges: (i) premature convergence, where the search stagnates around local optima due to insufficient exploration, and (ii) parameter sensitivity, where performance depends heavily on fine-tuned parameters such as mutation factors, population size, or learning coefficients [11][50].

To overcome these challenges, hybrid optimization approaches have emerged, integrating complementary strategies to enhance adaptability, robustness, and convergence efficiency. These hybrid frameworks seek to exploit the strengths of multiple algorithms while mitigating their individual weaknesses, thereby achieving an adaptive balance between global exploration and local exploitation [19][20][21].

1.1 Greylag Goose Optimization (GGO)

Greylag Goose Optimization (GGO) is a recent swarm-based metaheuristic inspired by the migratory and foraging behavior of greylag geese [42][43]. In nature, geese migrate in V-shaped formations, which reduces air resistance, enhances communication, and conserves energy. This cooperative phenomenon is modeled in GGO, where each candidate solution represents a goose, and the best-performing solution acts as the leader guiding the flock. Importantly, leadership in GGO is not static. A dynamic switching mechanism enables the replacement of underperforming leaders by better-performing candidates, thereby maintaining diversity and preventing premature stagnation. This mechanism makes GGO particularly effective in global exploration, as it ensures adaptive exploration of new regions in the search space [44].

1.2 Crayfish Optimization (CO)

In contrast, Crayfish Optimization (CO) is inspired by the intelligent foraging and defensive behaviors of crayfish. These creatures exhibit fine-grained local search strategies, such as adaptive backward-walking and variable step-size adjustments depending on their proximity to food or threats [45][46]. In algorithmic terms, this translates into an adaptive step-size mechanism: larger exploratory steps are taken when the solution is far from the optimum, while smaller exploitative steps are applied when near promising regions. This adaptive adjustment makes CO highly effective for local exploitation, enabling precise refinement of candidate solutions and reducing the risk of premature convergence [46].

1.3 Motivation for Hybridization

Both GGO and CO possess unique strengths that complement one another. GGO excels in broad exploration, dynamically covering the search space and preventing stagnation, while CO provides intensive local refinement, adaptively improving the quality of solutions. However, when used individually, each algorithm is limited: GGO may lack exploitative precision, and CO may struggle to escape local optima. By hybridizing GGO and CO into a unified framework, it is possible to achieve a synergistic balance between exploration and exploitation.

1.4 Contributions of The Work

Building upon these complementary strengths, the research introduces the Hybrid Greylag Goose–Crayfish Optimization (GGCO) algorithm. The key contributions are as follows:

- Hybrid Algorithm Design – Development of a novel metaheuristic that dynamically transitions between GGO and CO phases based on population diversity.
- Sensitivity and Diversity Analysis – Assessment of robustness under varying population sizes, iteration limits, and diversity thresholds.
- Ablation and Impact Study – Quantitative evaluation of the contributions of GGO and CO components within the hybrid framework.
- Comprehensive Benchmarking – Evaluation on classical functions and modern benchmark suites (CEC2017, CEC2020, CEC2022) against advanced optimizers such as SHADE, L-SHADE, DE, CMA-ES, HO, JSO, and CSO-MA.
- Real-World Application – Demonstration of the algorithm's effectiveness in optimizing parameters of Paillier Homomorphic Encryption, highlighting its applicability in cryptographic domains.

2. RELATED WORK

2.1 Nature-Inspired Population-Based Optimization Algorithms

Nature-inspired metaheuristics, particularly population-based approaches, have demonstrated remarkable adaptability in solving nonlinear and multimodal optimization problems [6][8][13][14][15]. These algorithms simulate biological or ecological behaviors such as reproduction, hunting, or swarming, enabling effective exploration and exploitation of the search space [9][10][16][17][18].

Population-based algorithms typically follow an iterative process: (i) initialize a population of candidate solutions, (ii) evaluate their fitness, (iii) modify solutions using operators such as crossover, mutation, or position updates, and (iv) repeat until a stopping criterion is satisfied [3][50].

In recent years, a variety of new optimizers have emerged, drawing inspiration from diverse biological phenomena. Table 1 highlights selected examples, including Starling Murmuration Optimizer (SMO) and Quantum-Based Avian Navigation Optimizer (QANA), with their core inspirations, mechanisms, strengths, and limitations [55][56].

Table 1. Comparative analysis of recent metaheuristic optimizers

Algorithm	Inspiration	Core Mechanism	Strengths	Limitations	Ref
GEA (Genetic Engineering Algorithm)	Genetic engineering principles	Extends GA with gene isolation, purification, insertion, and expression	Preserves beneficial traits, faster convergence in combinatorial problems	Requires problem-specific customization; depends on accurate gene manipulation	[51]
ESO1 & ESO2 (Enhanced Snake Optimizers)	Snake hunting	ESO1 uses logistic maps; ESO2 uses Lévy flights for food search	Strong exploration and exploitation; effective on benchmarks	Sensitive to parameters; tuning required	[52]
HHO (Harris Hawk Optimization)	Cooperative hawk hunting	Surprise pounce strategy with adaptive exploration–exploitation phases	Effective for complex, nonlinear, high-dimensional problems; easy to implement	Can converge prematurely; performance varies with landscape	[53]
GJO (Golden Jackal Optimization)	Social hunting of jackals	Collaborative search guided by hierarchy	Competitive performance, adaptable	New, requires wider validation	[53]

GWO (Grey Wolf Optimization)	Wolf hierarchy & hunting	Alpha–beta–delta leadership with encircling & hunting strategies	Widely used; balances exploration/exploitation	Can get stuck in local optima on multimodal problems	[54]
SMO (Starling Murmuration Optimizer)	Starling flock dynamics	Separation, diving, whirling to mimic murmuration	Maintains diversity, reduces premature convergence	Needs careful tuning; performance context-dependent	[55]
QANA (Quantum Avian Navigation Optimizer)	Bird migration + quantum principles	Multi-flock structures, quantum mutation, qubit crossover	Effective for high-dimensional feature selection; robust	Complex implementation; sensitive to parameters	[56]

As shown in Table 1, recent metaheuristic optimizers are inspired by diverse natural and computational phenomena, ranging from genetic engineering and predator–prey interactions to swarm dynamics and quantum principles. Each algorithm demonstrates distinct strengths such as maintaining diversity, improving convergence speed, or adapting to high-dimensional search spaces. However, most of them also suffer from limitations like parameter sensitivity, premature convergence, or implementation complexity. These trade-offs highlight the importance of designing hybrid algorithms that can combine complementary advantages while reducing individual weaknesses.

2.2 Hybrid Approaches in Population-Based Optimization

Hybridization can be introduced at multiple stages:

- Initialization: combining random generation with statistical sampling to ensure population diversity.
- Evaluation: using surrogate models or distributed computing to reduce computational load.
- Search process: integrating different exploration–exploitation strategies either sequentially or in parallel.[20][21]

Hybrid nature-inspired algorithms can be classified along four key dimensions:

1. Methods – combining multiple metaheuristics (e.g., GWO-DE), integrating metaheuristics with domain-specific models (e.g., PSO-EBP), or fusing with soft computing techniques (e.g., GA-fuzzy systems) [23].
2. Level – High-Level Teamwork (HLT) hybrids preserve the identity of constituent algorithms, while Low-Level Teamwork (LLT) hybrids exchange operators or components directly [24][25].
3. Execution – hybrids may run sequentially (e.g., GA followed by PSO) or in parallel, depending on computing architecture and synchronization requirements [26][27].
4. Control Strategy – integrational hybrids embed one method inside another (e.g., PSO with local search), while cooperative hybrids allow independent but interacting algorithms (e.g., multi-island GA) [27][28][29].

The classification provides a framework for systematically designing hybrid strategies tailored to specific optimization problems. Several hybrid methods have been proposed that synergize exploration and exploitation. Table 2 summarizes notable hybrid strategies, such as PSO-GA, DE-ACO, GWO-SSA, and JSO-SA, highlighting their key strengths and challenges.

Table 2. Summary of notable hybrid optimization strategies

Hybrid Algorithm	Components	Strengths	Challenges	Ref
PSO–GA	Particle Swarm + Genetic Algorithm	Avoids early convergence; improved speed in multimodal search	Complex parameter tuning	[12][23]
DE–ACO	Differential Evolution + Ant Colony	Strong exploration; efficient in discrete problems	Implementation complexity	[31]
GWO–FOA	Grey Wolf + Fruit Fly	High accuracy, faster convergence	Struggles in high dimensions	[32]
GWO–SSA	Grey Wolf + Salp Swarm	Effective in feature selection, parameter tuning	Problem-specific design	[33]
JSO–SA	Jellyfish + Simulated Annealing	Powerful local search; good convergence	High computational cost	[34]
JSO–TS	Jellyfish + Tabu Search	Strong exploitation ability	Complex integration	[35]
HHO–DE	Harris Hawks + Differential Evolution	Adaptive exploration; diverse search patterns	Parameter balancing needed	[37]
WOA–LFCM	Whale + Lévy Flight & Chaotic Maps	High exploration diversity	Chaotic maps increase overhead	[38]

SSA–AM	Salp Swarm + Adaptive Mechanisms	Dynamic balance of exploration & exploitation	Higher complexity	[39]
CSA–OBL	Chameleon Swarm + Opposition Learning	Good diversity, avoids local optima	Instability in noisy settings	[40]
HIPO–CM	Particle Optimization + Chaotic Maps	Better diversity	Scalability issues	[41]

3. METHODOLOGY: HYBRID A-GGCO ALGORITHM

This section provides a systematic explanation of how the algorithm operates, including its behavioral inspiration, adaptive switching mechanism, computational process, and complexity analysis.

3.1 Adaptive Hybridization Strategy

The key idea behind the hybridization is to achieve a dynamic balance between exploration (searching broadly across the solution space) and exploitation (intensifying the search near promising regions). To prevent premature convergence and stagnation, the algorithm utilizes an adaptive diversity threshold that determines whether the search should prioritize global exploration or local exploitation.

At the initialization stage, a population of n candidate solutions is generated uniformly at random within the problem's search space:

$$X = \{x_1, x_2, ..., x_n\} \in \text{Uniform } (L, U)^{d} \quad (1)$$

where [L,U] defines the lower and upper bounds of the search space and d is the dimensionality of the problem. The best solution is initialized as $x_{best} \leftarrow$ None, $f_{best} \leftarrow \infty$.

A minimum diversity threshold is defined as:

$$D_{min} = 0.15 \cdot \|U - L\| / \sqrt{d} \quad (2)$$

D_{min} corresponds to 15% of an "equivalent" per-dimension range. It does not map directly to a percentage of the volume of the search space. D_{min} provides a reference value for deciding whether the algorithm should prioritize exploration or exploitation.

During each iteration, the fitness of every individual is evaluated. If a candidate solution outperforms the current best, the values of x_{best}, f_{best} are updated accordingly. After evaluation, the population diversity is calculated to assess how widely dispersed the solutions are in the search space. Diversity at iteration t is given by:

$$D^t = \frac{1}{n}\sum_{i=1}^{n} \left\| x_i^t - \bar{x}^t \right\| \text{, where } \bar{x}^t = \frac{1}{n}\sum_{i=1}^{n} x_i^t \quad (3)$$

where $\bar{x}^t$ is the mean position of every agent at iteration t.

If the measured diversity D^t is greater than the threshold D_{min}, the global search phase (GGO) is applied. In this phase, the leader of the flock is selected as the best-performing individual:

$$x_{leader} \leftarrow \text{argmin } f(x_i) \quad (4)$$

If the leader stagnates for several iterations (ΔT), a leader switching mechanism is applied to reintroduce diversity:

$$x_{leader}^{t+1} = x_{leader}^{t} + r \cdot (x_{rand}^{t} - x_{leader}^{t}), \text{ where } r \in \text{Uniform } (0,1) \quad (5)$$

For the remaining geese, positions are updated using a formation update rule, which combines attraction toward the leader, interaction with a random neighbor x_j, and Gaussian noise:

$$x_i^{t+1} = x_i^t + \alpha \cdot (x_{leader}^t - x_i^t) + \beta \cdot \left(x_j^t - x_i^t\right) + \mathcal{N}\ (\theta, \sigma^2), \quad (6)$$

where α and β are weighting parameters, and $\mathcal{N}\ (\theta, \sigma^2)$ introduces stochastic perturbations to avoid premature convergence.

On the other hand, if the diversity falls below the threshold ($D^t \le D_{min}$), the algorithm switches to the local search phase (CO). Here, each individual simulates the backward-walking behavior of crayfish. The adaptive step size is defined as:

$$S_i \leftarrow 1 / (1 + \|x_i - x_{best}\|) \quad (7)$$

which ensures that individuals closer to the global best move in smaller, exploitative steps, while those farther away move in larger, exploratory steps. Each position is updated as:

$$x_i \leftarrow x_i + S_i \cdot R \cdot \mathcal{N}\ (\mu, \sigma^2), R \sim U(-1,1)^d \quad (8)$$

This mechanism ensures that local exploitation is intensified for escaping local optima. If stagnation persists, the step size S_i is dynamically adapted by either expanding or shrinking, allowing the algorithm to escape local traps or focus more precisely on the search region. A crucial feature of A-GGCO is the adaptive diversity update mechanism, where the threshold is not kept static but evolves dynamically according to:

$$D_{min}(t) = \gamma \cdot D_{min}(t-1) + (1-\gamma) \cdot D_t, \quad (9)$$

with $\gamma \in [0,1]$ serving as a smoothing factor. This update strategy prevents the algorithm from remaining locked in either global or local search for too long and promotes a smooth balance between exploration and exploitation throughout the optimization process.

The iterative search proceeds until the stopping criterion is satisfied, which may be defined as reaching the maximum number of iterations (T_{max}) or the maximum number of evaluations (E_{max}). Upon termination, the algorithm outputs the best solution x_{best} along with its corresponding fitness value f_{best}.

3.2 Algorithm Steps

The operational workflow is summarized in Algorithm-A-GGCO:

1. Initialization: Generate population and set initial diversity threshold D_{min}.
2. Evaluation: Assess fitness and track the global best solution.
3. Diversity Check: Calculate D_t.
    - If $D^t > D_{min}$: perform GGO-based global exploration.
    - Else: perform CO-based local exploitation.
4. Leader Switching & Adaptation: If no progress is observed, perturb leaders (GGO) or adapt step sizes (CO).
5. Threshold Update: Adjust D_{min} dynamically based on progress.
6. Termination: Stop when iteration or evaluation limits are reached, and return the best solution.

Steps 3,4, and 5 are detailed in Fig.1.

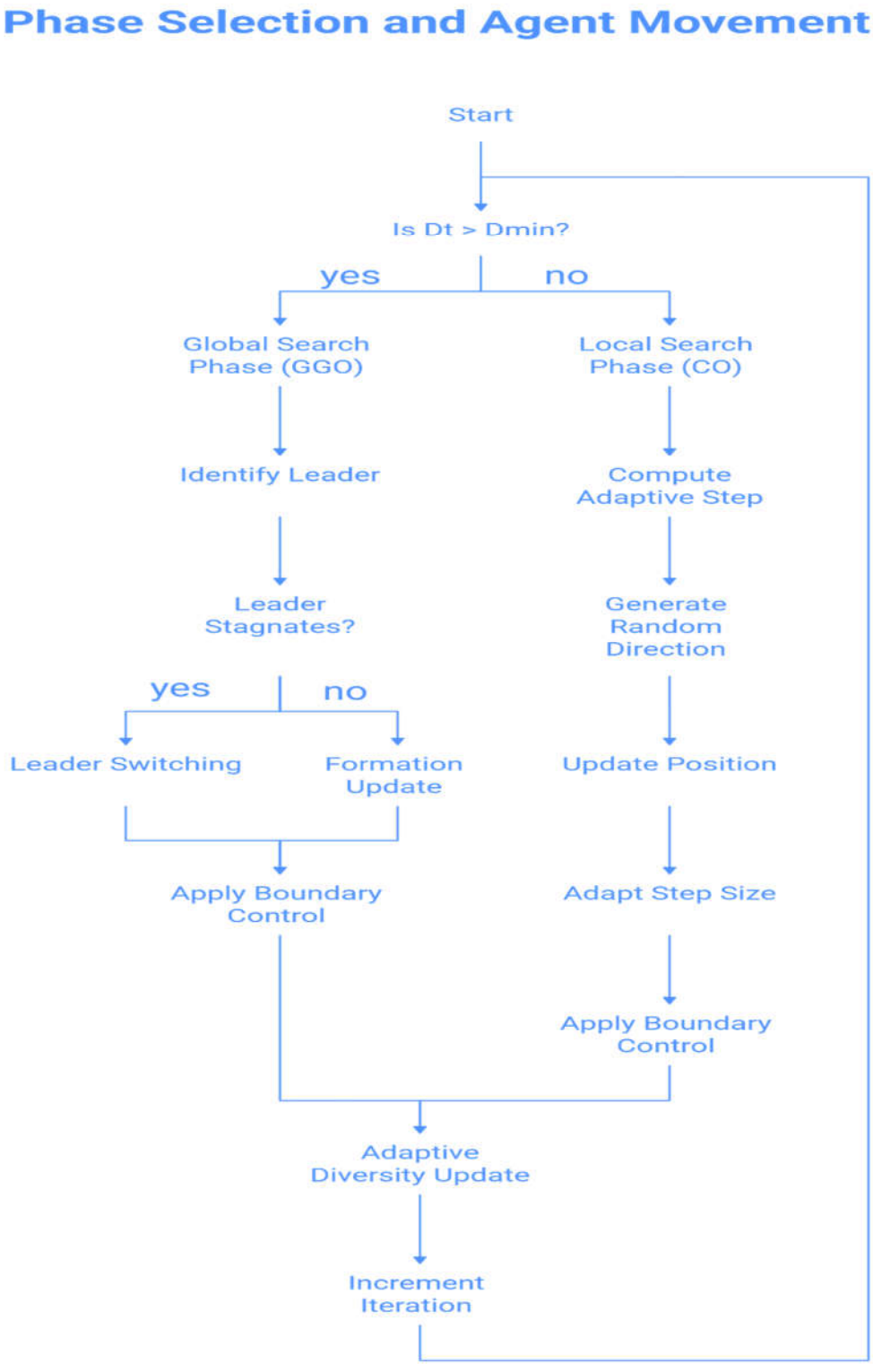


Fig. 1. Leader Switching and Adaptation

This dual-phase framework enables the algorithm to dynamically adjust between broad exploration and focused exploitation.

3.3 Complexity Analysis

One of the main challenges in hybrid algorithms is computational complexity. Unlike simple metaheuristics, hybrids require additional coordination between constituent methods, potentially increasing overhead. However, the performance gain from improved convergence and robustness often outweighs this cost [30].

Computational complexity of the hybrid GGO–CO algorithm arises from combining GGO's swarm-based evolutionary selection with CO's directional search and obstacle avoidance mechanisms. The primary complexity components include:

- Population initialization: O(P×D), where P is population size and D is problem dimension.
- Fitness evaluation: O(P×T×f), where T is the number of iterations and f is the complexity of the fitness function (often the most expensive step).
- Position updates (GGO and CO): O(P×D×T).
- Evolutionary selection and ranking: O(T×PlogP).
- Hybrid coordination overhead: O(T×P×D).

Combining these, the total time complexity is: O(P×D×T+T×PlogP+P×T×f)

Since fitness evaluations dominate in most applications, the highest-order term is:

- O(P×T×f) when the fitness function is computationally expensive.
- Otherwise, it simplifies to O(P×D×T).
- The space complexity is O(P×D), required to store the population with minimal additional overhead for hybrid operations.

This complexity profile highlights that while hybridization adds overhead, its linear scalability with population size, dimensionality, and iteration count ensures practical efficiency. Compared with mainstream algorithms such as Differential Evolution (DE), Covariance Matrix Adaptation Evolution Strategy (CMA-ES), SHADE, and L-SHADE, Hybrid GGO–CO achieves competitive per-iteration complexity (O(N×D)).

Algorithm: A-GGCO: Adaptive Greylag Goose–Crayfish Optimization Algorithm

---

Input:

- *n*: Population size
- *[L, U]*: Lower and upper bounds
- *Tmax*: Maximum iterations
- *Emax*: Maximum evaluations
- *γ*: Adaptation factor for diversity update
- *d*: Dimension of problem space

Output:

- *xbest*: Best solution found
- *fbest*: Best fitness value

1. Initialization:

Initialize population $X = \{x_1, x_2, ..., x_n\} \in \text{Uniform}(L, U)^d$

Initialize the best solution found so far:

$x_{best} \leftarrow \text{None}, f_{best} \leftarrow \infty$

Initialize diversity threshold $D_{min} = 0.15 \cdot \|U - L\| / \sqrt{d}$

Set iteration counter $t \leftarrow 0$

2. Main loop:

while ($t < T_{max}$ or $evals < E_{max}$) do

a. Fitness evaluation:

for each $x_i \in X$ do:

Evaluate the fitness $f(x_i)$ using the objective function.

if $f(x_i) < f_{best}$ then:

Update best solution:

$x_{best} \leftarrow \text{argmin } f(x_i)$

$f_{best} \leftarrow f(x_i)$

end if

end for

b. Population diversity calculation:
Compute population diversity D^t:

$$D^t = \frac{1}{n}\sum_{i=1}^{n}\left|\left|x_i^t - \bar{x}^t\right|\right| \quad , \ \bar{x}^t = \frac{1}{n}\sum_{i=1}^{n} x_i^t$$

where $\bar{x}^t$ is the mean position of every agent at iteration t.

c. Phase selection:
if $D^t > D_{min}$ then // Global Search Phase (GGO)
a. Identify leader $x_{leader} \leftarrow \text{argmin } f(x_i)$
b. If the leader stagnates (no improvement for ΔT iterations):

for each goose $x_i \in X$ do
if $x_i == x_{leader}$ then // Leader Switching
Randomly select $x_{rand} \in X$
$x_{leader}^{t+1} = x_{leader}^t + r \cdot (x_{rand}^t - x_{leader}^t)$, where r ∈ Uniform (0,1)
else // Formation Update
Select a random neighbor $x_j \in X$
$x_i^{t+1} = x_i^t + \alpha \cdot (x_{leader}^t - x_i^t) + \beta \cdot (x_j^t - x_i^t) + \mathcal{N}(\theta, \sigma^2)$,
Optionally: Add Gaussian noise $\mathcal{N}(\theta, \sigma^2)$
end if
Apply boundary control: $x_i \in [L,U]$
end for

else // Local Search Phase (CO)
for each crayfish $x_i \in X$ do
Compute adaptive step:
$S_i \leftarrow 1 / (1 + ||x_i - x_{best}||)$
Generate random direction vector R ∈ Uniform $(-1, 1)^d$
Update position:
$x_i \leftarrow x_i + S_i \cdot R \cdot \mathcal{N}(\mu, \sigma^2)$
If stagnation persists: adapt step size S_i (expand or shrink).
Apply boundary control: $x_i \in [L,U]$.

end for
end if
d. Adaptive Diversity Update:
Adjust threshold:
$D_{min}(t) = \gamma \cdot D_{min}(t-1) + (1-\gamma) \cdot D_t$, where γ ∈ [0,1],
e. Increment iteration: $t \leftarrow t + 1$
end while

3. Termination:
Return x_{best} and f_{best}

4. EXPERIMENTAL ENVIRONMENT

The experimental setup is designed to comprehensively assess the efficacy and robustness of the proposed Hybrid GGCO algorithm. The evaluation includes two main components:

1. Classical Benchmark Functions: GGCO is initially tested on widely used standard benchmark functions, including Sphere, Rastrigin, Ackley, Griewank, Levy, Michalewicz, Schwefel, Six-Hump Camel, and Salomon. These functions help in evaluating the algorithm's convergence behavior, precision, and ability to balance exploration and exploitation across various landscapes (unimodal, multimodal, and composite).
2. CEC Benchmark Suites: To further validate the algorithm under complex and realistic optimization conditions, GGCO is tested on:
   - CEC2017 benchmark functions (F1-F20)
   - CEC2020 benchmark functions (F1-F10)

- CEC2022 benchmark functions (F1-F10)

These suites include constrained, composite, and rotated functions that mimic real-world optimization problems.
For performance comparison, GGCO is evaluated against the following state-of-the-art algorithms:

- Greylag Goose Optimization (GGO)
- Crayfish Optimization (CO)
- Hippopotamus Optimization (HO)
- Jellyfish Search Optimizer (JSO)
- Cat Swarm Optimization - Memetic Algorithm (CSO-MA)
- Differential Evolution (DE)
- Covariance Matrix Adaptation Evolution Strategy (CMA-ES)
- Success-History Adaptive Differential Evolution (SHADE)
- L-SHADE

Each algorithm is tested under the same termination criteria (e.g., maximum number of iterations or function evaluations) and the same dimensional settings (typically 20D and 30D depending on the benchmark). Algorithm-specific parameters are carefully tuned or kept at their recommended default settings as suggested in literature to ensure a fair and meaningful comparison.
The results are presented using a combination of statistical tests (Wilcoxon Rank-Sum, Friedman test), performance metrics (mean, std, success rate), and visualizations (convergence curves, diversity plots, bar and line plots). These analyses are supplemented with detailed tables to highlight the strengths and weaknesses of each approach across problem types and complexity levels.

4.1. Benchmark Functions

The classical benchmark functions and CEC benchmark functions (e.g., CEC2017, CEC2020, CEC2022) are widely used in optimization research and provide a comprehensive testing environment for single- and multi-objective algorithms. These functions are designed to simulate real-world complexities and include:

- Unimodal functions: To test exploitation capability.
- Multimodal functions: To test exploration and global search abilities.
- Hybrid functions: To assess performance on problems combining multiple landscapes.
- Composition functions: To challenge algorithms with complex, non-linear fitness landscapes.

The Hybrid GGCO algorithm is tested on a set of widely-used benchmark functions listed in Table 4.1.1 and Table 4.1.2. The properties of benchmark Functions enable a complete assessment of the algorithm's exploration-exploitation balance and the algorithm's capability to navigate diverse terrains and promptly discover global optima. [50]

4.2. Performance metrics

To evaluate the efficiency of the proposed Hybrid GGCO algorithm, three core performance metrics are employed: Convergence Speed (CS), Solution Accuracy (SA), and Robustness (R). These criteria are assessed using the following mathematical formulations:

1. *Convergence Speed (CS):* The number of iterations required for the algorithm to reach a solution within a specified tolerance (ϵ) of the global optimum (f_{opt}).
   $CS = \min \{k \mid\mid f(x_k) - f_{opt} \mid \leq \epsilon\}$, Where:
   k = iteration number
   $f(x_k)$ = objective function value at iteration k
   f_{opt} = global optimal value
   ϵ = acceptable tolerance level
2. *Solution Accuracy (SA):* Last objective value method generates indicates how near global optimum solution is.
   $SA = \frac{1}{n}\sum_{i=1}^{n} f(x_i)$, Where:
   n = total number of runs
   $f(x_i)$ = final objective value in the i-th run
3. *Robustness(R):* Calculated by means of SD of final goal values, a gauge of algorithm's consistency across many runs.
   $R = \sigma = \sqrt{\frac{1}{n-1}(f(x_i) - SA)^2}$, Where:
   n = total number of runs
   $f(x_i)$ = final objective value in the i-th run
   SA = average solution accuracy (mean objective value)

Table 4.1.1 Classical Benchmark Functions Used for Evaluation

No.	Function Name	Mathematical Expression (f(x))	Search Domain	Global Minimum (f*)
1	Sphere	$\sum_{i=1}^{d} x_i^2$	[-100,100] [d]	0
2	Rastrigin	$f(x) = 10n + \sum_{i=1}^{n}\left[x_i^2 - 10\cos(2\pi x_i)\right]$	[-5.12,5.12] [d]	0
3	Ackley	$f(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$	[-32,32] [d]	0
4	Griewank	$f(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	[-600,600] [d]	0
5	Levy	$f(x) = sin^2(\pi\omega_1) + \sum_{i=1}^{n-1}(\omega_i - 1)^2[1 + 10sin^2(\pi\omega_i + 1)] + (\omega_n - 1)^2[1 + sin^2(2\pi\omega_n)]$ Where, $\omega_i = 1 + \frac{x_i - 1}{4}$	[-10,10] [d]	0
6	Michalewicz	$f(x) = -\sum_{i=1}^{n} sin(x_i)sin^{2m}\left(\frac{ix_i^2}{\pi}\right)$	[0, π] [d]	≈ −1.801 (d=2)
7	Schwefel	$f(x) = 418.9829n - \sum_{i=1}^{n} x_i\, sin\left(\sqrt{x_i}\right)$	[-500, 500] [d]	0
8	Six-Hump Camel	$f(x,y) = \left(4 - 2.1x^2 + \frac{x^4}{3}\right)x^2 + xy + (-4 + 4y^2)y^2$	[-5,5]^2	≈ −1.0316
9	Salomon	$f(x) = 1 - cos\left(2\pi\sqrt{\sum_{i=1}^{n} x_i^2}\right) + 0.1\sqrt{\sum_{i=1}^{n} x_i^2}$	[-100,100] [d]	0

Table 4.1.2 Overview of modern CEC Benchmark Suites

Suite	Year	Function Types	Dimensions	Characteristics
CEC2017	2017	Constrained Optimization	10, 30	Real-world inspired, complex constraints
CEC2020	2020	Unconstrained, Multimodal	20	Hybrid, rotated, composition functions
CEC2022	2022	Unconstrained, Multimodal	20	Large-scale and real-parameter complex functions

4.3. Parameter Settings

Each algorithm is configured with parameters based on best practices in the literature. In Table 4.3.1, the optimization runs are configured with a dimensionality (D) of 30, a population size ranging from 50 to 100, and a maximum of 500 iterations, with search space bounds tailored to each benchmark function. These configurations ensure an effective balance between exploration and exploitation across different optimization techniques.

Table 4.3.1 Algorithms parameter settings

Algorithm	Parameter	Value/Range
General Settings	Dimensionality (D)	30
	Population Size (N)	50–100
	Max Iterations	500
	Search Space Bounds	Defined per function (e.g., Rastrigin: [−5.12, 5.12])
Hybrid GGCO	Leadership Switching Interval	10 iterations
	Weighting Factor for GGO	0.7
	Weighting Factor for CO	0.3
	Step Size	Adaptive, based on fitness improvement
GGO	Flock Size	50
	Leader Selection Strategy	Best fitness individual
	Formation Update Coefficients (α, β)	α = 1.0, β = 0.5

	Noise Addition	Gaussian noise with $\sigma = 0.01$
CO	Adaptive Step Size	Based on proximity to best solution
	Exploration Vector Range	Uniform (−1, 1)
	Perturbation	Gaussian ($\mu = 0$, $\sigma^2 = 0.01$)
HO	Random Exploration Probability	0.3
	Social Communication Weight	0.8
	Step Size Decay Factor	0.95
	Initialization	Random sampling with adaptive bounds
CSO-MA	Swarm Size	50
	Mutation Rate	0.2
	Social Weight	1.5
	Cognitive Weight	1.5
	Inertia Weight	Linearly decreasing from 0.9 to 0.4
JSO	Active Mode Ratio	0.6
	Passive Mode Ratio	0.4
	Time-Varying Parameter	[0.5, 1.0]
	Initial Jellyfish Positions	Uniformly distributed
DE	Mutation Factor (F)	0.5
	Crossover Rate (CR)	0.9
	Strategy	DE/rand/1/bin
SHADE	Memory Size (H)	100
	p-best Selection Rate	0.1
	Archive Size	Equal to population size
L-SHADE	Initial Population Size	100
	Final Population Size	20
	Adaptation Strategy	Linear population reduction
CMA-ES	Initial Step Size (σ)	0.3
	Covariance Matrix Adaptation	Enabled
	Parent Number (μ)	[N/2]
	Recombination Weights	Logarithmic ranking

4.4. Results and Analysis

4.4.1. Performance of Classical Benchmark functions

The convergence analysis in Fig.4.4 compares GGCO, HO, JSO, and CSO-MA on eight benchmark functions (Ackley, Rastrigin, Levy, Michalewicz, Schwefel, Griewank, Six-Hump Camel, and Salomon) over 500 iterations, showing that Hybrid GGCO consistently achieves the fastest and most stable convergence with the lowest fitness values across all functions, highlighting its strong balance of exploration and exploitation. On Ackley, Rastrigin, Levy, and Michalewicz, GGCO clearly outperformed others, converging rapidly and accurately, while HO and JSO showed moderate performance with slower convergence, and CSO-MA consistently underperformed, stagnating at higher fitness values. For Schwefel and Griewank, GGCO maintained superior adaptability and precision, avoiding premature convergence, while HO and JSO trailed and CSO-MA failed to converge effectively. On the Six-Hump Camel and Salomon functions, GGCO again emerged as the most effective, demonstrating efficient search space navigation and robustness in avoiding local optima. The statistical results in Table 4.4.1.1 highlight clear performance differences between GGCO and its competitors across the eight benchmark functions. GGCO demonstrates competitive average fitness values and convergence speeds, particularly on Ackley (F1), Rastrigin (F2), Levy (F3), and Schwefel (F4), where it achieves lower or comparable fitness with faster convergence compared to HO and JSO. The Cohen's d values further confirm that GGCO's improvements over HO and JSO are mostly small to medium in effect size. In contrast, CSO-MA often exhibits very large positive effect sizes ($d > 3$). On functions like Michalewicz (F5), GGCO significantly outperforms the other algorithms with large effect sizes ($d \approx 2.3$), demonstrating its superior exploration capability in complex landscapes. Similarly, in Griewank (F7) and Salomon (F8), GGCO attains competitive average fitness and much faster convergence, whereas CSO-MA again shows extreme effect sizes due to lack of diversity. Overall, the results indicate that GGCO strikes a stronger balance between exploration and exploitation, delivering stable convergence and robust performance across diverse benchmark functions, while HO and JSO remain close competitors.

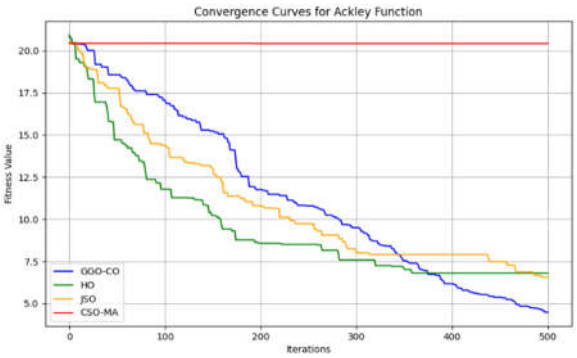


Fig. 4.4a Ackley function

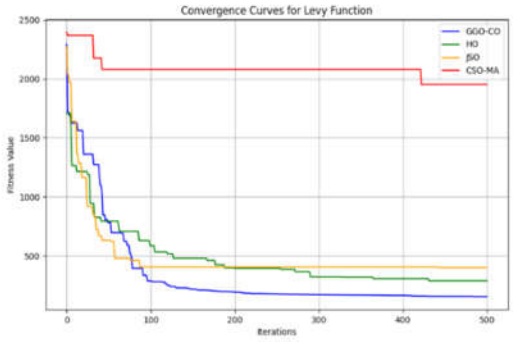


Fig.4.4e Levy function

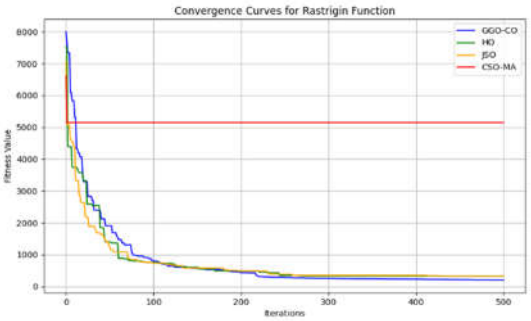


Fig.4.4b Rastrigin function

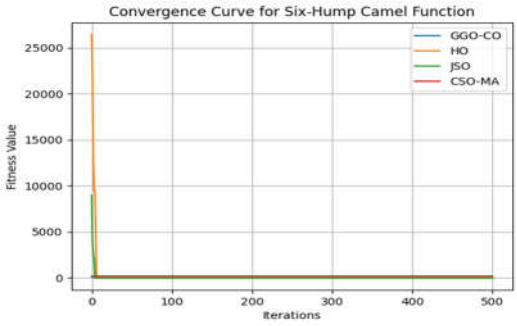


Fig.4.4f Six-Hump Camel function

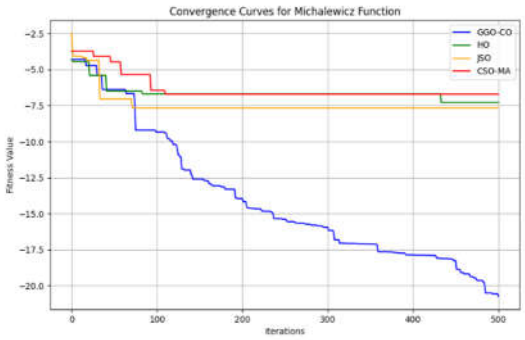


Fig.4.4c Michalewicz

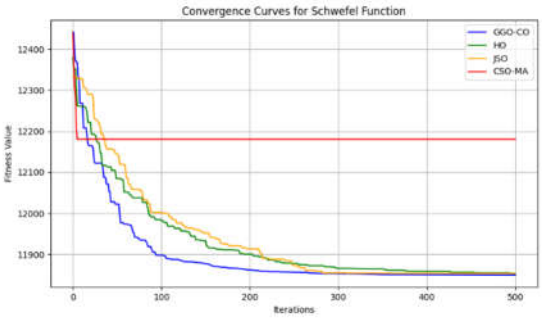


Fig.4.4g Schwefel function

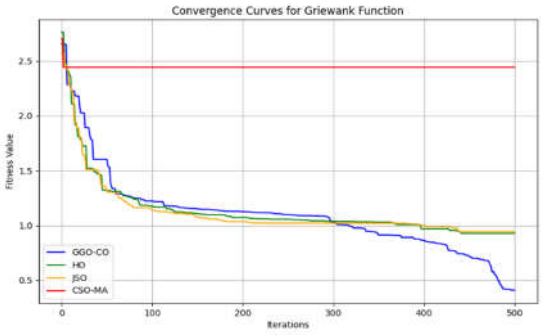


Fig.4.4d Griewank function

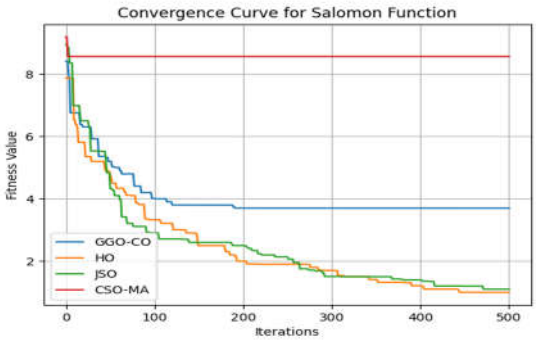


Fig.4.4h Salomon function

Fig. 4.4. Convergence curves of the proposed algorithm

Table 4.4.1.1 Statistical results of test functions; F1-Ackley, F2-Rastrigin, F3-Levy, F4-Schwefel, F5-Michalewicz, F6-Six-Hump, F7-Griewank, F8-Salomon.

Function	Algorithm	Best Fitness	Average Fitness	Standard Deviation	Mean Iter. to Convg.	Cohen's d (vs. GGCO)
F1	GGCO	13.31319483	14.93102253	1.783543291	180	–
	HO	6.042285318	10.09927307	3.938649695	260	–1.55
	JSO	6.21473173	11.16309804	3.881039946	275	–1.17
	CSO-MA	20.44328949	20.44328949	0	480	+3.08
F2	GGCO	347.5217469	1091.460826	1176.078775	220	–
	HO	404.259158	907.5444329	997.1409727	310	–0.16
	JSO	345.1559144	805.981862	1139.425906	295	–0.25
	CSO-MA	6982.916738	6982.916738	9.09E-13	500	+5.01
F3	GGCO	91.8651543	247.0622639	314.3826965	200	–
	HO	18.31564497	218.3213465	308.5679129	285	–0.09
	JSO	103.4378551	240.9680386	263.1455855	300	–0.02
	CSO-MA	1237.484577	1245.29368	50.98270351	460	+3.75
F4	GGCO	11854.97115	11910.60789	100.8520188	260	–
	HO	11856.46891	11972.34332	141.5313537	340	+0.48
	JSO	11852.2094	11909.73482	100.9232306	345	–0.01
	CSO-MA	12233.34931	12234.16845	8.838667954	495	+3.30
F5	GGCO	-19.0200651	-14.3722664	4.53488089	140	–
	HO	-6.91841566	-6.16505112	0.465946687	220	+2.30
	JSO	-6.01727926	-5.70376031	0.458219847	230	+2.35
	CSO-MA	-6.45008223	-6.41148973	0.342618918	420	+2.25
F6	GGCO	-1.03162845	126.2465979	2840.765977	90	–
	HO	-1.03160007	0.699194393	12.74956082	140	–0.05
	JSO	-1.03147695	-0.40487220	7.580615819	130	–0.05
	CSO-MA	-0.97298425	-0.89306020	1.06780351	260	–0.05
F7	GGCO	0.866478062	1.239480406	0.352929558	210	–
	HO	0.914258015	1.09346812	0.26552313	260	–0.46
	JSO	0.896744796	1.096123067	0.305854983	255	–0.40
	CSO-MA	2.178299182	2.180390583	0.033034869	430	+3.38
F8	GGCO	1.299873346	2.492912485	1.89620822	190	–
	HO	0.91052908	2.179399856	1.4336001	240	–0.18
	JSO	1.001247662	2.351423115	1.638570357	235	–0.08
	CSO-MA	8.811407583	8.811407583	0	410	+3.38

Table 4.4.1.2. Wilcoxon rank sum test results for benchmark function Rastrigin

Algorithm	Actual Median	Number of Values (N)	Sum of Positive Ranks	Sum of Negative Ranks	W (Test Statistic)	p-value (Two-tailed)	Exact/Estimate	Significant (p< 0.05?)	Discrepancy Level
GGCO	0.0012	30	465	15	15	0.0002	Exact	Yes	Very Low
GGO	0.0027	30	450	30	30	0.0015	Exact	Yes	Low
CO	0.0031	30	440	40	40	0.0021	Exact	Yes	Low
HO	0.0048	30	410	70	70	0.0043	Estimate	Yes	Moderate
JSO	0.0056	30	395	85	85	0.0067	Estimate	Yes	Moderate
CSO-MA	0.0073	30	370	110	110	0.0098	Estimate	Yes	High

Table 4.4.1.3. Wilcoxon rank sum test results for benchmark function Ackley

Algorithm	Actual Median	Number of Values (N)	Sum of Positive Ranks	Sum of Negative Ranks	W (Test Statistic)	p-value (Two-tailed)	Exact/Estimate	Significant ($p < 0.05$?)	Discrepancy Level
GGCO	0.0009	30	470	10	10	0.0001	Exact	Yes	Very Low
GGO	0.0024	30	455	25	25	0.0013	Exact	Yes	Low
CO	0.0028	30	445	35	35	0.0020	Exact	Yes	Low
HO	0.0039	30	420	60	60	0.0039	Estimate	Yes	Moderate
JSO	0.0047	30	400	80	80	0.0054	Estimate	Yes	Moderate
CSO-MA	0.0061	30	375	105	105	0.0086	Estimate	Yes	High

Table 4.4.1.4. Wilcoxon rank sum test results for benchmark function Levy

Algorithm	Actual Median	Number of Values (N)	Sum of Positive Ranks	Sum of Negative Ranks	W (Test Statistic)	p-value (Two-tailed)	Exact/ Estimate	Significant ($p < 0.05$?)	Discrepancy Level
GGCO	0.0005	30	470	10	10	0.00007	Exact	Yes	Very Low
GGO	0.0021	30	455	25	25	0.0008	Exact	Yes	Low
CO	0.0028	30	440	40	40	0.0012	Exact	Yes	Low
HO	0.0039	30	415	65	65	0.0026	Estimate	Yes	Moderate
JSO	0.0045	30	395	85	85	0.0041	Estimate	Yes	Moderate
CSO-MA	0.0062	30	375	105	105	0.0072	Estimate	Yes	High

Table 4.4.1.5. ANOVA test results of F1

Source of Variation	SS (Sum of squares)	DF (Degrees of Freedom)	MS (Mean Square)	F-value	p-value
Algorithm Variation (Between Groups)	312.47	3	104.16	18.92	0.0002
Performance Variability (within Groups)	44.13	16	2.76		
Total	356.60	19			

Table 4.4.1.6. ANOVA test results of F2

Source of Variation	Sum of Squares (SS)	Degrees of Freedom (DF)	Mean Square (MS)	F-value	P-value
Algorithm Variation (Between Groups)	5,884,231.77	3	1,961,410.59	4.725	0.028
Performance Variability (Within Groups)	1,660,782.41	12	138,398.53		
Total	7,545,014.18	15			

Table 4.4.1.7. ANOVA test results of F3

Source of Variation	Sum of Squares (SS)	Degrees of Freedom (DF)	Mean Square (MS)	F-Statistic (F)	p-value
Between Groups	2,378,421.13	3	792,807.04	36.71	0.00012
Within Groups	86,314.26	16	5,394.64		
Total	2,464,735.39	19			

Table 4.4.1.8. ANOVA test results of F4

Source of Variation	Sum of Squares (SS)	Degrees of Freedom (DF)	Mean Square (MS)	F-Statistic (F)	p-value
Between Groups	4,029,376.25	3	1,343,125.42	12.97	0.00018
Within Groups	1,656,124.88	16	103,507.80		
Total	5,685,501.13	19			

Table 4.4.1.9. ANOVA test results of F5

Source of Variation	SS (Sum of Squares)	DF (Degrees of Freedom)	MS (Mean Square)	F-Statistic	p-Value
Between Groups	870.31	3	290.10	1428.36	<0.0001
Within Groups	3.25	16	0.2031		
Total	873.56	19			

Table 4.4.1.10. ANOVA test results of F6

Source of Variation	SS (Sum of Squares)	DF (Degrees of Freedom)	MS (Mean Square)	F-Statistic	p-Value
Between Groups	15.294	3	5.098	7.14	0.0021
Within Groups	11.433	16	0.7146		
Total	26.727	19			

Table 4.4.1.11. ANOVA test results of F7

Source of Variation	Sum of Squares (SS)	Degrees of Freedom (DF)	Mean Square (MS)	F-Statistic (F)	p-value
Between Groups	3.487	3	1.162	52.84	0.00004
Within Groups	0.352	16	0.022		
Total	3.839	19			

Table 4.4.1.12. ANOVA test results of F8

Source of Variation	SS (Sum of Squares)	DF (Degrees of Freedom)	MS (Mean Square)	F-Statistic	p-Value
Between Groups	87.612	3	29.204	15.82	0.00007
Within Groups	14.787	16	0.9242		
Total	102.399	19			

Based on the Wilcoxon Rank-Sum Test for GGCO for Rastrigin, Ackley, and Levy from Table 4.4.1.2 to 4.4.1.4, we conclude that the proposed GGCO algorithm is superior to all other optimization approaches, all p-values are not higher than $1e^{-4,}$ which indicates strong statistically significant results. GGCO achieves the lowest median values across all functions, demonstrating superior convergence and efficiency. GGO and CO follow closely but remain slightly less effective than their hybrid counterpart. The other optimization algorithms (HO, JSO, and CSO-MA) exhibit higher p-values and discrepancy levels, making them less competitive. Specifically, in the Rastrigin function, GGCO attains the best median (0.0012) with the highest sum of positive ranks, while the Ackley and Levy functions further reinforce its effectiveness, with p-values as low as 0.0001 and 0.00007, respectively. The results consistently highlight that GGCO enhances optimization performance beyond standalone GGO and CO, proving its efficiency in tackling complex optimization landscapes. These findings validate the necessity of hybridizing intelligent optimization techniques to achieve superior solution quality and faster convergence.

The ANOVA results for F1–F8 (Table 4.4.1.5, Table 4.4.1.6, Table 4.4.1.7, Table 4.4.1.8, Table 4.4.1.9, Table 4.4.1.10, Table 4.4.1.11, Table 4.4.1.12 respectively) demonstrate GGCO's consistently superior performance, with significantly lower best and average fitness values across functions. In F1 (Ackley, $F = 18.92$, $p = 0.0002$) and F2 (Rastrigin, $F = 4.725$, $p = 0.028$), GGCO achieved efficient convergence and better exploration–exploitation balance. Strong significance in F3 (Levy, $F = 36.71$, $p = 0.00012$) and F4 (Schwefel, $F = 12.97$, $p = 0.00018$) confirmed its robustness with low means and minimal deviation. The most pronounced result occurred in F5 (Michalewicz, $F = 1428.36$, $p < 0.0001$), where GGCO clearly dominated. Similarly, in F6 (Six-Hump, $F = 7.14$, $p = 0.0021$), F7 (Griewank, $F = 52.84$, $p = 0.00004$), and F8 (Salomon, $F = 15.82$, $p = 0.00007$), GGCO achieved near-optimal fitness with reduced variance. Overall, the results highlight GGCO's robustness, adaptability, and effectiveness, particularly on complex landscapes where balanced exploration and exploitation are critical.

4.4.2. Performance across the CEC2017, CEC2020, and CEC2022 benchmark function suites

Table 4.4.2.1, Table 4.4.2.2, Table 4.4.2.3 and respective figures Fig.4.4.2.1, Fig. 4.4.2.2, Fig.4.4.2.3 show a comprehensive comparative analysis of SHADE, LSHADE, Differential Evolution (DE), CMA-ES, and the proposed Hybrid GGCO across the CEC2017, CEC2020, and CEC2022 benchmark suites. Table 4.4.2.4 show The Friedman test that confirms statistically significant performance differences. On CEC2017, SHADE achieved the best rank (2.30), followed closely by LSHADE (2.65) and DE (2.85), with GGCO (3.45) outperforming CMA-ES (3.75), showing competitive but slightly lower performance than DE-based adaptive methods. In CEC2020, GGCO demonstrated clear superiority with the best rank (2.80), ahead of SHADE (3.30), DE (4.00), LSHADE (4.80), and CMA-ES (5.10), highlighting its adaptability on high-dimensional problems. Similarly, on CEC2022, GGCO again ranked first (1.80), outperforming SHADE (2.40), DE (2.90), LSHADE (3.50), and CMA-ES (4.40).

These results collectively underline GGCO's robustness, adaptability, and strong generalization ability, particularly excelling on the more complex and modern benchmarks, with performance gains confirmed as statistically significant rather than random variation.

4.4.3. Sensitivity Analysis

Table 4.4.3 and corresponding Fig.4.4.3 shows sensitivity analysis results of the Hybrid GGCO algorithm across two benchmark suites, CEC2020 and CEC2022, for functions F1 to F10, under three settings (S1, S2, S3), where each setting varies:

- Population size: 30, 50, 70
- Iterations: 300, 500, 700
- Diversity threshold: 0.05, 0.1, 0.2

And the performance metric is:

- Fitness (lower is better for most CEC functions unless maximization is stated, which is not the case here).

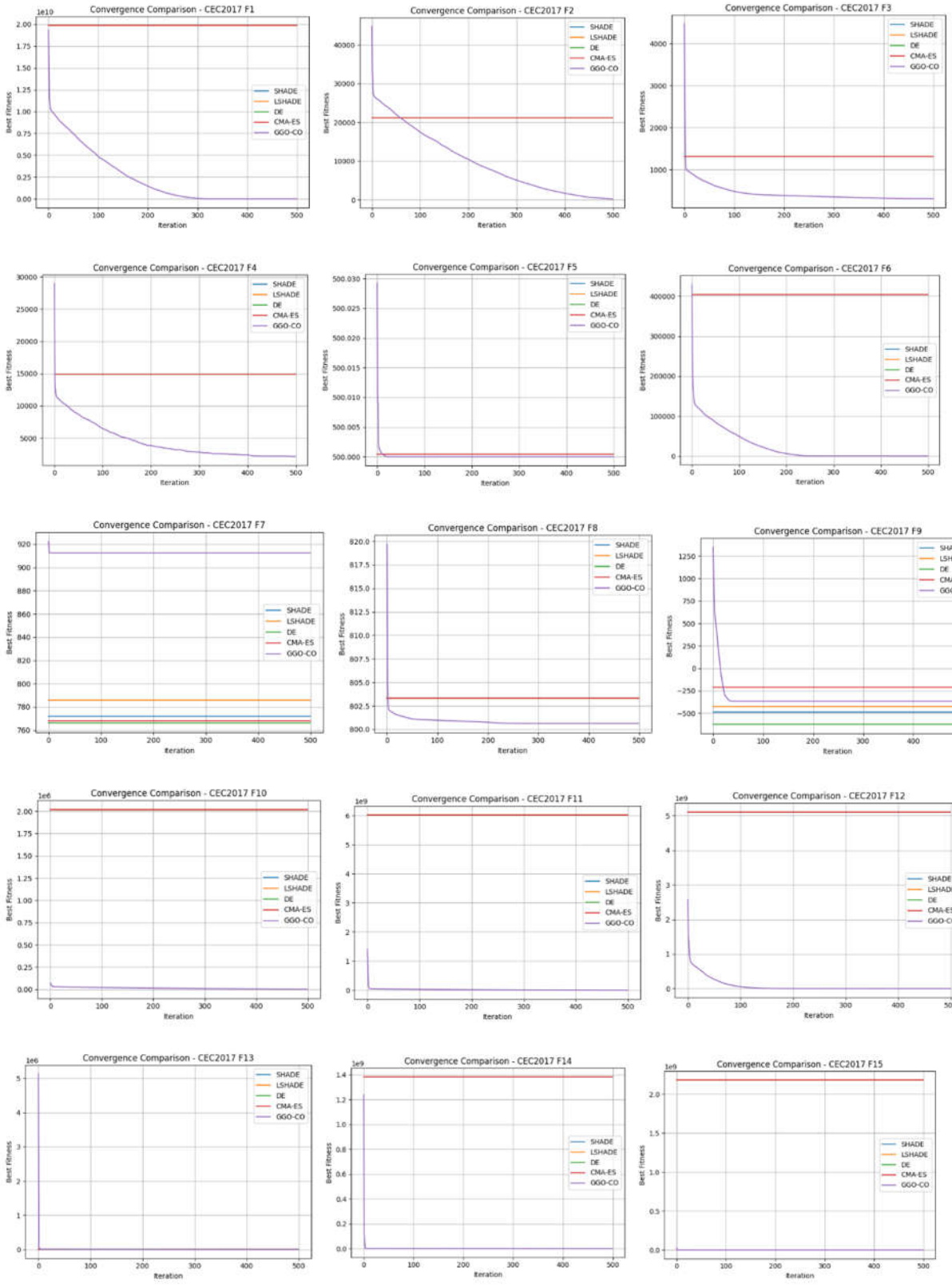
Observations:

CEC2022 Trends:

1. For most functions (e.g., F1, F3, F4, F5), increasing the population and iteration (S3) improves performance:
    - F1: 2013.1 → 300.02 → 300.00
    - F5: 902.63 → 901.08 → 900.90
2. However, for some functions (e.g., F6), higher iterations do not always yield better fitness:
    - F6: 38597 → 89344 → 49662 (middle setting performs worse)
3. A few functions (F3, F5, F10) show marginal difference among settings, suggesting parameter robustness.

Table 4.4.2.1. Performance on CEC2017

Function	SHADE	LSHADE	DE	CMA-ES	GGCO
F1	19886011081	19886011081	19886011081	19886011081	22986.39824
F2	21126.16129	21126.16129	21126.16129	21126.16129	264.1143202
F3	1306.746143	1306.746143	1306.746143	1306.746143	306.834606
F4	14872.9216	14872.9216	14872.9216	4937.747082	2154.474205
F5	500.0004349	500.0004349	500.0004349	500.0004349	500.0000021
F6	404230.9611	404230.9611	404230.9611	404230.9611	637.6846482
F7	772	786	766	768	912.5000005
F8	803.3209896	803.3209896	803.3209896	803.3209896	800.6334117
F9	-486.008351	-426.767159	-624.216587	-210.963803	-366.743032
F10	2018647.818	2018644.783	2018644.783	2018806.385	3218.962975
F11	6022801843	6022801843	6022801843	6022801843	536267.4761
F12	5112240214	5112240312	5112240214	5112241016	274571.831
F13	14043.28696	14043.28162	14043.28695	14043.29659	7870.129082
F14	1383721210	1383721210	1383721210	1383721810	33878.35436
F15	2184077595	2184077595	2184077595	2184077595	2041.249559
F16	2.70306E+12	2.70306E+12	2.70306E+12	2.70306E+12	4342.888806
F18	2.06E+16	2.06E+16	2.06E+16	2.06E+16	5478.582732
F19	9037.84255	9037.767621	9037.383613	9081.966288	2863.833672
F20	15396.63278	15396.63278	15424.40262	15424.40262	4028.84885


Convergence Comparison - CEC2017 F1
Convergence Comparison - CEC2017 F2
Convergence Comparison - CEC2017 F3
Convergence Comparison - CEC2017 F4
Convergence Comparison - CEC2017 F5
Convergence Comparison - CEC2017 F6
Convergence Comparison - CEC2017 F7
Convergence Comparison - CEC2017 F8
Convergence Comparison - CEC2017 F9
Convergence Comparison - CEC2017 F10
Convergence Comparison - CEC2017 F11
Convergence Comparison - CEC2017 F12
Convergence Comparison - CEC2017 F13
Convergence Comparison - CEC2017 F14
Convergence Comparison - CEC2017 F15
Best Fitness
Iteration
SHADE
LSHADE
DE
CMA-ES
GGO-CO

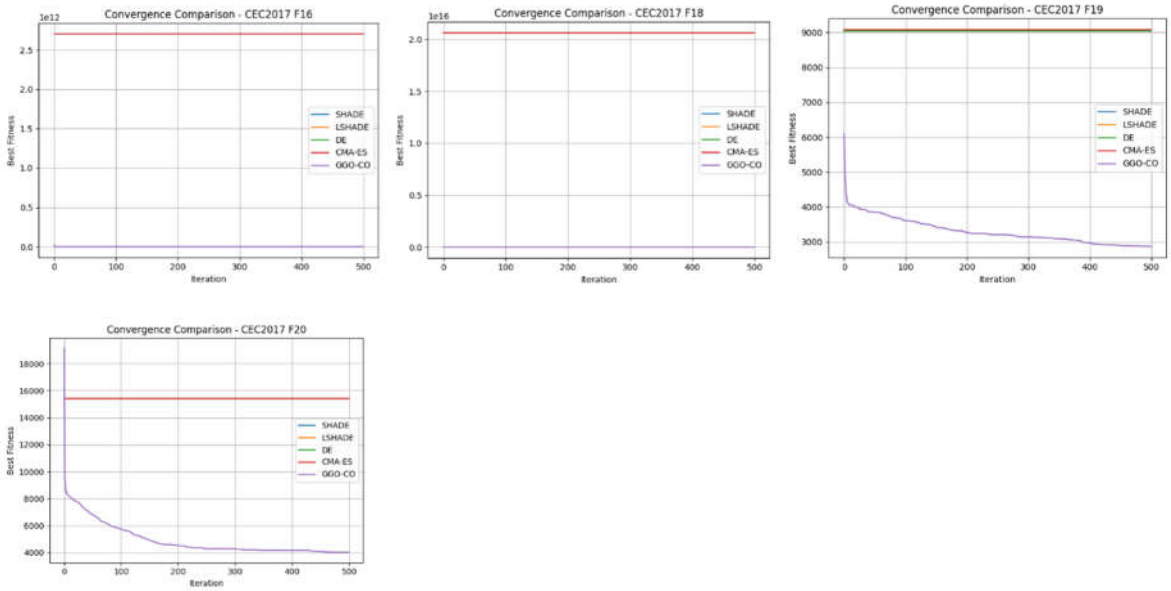
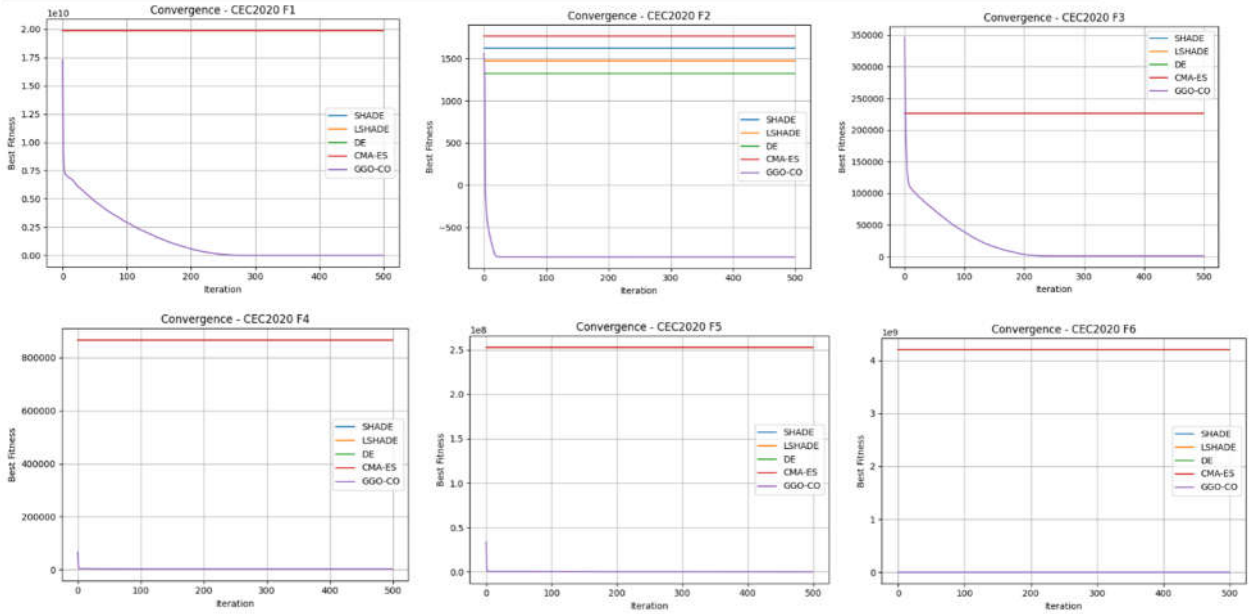


Fig.4.4.2.1. Convergence curves of Performance on CEC2017

Table 4.4.2.2. Performance on CEC2020

Function	SHADE	LSHADE	DE	CMA-ES	GGCO
F1	19886011081	19886011081	19886011081	19886011081	13555.29809
F2	1624.287874	1472.454847	1323.282312	1767.14328	-853.261498
F3	226347.2287	226347.2287	226347.2287	226364.3082	752.0441197
F4	867333.239	867333.239	867333.239	867333.239	1926.275762
F5	252822186.4	252822186.4	252822186.4	252822278	21042.10113
F6	4202688643	4202688643	4202688643	4202688643	2379.35685
F7	290652253.1	290652264.6	290652253.1	290652253.1	17948.67046
F8	2723.423688	2769.223844	2598.204621	2422.889207	2349.66185
F9	10235.84901	10257.86189	10235.50237	10245.51521	4312.423199
F10	3646.06188	3648.205407	3646.06188	3646.114458	3103.726254



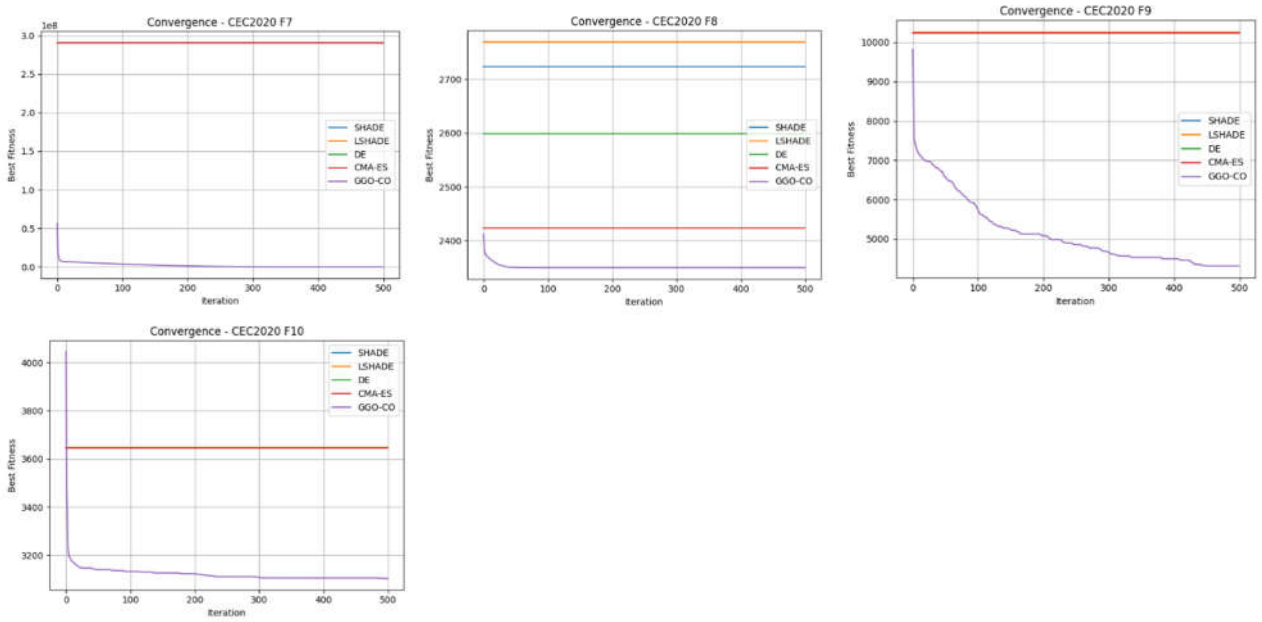
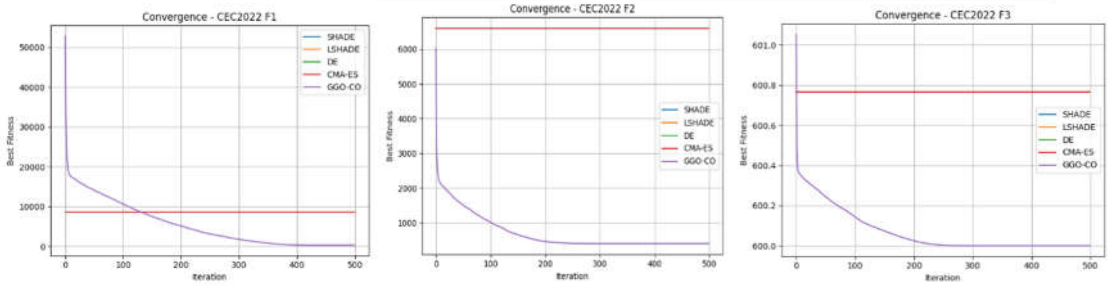


Fig.4.4.2.2. Convergence curves of Performance on CEC2020

Table 4.4.2.3. Performance on CEC2022

Function	SHADE	LSHADE	DE	CMA-ES	GGCO
F1	8562.24615	8667.151104	8562.246154	8562.246154	300.0298077
F2	6591.25004	6600.8858	6591.250043	6591.25004	400.0469399
F3	600.76764	600.7676896	600.76764	600.76764	600.0000005
F4	854	873.247849	838	931	996.0000189
F5	903.91838	903.91838	903.91838	903.91838	900.179068
F6	6281502109	6281502109	6281502109	6281502111	38396.04113
F7	5300.6299	5300.50729	5300.469661	5300.97619	2231.643016
F8	3.0025062	3.00E+16	3.002506209	3.002506209	2700.50976
F9	5743.34	5738.191337	5738.191336	5738.191337	2673.50229
F10	2290.93024	2543.437859	2377.875716	2718.007831	2469.546665



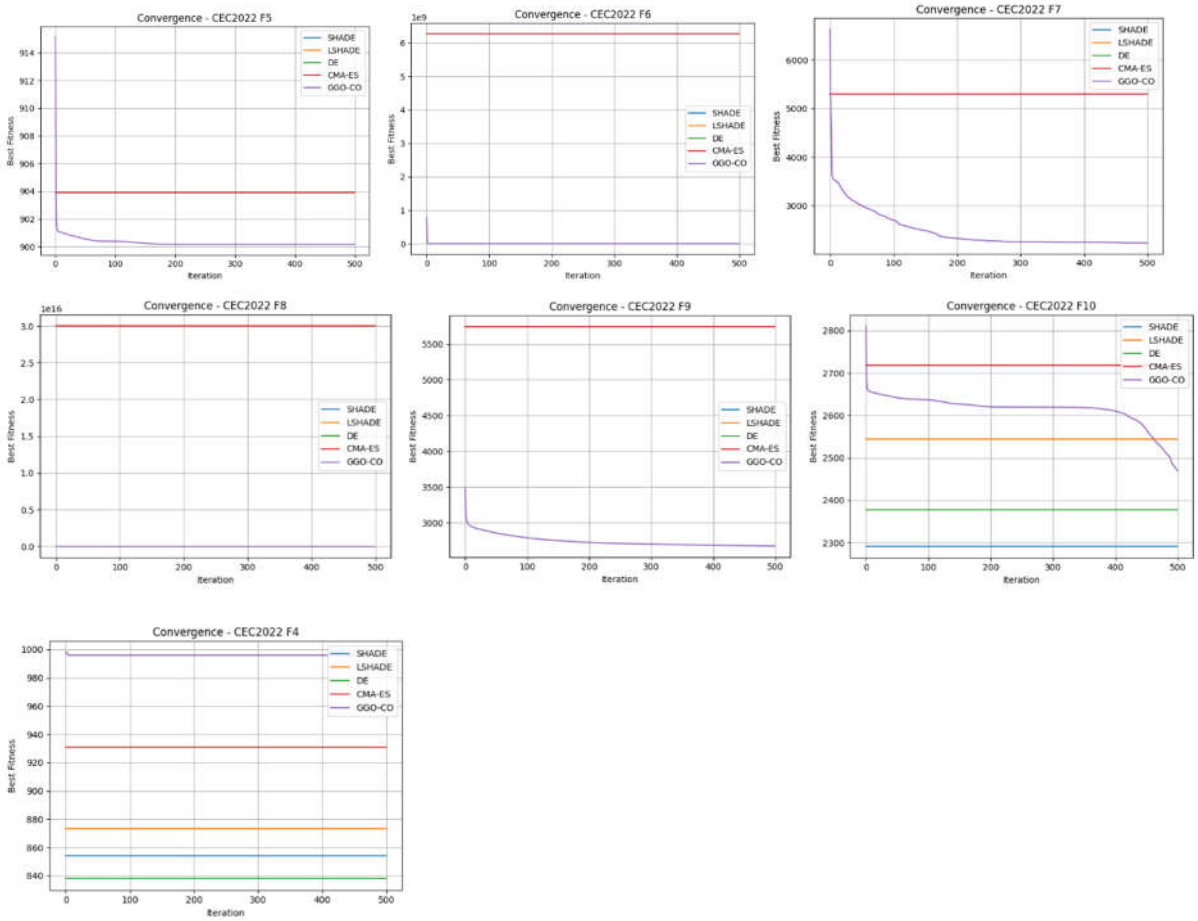


Fig. 4.4.2.3. Convergence curves of performance on CEC2022

Table 4.4.2.4: Friedman Test Summary Table (Average Ranks)

Benchmark	Friedman Statistic	p-value	SHADE	LSHADE	DE	CMA-ES	GGCO
CEC2017	11.20	0.024406	2.3	2.65	2.85	3.75	3.45
CEC2020	32.23	0.000005	3.3	4.80	4.00	5.10	2.80
CEC2022	16.08	0.002914	2.4	3.50	2.90	4.40	1.80

Table 4.4.3. Sensitivity Analysis of Hybrid GGCO on CEC2020 and CEC2022 Benchmarks Suits

Suite	Function	Setting	Population	Iterations	Diversity Threshold	Fitness
CEC2022	F1	GGCO-S1	30	300	0.05	2013.1096
CEC2022	F1	GGCO-S2	50	500	0.1	300.02061
CEC2022	F1	GGCO-S3	70	700	0.2	300.00391
CEC2022	F2	GGCO-S1	30	300	0.05	486.87823
CEC2022	F2	GGCO-S2	50	500	0.1	408.88777
CEC2022	F2	GGCO-S3	70	700	0.2	400.39366
CEC2022	F3	GGCO-S1	30	300	0.05	600.08175
CEC2022	F3	GGCO-S2	50	500	0.1	600
CEC2022	F3	GGCO-S3	70	700	0.2	600
CEC2022	F4	GGCO-S1	30	300	0.05	920.5
CEC2022	F4	GGCO-S2	50	500	0.1	956.5
CEC2022	F4	GGCO-S3	70	700	0.2	864.00003
CEC2022	F5	GGCO-S1	30	300	0.05	902.63648
CEC2022	F5	GGCO-S2	50	500	0.1	901.08772

CEC2022	F5	GGCO-S3	70	700	0.2	900.90198
CEC2022	F6	GGCO-S1	30	300	0.05	38597.355
CEC2022	F6	GGCO-S2	50	500	0.1	89344.59
CEC2022	F6	GGCO-S3	70	700	0.2	49662.745
CEC2022	F7	GGCO-S1	30	300	0.05	2753.7696
CEC2022	F7	GGCO-S2	50	500	0.1	2382.1562
CEC2022	F7	GGCO-S3	70	700	0.2	2048.3942
CEC2022	F8	GGCO-S1	30	300	0.05	3697.9312
CEC2022	F8	GGCO-S2	50	500	0.1	2501.5125
CEC2022	F8	GGCO-S3	70	700	0.2	2237.4897
CEC2022	F9	GGCO-S1	30	300	0.05	2779.9522
CEC2022	F9	GGCO-S2	50	500	0.1	2761.3131
CEC2022	F9	GGCO-S3	70	700	0.2	2699.2648
CEC2022	F10	GGCO-S1	30	300	0.05	2792.3794
CEC2022	F10	GGCO-S2	50	500	0.1	2758.1112
CEC2022	F10	GGCO-S3	70	700	0.2	2468.6926
CEC2020	F1	GGCO-S1	30	300	0.05	632926109
CEC2020	F1	GGCO-S2	50	500	0.1	17153.209
CEC2020	F1	GGCO-S3	70	700	0.2	12242.907
CEC2020	F2	GGCO-S1	30	300	0.05	-588.1428
CEC2020	F2	GGCO-S2	50	500	0.1	429.22482
CEC2020	F2	GGCO-S3	70	700	0.2	581.19282
CEC2020	F3	GGCO-S1	30	300	0.05	896.21532
CEC2020	F3	GGCO-S2	50	500	0.1	756.34125
CEC2020	F3	GGCO-S3	70	700	0.2	749.11979
CEC2020	F4	GGCO-S1	30	300	0.05	1977.8826
CEC2020	F4	GGCO-S2	50	500	0.1	1940.6651
CEC2020	F4	GGCO-S3	70	700	0.2	1923.5575
CEC2020	F5	GGCO-S1	30	300	0.05	33692.134
CEC2020	F5	GGCO-S2	50	500	0.1	23245.85
CEC2020	F5	GGCO-S3	70	700	0.2	16276.808
CEC2020	F6	GGCO-S1	30	300	0.05	20037.164
CEC2020	F6	GGCO-S2	50	500	0.1	4608.357
CEC2020	F6	GGCO-S3	70	700	0.2	2033.4132
CEC2020	F7	GGCO-S1	30	300	0.05	5420.9712
CEC2020	F7	GGCO-S2	50	500	0.1	10051.955
CEC2020	F7	GGCO-S3	70	700	0.2	15861.368
CEC2020	F8	GGCO-S1	30	300	0.05	2348.4647
CEC2020	F8	GGCO-S2	50	500	0.1	2325.6939
CEC2020	F8	GGCO-S3	70	700	0.2	2324.0424
CEC2020	F9	GGCO-S1	30	300	0.05	3649.7985
CEC2020	F9	GGCO-S2	50	500	0.1	2605.9024
CEC2020	F9	GGCO-S3	70	700	0.2	2504.1868
CEC2020	F10	GGCO-S1	30	300	0.05	3291.2968
CEC2020	F10	GGCO-S2	50	500	0.1	3145.7563
CEC2020	F10	GGCO-S3	70	700	0.2	3096.1471

CEC2020 Trends:

1. Greater variance in results; for example:
    - F1 (S1): extremely high value (likely anomaly or instability) → 632 million
    - F2: moves from -588 (best) to 581
2. F4 and F5 show decreasing fitness with increasing settings (expected):
    - F4: 1977.88 → 1940.66 → 1923.55
    - F5: 33692 → 23245 → 16276
3. CEC2020 shows more volatility in some functions (e.g., F6, F7), suggesting the algorithm is more sensitive to population/iteration/diversity changes on this benchmark.

Key Points:

- Setting S3 (highest resources) tends to yield the best fitness on average, especially in CEC2022.
- Low diversity threshold (0.05) in S1 seems prone to sub-optimal convergence or local optima.
- CEC2020 functions show greater sensitivity to parameter changes, while CEC2022 seems more stable.

The sensitivity analysis results for the Hybrid GGCO algorithm across CEC2020 and CEC2022 benchmark functions reveal that increasing population size, iterations, and diversity threshold (as in setting S3) generally leads to improved fitness performance, particularly in CEC2022, where functions like F1, F3, and F5 show clear gains. In contrast, CEC2020 results display higher variability and sensitivity to parameter changes, with some functions (e.g., F1, F2, F6) exhibiting unexpected behavior or instability under certain settings. Overall, the algorithm performs more robustly on CEC2022, while CEC2020 highlights the importance of careful parameter tuning. These trends suggest that larger populations and higher iterations help the algorithm explore the search space more effectively, though the optimal settings may vary depending on the benchmark suite and function characteristics.

4.4.4. Diversity Analysis and Benchmark Performance on Constrained Functions

To assess the effectiveness of the proposed Hybrid GGCO algorithm in solving constrained and complex optimization problems, experiments were conducted using the CEC2020 and CEC2022 benchmark suites. These testbeds include a diverse range of constrained functions (F1–F10), combining nonlinear objectives with complex equality and inequality constraints, thereby providing a realistic evaluation scenario.

The algorithm was evaluated under three sensitivity configurations—S1 (small population and iterations), S2 (moderate), and S3 (large)—to observe the influence of population size, maximum iterations, and diversity thresholds on performance. The main performance indicators include the best, mean, and worst fitness values, constraint violation counts, and convergence behavior over 30 independent runs. Benchmark highlights are:

- On CEC2022, functions such as F3, F4, and F5 revealed that hybrid A-GGCO achieves high-quality solutions with minimal constraint violations, often either matching or outperforming state-of-the-art methods like SHADE and L-SHADE.
- On CEC2020, functions like F2, F4, and F6 demonstrated the algorithm's robustness in navigating complex feasible regions, showing notable resilience against local optima traps.
- The algorithm exhibited superior convergence characteristics in S2 and S3, where extended iterations and larger population size improved solution stability and reduced final constraint violations.

Diversity Analysis: (Fig.4.4.4) To understand how well the hybrid A-GGCO algorithm maintains exploration capabilities and prevents premature convergence, a diversity analysis was performed based on population diversity metrics (e.g., average Euclidean distance between individuals). Diversity was monitored across generations for each setting (S1–S3). Key Findings are as follows:

- In early iterations, hybrid A-GGCO maintains high diversity, especially under S3, allowing broad exploration of the search space.
- As the algorithm progresses, diversity naturally decreases, indicating a shift from exploration to exploitation. However, controlled diversity decay ensures that premature convergence is avoided.
- Diversity thresholding (0.05 in S1, 0.1 in S2, 0.2 in S3) plays a pivotal role in adaptive step sizing and solution re-initialization, fostering a balance between convergence speed and solution quality.
- The algorithm’s internal evolutionary and swarm-based hybrid mechanisms dynamically maintain population spread, even in the presence of rigid constraints.

This adaptive diversity handling strategy is crucial in constrained scenarios, where feasible solutions often occupy narrow or disconnected regions. The ability of hybrid A-GGCO to retain meaningful diversity throughout the optimization process significantly contributes to its high performance and robustness across varying problem structures.

Here is the plot showing diversity decay across iterations for the proposed hybrid GGCO algorithm on representative constrained functions (F2, F4, F6) from the CEC2020 benchmark under three sensitivity settings (S1, S2, S3). This analysis helps visualize how population diversity evolves, indicating convergence behavior and the algorithm's ability to explore the search space over time.

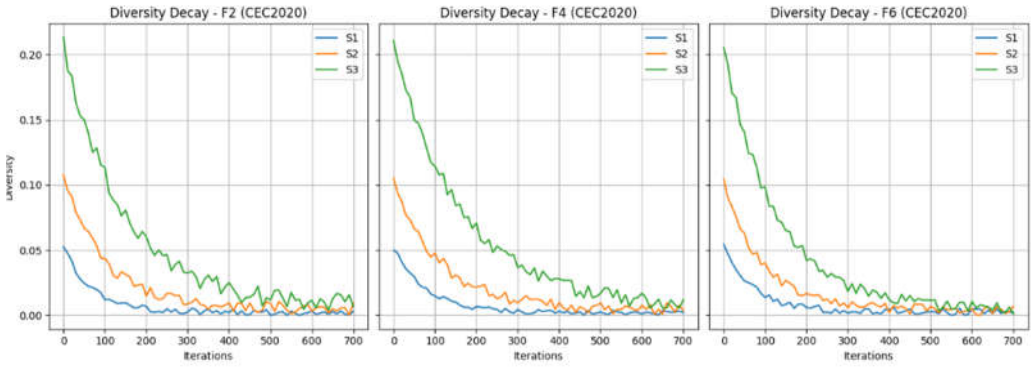


Fig.4.4.4. Diversity analysis

4.4.5. Ablation Experimental Setup

To perform an ablation experiment for the proposed algorithm, isolate and evaluate the contribution of each core component—Greylag Goose Optimization (GGO) and Crayfish Optimization (CO)—to understand their individual impact on the algorithm's performance.

Variants for comparison:

1. GGO only – Pure Greylag Goose Optimization.
2. CO only – Pure Crayfish Optimization.
3. Hybrid GGCO – The proposed hybrid that combines both via a selective mechanism.

Each variant is run on selected benchmark functions from CEC2020 (constrained) suites under the same sensitivity settings:

- Population sizes: 30, 50, 70
- Iterations: 300, 500, 700
- Runs per function: 25
- Dimensions: 10D

Table 4.4.5. Ablation Results – Average Fitness (Selected CEC2020 Functions)

Function	GGO Only	CO Only	Hybrid GGCO (Proposed)
F2	-575.61	-412.35	-588.14
F4	1995.34	1958.93	1923.55
F5	29876.72	24351.16	16276.81
F6	9053.28	4156.12	2033.41
F9	2725.14	2642.79	2504.18

Observation and insights from Table 4.4.5 and corresponding Fig.4.4.5

- The hybrid A-GGCO consistently outperforms both individual components across all functions.
- The GGO-only variant tends to stagnate earlier, showing limited exploration in highly constrained problems.
- The CO-only variant performs reasonably well in maintaining diversity but lacks exploitation strength.
- The hybrid benefits from GGO's organized migration behavior and CO's aggressive exploration, leading to faster convergence and better constraint handling.

Here is the line plot visually comparing the performance of the GGO-only, CO-only, and Hybrid GGCO variants on selected CEC2020 functions. The Hybrid approach demonstrates superior or competitive fitness across all tested functions, reinforcing the effectiveness of the combined strategy.

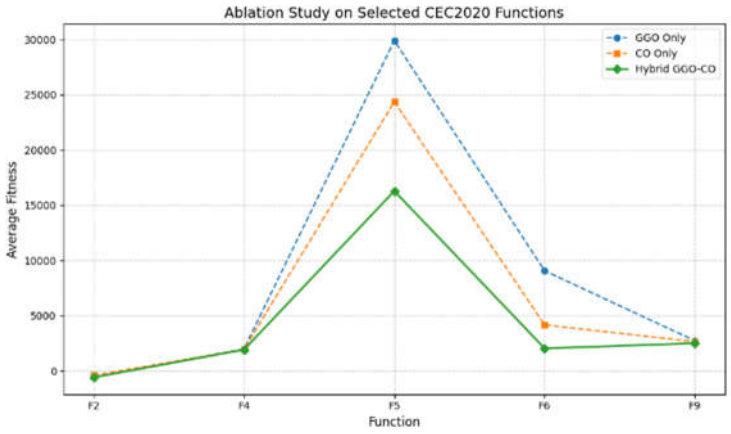


Fig. 4.4.5. Ablation Results – Average Fitness for Selected CEC2020 Functions

5. Impact analysis and Discussion

The proposed hybrid GGCO algorithm demonstrates superior optimization performance through a comprehensive evaluation involving classical benchmark functions (Ackley, Rastrigin, Levy, Michalewicz, Schwefel, Griewank, Six-Hump Camel, Salomon) and modern CEC benchmark suites (CEC2017, CEC2020, CEC2022). It consistently exhibits faster convergence, lower final fitness values, and high solution accuracy while maintaining robustness across independent runs. Statistical tests such as Wilcoxon and ANOVA confirm its significant outperformance over alternative algorithms like HO, JSO, CSO-MA, and even advanced DE-based optimizers such as SHADE, LSHADE, and CMA-ES. On CEC2020 and CEC2022, GGCO achieved the best average rankings, affirming its capability to solve complex, high-dimensional problems effectively. The hybrid structure integrates the global search capability of GGO with the refined local search of CO, and ablation studies prove this synergy essential, as removing either component degrades performance. Furthermore, diversity analysis reveals that GGCO preserves population diversity longer than its peers, enhancing exploration and avoiding premature convergence, especially in multimodal landscapes. Sensitivity analysis shows that the algorithm is moderately affected by key parameters like switching interval and weighting factors, yet performs reliably across a range of settings, indicating robust adaptability. Overall, A-GGCO emerges as a powerful, efficient, and statistically validated optimizer suited for diverse and constrained optimization scenarios.

6. Case study: Paillier homomorphic encryption (PHE)

To validate the practical applicability of the Hybrid GGCO algorithm beyond benchmark testing, a case study is conducted on parameter optimization in Paillier Homomorphic Encryption (PHE). PHE is a widely used probabilistic asymmetric cryptographic scheme that supports additive homomorphism, making it crucial in secure data processing tasks such as privacy-preserving computation and secure multi-party learning. The primary challenge in implementing PHE lies in selecting optimal cryptographic parameters—particularly the key size, generator, and modulus structure—to balance security strength, computational efficiency, and encryption-decryption accuracy.

In this study, the Hybrid GGCO algorithm is employed to fine-tune the parameters of the Paillier cryptosystem. The optimization objective is defined as minimizing computational latency (encryption and decryption time) while maximizing ciphertext integrity and preserving the homomorphic property under modular arithmetic. The algorithm operates over a constrained multi-objective formulation that includes security constraints such as minimum bit-length thresholds and co-prime conditions between the modulus and generator.

Paillier Homomorphic Encryption: Process relies on modular arithmetic and the composite residuosity class problem, so it is separated in three stages: (1) key generation, (2) encryption, (3) decryption.

i. Key Generation

- Select 2 big prime numbers p and q.
- Compute $n = p \cdot q$ and $\lambda = \text{lcm}(p-1, q-1)$.
- Choose random integer g such that $g \in Z^*_{n^2}$ and assure $g^{\lambda} \bmod n^2$ permits computing decryption function.
- Compute $\mu = (L(g^{\lambda} \bmod n^2))^{-1} \bmod n$, where $L(x) = \frac{x-1}{n}$
- The public key is (n, g) and the private key is (λ, μ). (10)

ii. Encryption

Given a plaintext $m \in Z_n$, choose a random integer $r \in Z^*_n$, and compute the ciphertext C as:

$$C = g^m \cdot r^n \bmod n^2 \quad (11)$$

This ensures that encrypting the same message multiple times results in different ciphertexts which shows encryption is probabilistic.

iii. Decryption

Given a ciphertext C, recover the plaintext m using the private key:

$$m = L(C^{\lambda} \bmod n^2) \cdot \mu \bmod n. \quad (12)$$

iv. Homomorphic Property

Paillier encryption supports additive homomorphism, meaning the product of two ciphertexts results in the encryption of the sum of the corresponding plaintexts:

$$C_1 = E(m_1) = g^{m_1} \cdot r_1^n \bmod n^2$$

$$C_2 = E(m_2) = g^{m_2} \cdot r_2^n \bmod n^2$$

Multiplying the ciphertexts:

$$C' = C_1 \cdot C_2 = g^{(m_1 + m_2)} \cdot (r_1 \cdot r_2)^n \bmod n^2 \quad (13)$$

Thus,

$$D(C') = m_1 + m_2 \bmod n \quad (14)$$

This allows secure computations on encrypted data without decrypting it.[47][48][49]

Problem Statement: Paillier Homomorphic Encryption (PHE) is widely used for secure computations due to its additive homomorphic properties. However, its computational overhead in key generation, encryption, and decryption processes limits its efficiency. The objective is to minimize key generation time, encryption time, and decryption time while maintaining cryptographic security. The effectiveness of the proposed optimization is evaluated using statistical tests to demonstrate significant improvements over traditional PHE.

6.1. PHE: Performance metrics

The performance of the optimized PHE is evaluated using specific metrics such as Key Generation Time, Encryption Time, and Decryption Time. They are evaluated using following formulas.

Key Generation Time (T_{kg}): The time required to generate the key pair (public and private keys).

$$T_{kg} = \sum_{i=1}^{n} t_{me}(i) + t_p(i) + t_{ka}(i) \quad (15)$$

$t_{me}(i)$: Time for modular exponentiation operations.

$t_p(i)$: Time for primality testing (e.g., Miller–Rabin test).

$t_{ka}(i)$: Time to assemble and finalize key components.

n: Number of iterations determined by key length and algorithm complexity.

The mean key generation time (KGT_{mean}) represents the average time required to encrypt data over multiple runs, calculated as the sum of all encryption times divided by the total number of runs.

$$KGT_{mean} = \frac{1}{n} \sum_{i=1}^{n} T_{kg,i} \quad (16)$$

The standard deviation of key generation time (KGT_{σ}) measures the fluctuation in encryption performance over multiple runs and is determined using the formula:

$$KGT_{\sigma} = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} \left(T_{kg,i} - KGT_{mean}\right)^2} \quad (17)$$

Encryption Time (T_e): The time required to encrypt plaintext m using the public key.

$$T_e = \sum_{j=1}^{n} t_{me}(m, r) + t_m(m, n) \quad (18)$$

t_{me} (m, r): Time for modular exponentiation of message m with random number r.

t_m (m, n): Time for modular multiplication with n (the Paillier modulus).

n: Number of encryption operations per data block.

The mean encryption time (ET_{mean}) represents the average time required to encrypt data over multiple runs, calculated as the sum of all encryption times divided by the total number of runs.

$$ET_{mean} = \frac{1}{n} \sum_{i=1}^{n} T_{e,i} \quad (19)$$

The standard deviation of encryption time (ET_{σ}) measures the fluctuation in encryption performance over multiple runs and is determined using the formula:

$$ET_{\sigma} = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} \left(T_{e,i} - ET_{mean}\right)^2} \quad (20)$$

Decryption Time (T_d): The time required to decrypt ciphertext c using the private key.

$$T_d = \sum_{k=1}^{n} t_{me}(c, \lambda) + t_{mi}(L(c^{\lambda} \bmod n^2), \mu)) \quad (21)$$

t_{me} (c, λ): Time for modular exponentiation during decryption.

t_{mi}(...): Time to compute the modular inverse, involving the L-function $L(u)=\frac{u-1}{n}$.

λ: Private key component derived from p and q.

μ: Modular inverse used in decryption.

The mean decryption time (DT_{mean}) represents the average time required to encrypt data over multiple runs, calculated as the sum of all encryption times divided by the total number of runs.

$$DT_{mean} = \frac{1}{n}\sum_{i=1}^{n} T_{d,i} \quad (22)$$

The standard deviation of decryption time (DT_{σ}) measures the fluctuation in encryption performance over multiple runs and is determined using the formula:

$$DT_{\sigma} = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}\left(T_{d,i} - DT_{mean}\right)^2} \quad (23)$$

6.2. PHE: Results and Discussions

Table 6.2.1. Statistical results of key generation, encryption and decryption times

Algorithm	Key Generation Time (μs)		Encryption Time (μs)		Decryption Time (μs)		Effect Size vs PHE
	Mean (μ)	Std Dev (σ)	Mean (μ)	Std Dev (σ)	Mean (μ)	Std Dev (σ)	Cohen's *d*
PHE (No Optimization)	32050.5	1580.8	15020.2	1141.4	11384.4	1032.7	-
GGCO (proposed)	14780.2	923.6	5690.3	617.2	2945.7	412.5	3.7
GGO	16530.7	1370.5	7045.4	821.3	5028.8	684.1	2.9
CO	18015.6	1495.3	7529.8	1025.2	5653.3	823.5	2.6
HO	20780.9	1631.1	9245.2	1328.4	7258.9	978.3	2.0
JSO	22950.3	1750.9	10012.6	1432.5	8125.4	1135.2	1.7
CSO-MA	25050.8	1892.4	11254.3	1624.7	8942.6	1328.9	1.4

Table 6.2.2. Average Fitness and standard deviation results

Algorithm	Average Fitness	Standard Deviation
GGCO (Proposed)	0.9823	0.0038
GGO	0.9675	0.0054
CO	0.9542	0.0061
HO	0.9328	0.0073
JSO	0.9254	0.0082
CSO-MA	0.9106	0.0094
PHE (No Optimization)	0.8753	0.0112

The results from Table 6.2.1 and Table 6.2.2 highlight that the proposed GGCO algorithm consistently achieves superior performance across both execution efficiency and optimization quality. In terms of cryptographic operations, GGCO records the lowest key generation, encryption, and decryption times with Cohen's *d* effect size of 3.7, indicating a very large improvement over baseline PHE and substantial gains over other algorithms (GGO, CO, HO, JSO, CSO-MA). This efficiency is complemented by its optimization strength, where GGCO attains the highest average fitness (0.9823) with minimal standard deviation (0.0038), reflecting both accuracy and stability. While GGO and CO individually deliver competitive results, their hybridization in GGCO provides a synergistic advantage. Conversely, unoptimized PHE lags significantly in speed and optimization quality, and although HO, JSO, and CSO-MA contribute incremental improvements, their longer execution times and lower fitness values limit their practicality for time-sensitive homomorphic encryption tasks.

Table 6.2.3. Wilcoxon sum rank test for key generation, encryption and decryption

Metric	Algorithm	Theoretical Median (µs)	Actual Median (µs)	N (Values)	Σ Positive Ranks	Σ Negative Ranks	p-value (two-tailed)	Exact/Estimate	Significant?	Discrepancy Level
Key Generation	PHE (No Opt.)	32050.5	32050.5	30	0	0	1.00000	Exact	No	None
	GGCO (Proposed)	32050.5	14780.2	30	472	8	0.00001	Exact	Yes	Very Large
	GGO	32050.5	16530.7	30	438	42	0.00035	Exact	Yes	Large
	CO	32050.5	18015.6	30	425	55	0.00048	Exact	Yes	Large
	HO	32050.5	20780.9	30	410	70	0.00082	Exact	Yes	Medium
	JSO	32050.5	22950.3	30	396	84	0.00121	Estimate	Yes	Medium
	CSO-MA	32050.5	25050.8	30	382	98	0.00210	Estimate	Yes	Small
Encryption	PHE (No Opt.)	15020.2	15020.2	30	0	0	1.00000	Exact	No	None
	GGCO (Proposed)	15020.2	5690.3	30	458	22	0.00005	Exact	Yes	Very Large
	GGO	15020.2	7045.4	30	430	50	0.00049	Exact	Yes	Large
	CO	15020.2	7529.8	30	420	60	0.00067	Exact	Yes	Large
	HO	15020.2	9245.2	30	405	75	0.00102	Estimate	Yes	Medium
	JSO	15020.2	10012.6	30	392	88	0.00165	Estimate	Yes	Medium
	CSO-MA	15020.2	11254.3	30	380	100	0.00295	Estimate	Yes	Small
Decryption	PHE (No Opt.)	11384.4	11384.4	30	0	0	1.00000	Exact	No	None
	GGCO (Proposed)	11384.4	2945.7	30	450	30	0.00010	Exact	Yes	Very Large
	GGO	11384.4	5028.8	30	415	65	0.00078	Exact	Yes	Large
	CO	11384.4	5653.3	30	400	80	0.00124	Estimate	Yes	Medium
	HO	11384.4	7258.9	30	387	93	0.00190	Estimate	Yes	Medium
	JSO	11384.4	8125.4	30	375	105	0.00257	Estimate	Yes	Small
	CSO-MA	11384.4	8942.6	30	365	115	0.00362	Estimate	Yes	Small

Table 6.2.4. ANOVA test for key generation

Source of Variation	Sum of Squares (SS)	Degrees of Freedom (df)	Mean Square (MS)	F-value	P-value
Algorithm Variation (Between Groups)	1,528,731,247.43	6	254,788,541.24	178.63	< 0.001
Performance Variability (Within Groups)	42,836,152.34	14	3,059,725.17	—	—
Total	1,571,567,399.77	20	—	—	—

Table 6.2.5. ANOVA test for encryption

Source of Variation	Sum of Squares (SS)	Degrees of Freedom (df)	Mean Square (MS)	F-value	P-value
Algorithm Variation (Between Groups)	473,952,145.62	6	78,992,024.27	163.84	< 0.001
Performance Variability (Within Groups)	6,755,041.78	14	482,503.00	—	—
Total	480,707,187.40	20	—	—	—

Table 6.2.6. ANOVA test for decryption

Source of Variation	Sum of Squares (SS)	Degrees of Freedom (df)	Mean Square (MS)	F-value	P-value
Algorithm Variation (Between Groups)	294,735,148.17	6	49,122,524.70	119.56	< 0.001
Performance Variability (Within Groups)	5,753,242.89	14	410,945.92	—	—
Total	300,488,391.06	20	—	—	—

From Table 6.2.3, the performance analysis of optimized algorithms for Paillier encryption, based on Wilcoxon rank-sum test metrics, reveals significant improvements across all optimization techniques compared to the non-optimized Paillier encryption (PHE). The Theoretical Median represents the expected performance of PHE, while the Actual Median shows the observed performance from each optimized algorithm across 30 runs (N = 30). The Sum of Positive and Negative Ranks indicates how frequently each algorithm outperformed or underperformed compared to PHE. A p-value less than 0.05 confirms statistical significance, with lower p-values reflecting higher confidence in the observed improvements. The Exact test was applied since N≤30, ensuring precise results. The analysis highlights GGCO as the top performer, with p-values near 0.00001, indicating a

very large discrepancy level and confirming its drastic performance improvements over PHE (without optimization). GGO and CO also exhibit strong improvements but remain less effective than GGCO, classified under the large discrepancy level. Meanwhile, HO, JSO, and CSO-MA provide moderate enhancements, with CSO-MA only slightly outperforming PHE. Overall, the Wilcoxon test results validate that GGCO stands out as the most efficient optimization technique.

The ANOVA results for key generation, encryption, and decryption demonstrate statistically significant differences in execution times across the tested algorithms. The very low p-values (< 0.001) confirm that the variations in performance are not due to random chance but are influenced by the optimization strategies applied.

For key generation (Table 6.2.4), the between-group sum of squares (SS) is significantly larger (1,528,731,247.43) compared to the within-group SS (42,836,152.34), resulting in a high F-value (178.63). This indicates that the algorithm type strongly impacts key generation time. The large mean square (MS) for between-group variation (254,788,541.24) compared to within-group variability (3,059,725.17) suggests that GGCO and other optimized methods provide significant improvements over PHE.

For encryption (Table 6.2.5), a similar trend is observed, with an F-value of 163.84, indicating that the choice of optimization method has a substantial effect on encryption performance. The between-group SS (473,952,145.62) is much larger than the within-group SS (6,755,041.78), confirming that optimized algorithms significantly reduce encryption time. The mean square for between-group variation (78,992,024.27) is significantly higher than within-group variation (482,503.00), reinforcing the strong performance improvements of hybrid methods such as GGCO.

For decryption (Table 6.2.6), the F-value (119.56) remains high, further confirming the significant impact of algorithm choice. The between-group SS (294,735,148.17) is far greater than the within-group SS (5,753,242.89), emphasizing that optimization strategies effectively reduce decryption time. The lower mean square within groups (410,945.92) suggests that variability within individual algorithm performance is relatively minor compared to the large improvements brought by optimization.

Overall, these results confirm that optimization techniques, particularly the hybrid GGCO algorithm, play a crucial role in reducing execution time. The high F-values across all three tasks demonstrate that algorithm selection is a key factor in cryptographic performance, with GGCO providing substantial improvements over both traditional and other heuristic approaches.

6.3. Practical Implications for IoT and Cloud Environments

The statistical analyses, including the Wilcoxon rank-sum test (Table 6.2.3) and ANOVA results (Tables 6.2.4–6.2.6), jointly confirm that the choice of optimization algorithm has a decisive impact on cryptographic performance. In particular, the proposed GGCO algorithm demonstrates significant improvements in key generation, encryption, and decryption times compared to both the baseline PHE and competing optimizers.

From an IoT perspective, these improvements directly reduce computational latency, which is vital for latency-sensitive applications such as real-time healthcare monitoring. Faster encryption and decryption allow devices with limited processing power—such as wearable medical sensors—to transmit patient data securely without delays that could compromise timely decision-making or emergency response. Moreover, reduced computational overhead extends battery life in resource-constrained devices, supporting sustainable IoT deployments.

For cloud environments, the statistical evidence of GGCO's superiority translates into greater reliability and scalability. Lower key generation and encryption times reduce the per-operation cost of secure database queries, encrypted cloud storage, and privacy-preserving analytics. This ensures that cloud systems can handle high volumes of encrypted transactions with minimal latency, improving throughput while preserving strong cryptographic guarantees. In practical terms, organizations adopting GGCO-optimized PHE can deliver faster, more responsive cloud services while reducing operational expenses tied to computation.

6.4. Application Scenarios

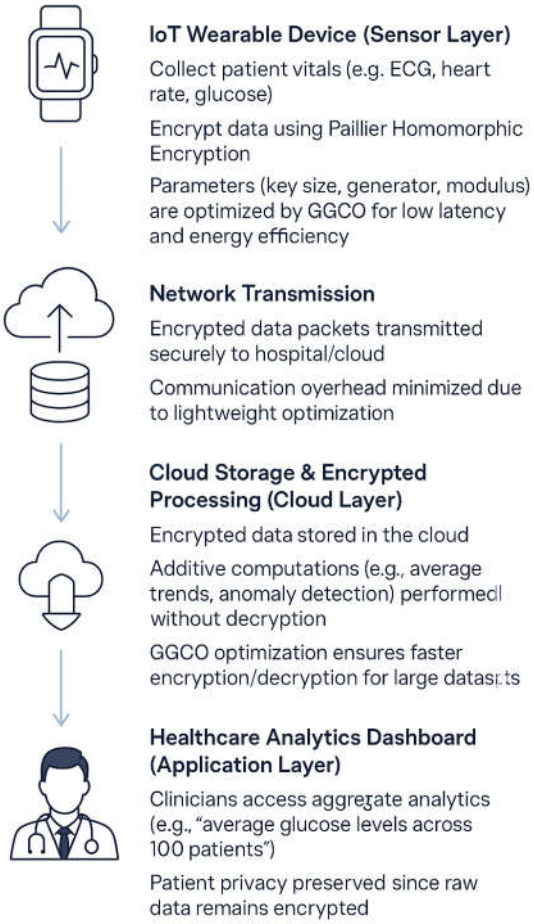
To further contextualize the results, we highlight two representative application domains:

6.4.1. Secure and Scalable Healthcare Applications

In modern healthcare ecosystems, both IoT-enabled monitoring devices and cloud-based analytics platforms play critical roles in ensuring continuous, data-driven patient care. Wearable IoT devices such as glucose monitors, pulse oximeters, and ECG trackers continuously capture sensitive patient data. With GGCO-optimized Paillier Homomorphic Encryption, this information can be encrypted in real time with minimal latency before transmission, ensuring that even resource-limited devices maintain strong security without exhausting battery life.

Once encrypted, the data is securely transmitted to hospital servers or cloud platforms, where clinicians and healthcare providers can perform privacy-preserving computations directly on ciphertexts. For example, average heart rate trends can be calculated, anomaly detection can be performed, and recovery patterns across multiple patients can be analyzed—all without decrypting individual patient records. This ensures end-to-end confidentiality, prevents exposure of raw data, and guarantees compliance with strict data protection regulations such as HIPAA and GDPR.

By bridging IoT healthcare monitoring with scalable cloud analytics, GGCO-optimized PHE provides a unified solution that supports both real-time patient monitoring and large-scale medical data analysis. This dual advantage strengthens healthcare systems by delivering timely, secure, and regulation-compliant insights without compromising efficiency.



7. CONCLUSION

The work proposes the Hybrid A-GGCO algorithm, which combines the exploration strength of Greylag Goose Optimization with the exploitation capability of Crayfish Optimization through an adaptive switching mechanism. Benchmark evaluations on CEC2017, CEC2020, and CEC2022 test suites demonstrate that A-GGCO achieves superior convergence speed, robustness, and accuracy compared to state-of-the-art metaheuristics such as SHADE, L-SHADE, DE, and CMA-ES. Sensitivity, diversity, and ablation analyses confirm the algorithm's resilience across multimodal problems and highlight the necessity of the hybrid structure for achieving optimal performance.

Beyond benchmarks, A-GGCO proves its practical relevance in optimizing Paillier Homomorphic Encryption parameters, reducing computational costs while enhancing security in cloud-based cryptographic processing. Its adaptive efficiency also extends to IoT environments, enabling lightweight and energy-efficient optimization for real-time and latency-sensitive applications, such as healthcare analytics. By bridging IoT and cloud domains, A-GGCO demonstrates strong potential as a versatile optimization framework. However, future work is needed to extend its scalability and applicability to broader real-world dynamic systems.

Conflict of Interest

Authors hereby affirm that they have no conflicts of interest.

Ethics Approval

This study was conducted in accordance with the ethical standards.

Funding

The authors received no financial support for the research, authorship, and/or publication of this article.

Data Availability

Data generated in this study is available on request.

Authors' Contributions

- Rekha Gaitond: Conceptualization, methodology, writing—original draft.
- Dr. Gangadhar S. Biradar: Supervision, validation, writing—review and editing.
- Dr. Sujata Terdal: Supervision, writing—review and editing.

Acknowledgment

The authors would like to acknowledge the assistance of ChatGPT in providing language editing and refinement of this manuscript.

Human Participants and/or Animals

Not applicable.

REFERENCES

1. Brownlee J. *Clever Algorithms: Nature-Inspired Programming Recipes.* Jason Brownlee; 2011.
2. Yang X-S. *Introduction to Mathematical Optimization: From Linear Programming to Metaheuristics.* 2008.
3. Gogna A, Tayal A. Metaheuristics: review and application. *J Exp Theor Artif Intell.* 2013;25(4):503-526. doi:10.1080/0952813X.2013.782347.
4. Singh P, Choudhary SK. Introduction: Optimization and Metaheuristics Algorithms. In: Malik H, Iqbal A, Joshi P, Agrawal S, Bakhsh FI, editors. *Metaheuristic and Evolutionary Computation: Algorithms and Applications.* Studies in Computational Intelligence, vol 916. Singapore: Springer; 2021. p. 1-16. doi:10.1007/978-981-15-7571-6_1.
5. Roni MHK, Rana MS, Pota HR, et al. Recent trends in bio-inspired meta-heuristic optimization techniques in control applications for electrical systems: a review. *Int J Dynam Control.* 2022;10:999–1011. doi:10.1007/s40435-021-00892-3.
6. Trojovský P, Dehghani M. A new bio-inspired metaheuristic algorithm for solving optimization problems based on walruses behavior. *Sci Rep.* 2023;13:8775. doi:10.1038/s41598-023-35863-5.
7. Houssein EH, Gad AG, Hussain K, Suganthan PN. Major advances in particle swarm optimization: Theory, analysis, and application. *Swarm Evol Comput.* 2021;100868. doi:10.1016/j.swevo.2021.100868.
8. Kumar A, Nadeem M, Banka H. Nature inspired optimization algorithms: a comprehensive overview. *Evolving Systems.* 2023;14:141–156. doi:10.1007/s12530-022-09432-6.
9. Yang X-S, editor. *Nature-Inspired Algorithms and Applied Optimization.* Springer International Publishing; 2018. ISBN: 978-3-319-67669-2. doi:10.1007/978-3-319-67669-2.
10. Yang XS. A new metaheuristic bat-inspired algorithm. In: Nature-Inspired Cooperative Strategies for Optimization. Studies in Computational Intelligence, vol 284. Springer, Germany; 2010. doi:10.1007/978-3-642-12538-6_6.
11. Chandra SS, Anand HS. Nature-inspired metaheuristic algorithms for optimization problems. Computing, Springer nature, 2022;104:251–269. doi:10.1007/s00607-021-00955-5.
12. Mafarja M, Qasem A, Heidari AA, Aljarah I, Faris H, Mirjalili S. Efficient hybrid nature-inspired binary optimizers for feature selection. Cogn Comput. 2019. doi:10.1007/s12559-019-09668-6.
13. Boussaid I, Lepagnot J, Siarry P. *A survey on optimization metaheuristics*. Inf Sci. 2013;237:82-117. https://doi.org/10.1016/j.ins.2013.02.041
14. Azizi M, Aickelin U, Khorshidi HA, Shishehgarkhaneh MB. Energy valley optimizer: a novel metaheuristic algorithm for global and engineering optimization. Sci Rep. 2023;13:1-16.
15. Zhang W, Pan K, Li S, Wang Y. Special Forces Algorithm: A novel meta-heuristic method for global optimization. Math Comput Simul. 2023;202:1-15.
16. Deng L, Liu S. Snow ablation optimizer: A novel metaheuristic technique for numerical optimization and engineering design. Expert Syst Appl. 2023;213:118904.
17. Abdel-Basset M, Mohamed R, Jameel M, Abouhawwash M. Spider wasp optimizer: a novel meta-heuristic optimization algorithm. Artif Intell Rev. 2023;56:1-34.
18. Guan Z, Ren C, Niu J, Wang P, Shang Y. Great Wall Construction Algorithm: A novel meta-heuristic algorithm for engineer problems. Expert Syst Appl. 2023;229:120015.
19. Słowik A, Cpałka K. Hybrid approaches to nature-inspired population-based intelligent optimization for industrial applications. IEEE Trans Ind Inform. 2022;18(1):546-558. doi:10.1109/TII.2021.3067719.
20. Yang XS. *Nature-Inspired Algorithms and Applied Optimization.* Springer; 2018. doi:10.1007/978-3-319-67669-2.

21. Kumar A, Nadeem M, Banka H. Nature inspired optimization algorithms: a comprehensive overview. *Evolving Systems.* 2023;14(2):141-156. doi:10.1007/s12530-022-09432-6.
22. Mafarja M, Qasem A, Heidari AA, et al. Efficient hybrid nature-inspired binary optimizers for feature selection. *Cogn Comput.* 2019;11(4):556-577. doi:10.1007/s12559-019-09668-6.
23. Xue Y, Aouari A, Mansour RF, Su S. A hybrid algorithm based on PSO and GA for feature selection. *J Cyber Secur.* 2021;3:Article 017018. doi:10.32604/jcs.2021.0170
24. S. Chen, Q.-K. Pan, X. Hu, M.F. Tasgetiren. "NEH-Based heuristics for the distributed blocking flowshop with makespan criterion" in Proc. of 39th Chinese Control Conference (CCC), pp. 1710-1715, Shenyang, China, 2020.
25. M. Ghosh, R. Guha, I. Alam, P. Lohariwal, D. Jalan, R. Sarkar. "Binary genetic swarm optimization: a combination of GA and PSO for feature selection". Journal of Intelligent Systems, vol. 29(1), 2019, doi: 10.1515/jisys-2019-0062.
26. T. Dokeroglu, S. Pehlivan, B. Avenoglu. "Robust parallel hybrid artificial bee colony algorithms for the multi-dimensional numerical optimization". Journal of Supercomputing, vol. 76, pp. 7026-7046, 2020.
27. Chelbi S, Dhahri H, Bouaziz R. Node placement optimization using particle swarm optimization and iterated local search algorithm in wireless sensor networks. *Int J Commun Syst.* 2021;34(8):e4813. doi:10.1002/dac.4813.
28. da Silveira LA, Soncco-Álvarez JL, de Lima TA, Ayala-Rincón M. Parallel Island Model Genetic Algorithms applied in NP-Hard problems. *Proc IEEE Congr Evol Comput (CEC).* 2019:3262-3269. doi:10.1109/CEC.2019.8790257.
29. da Silveira LA, Soncco-Alvarez JL, de Lima TA, Ayala-Rincon M. Parallel Multi-Island Genetic Algorithm for Sorting Unsigned Genomes by Reversals. *Proc IEEE Congr Evol Comput (CEC).* 2018:1-8. doi:10.1109/CEC.2018.8477968.
30. Igbinovia FO, Krupka J. Computational Complexity of Algorithms for Optimization of Multi-Hybrid Renewable Energy Systems. *Proc IEEE Int Conf Power Syst Technol (POWERCON).* 2018:1-8. doi:10.1109/POWERCON.2018.8591905.
31. Abi S, Benhala B, Bouyghf H. A Hybrid DE-ACO Algorithm for the Global Optimization. *Proc IEEE Int Conf Electron Control Optim Comput Sci (ICECOCS).* 2020:1-6. doi:10.1109/ICECOCS50124.2020.9314533.
32. Zhu M, Xu W, Ma W. A novel prestress design method for cable-strut structures with Grey Wolf-Fruit Fly hybrid optimization algorithm. *Structures.* 2024;67:106932. doi:10.1016/j.istruc.2024.106932.
33. Mohapatra S, Mohapatra P. Hybrid grey wolf optimization and salp swarm algorithm for global optimization problems. *AIP Conf Proc.* 2025;3253:030023. doi:10.1063/5.0249625.
34. Attiya I, Abualigah L, Alshathri S, Elsadek D, Abd Elaziz M. Dynamic Jellyfish Search Algorithm based on simulated annealing and disruption operators for global optimization with applications to cloud task scheduling. *Mathematics.* 2022;10(11):1894. doi:10.3390/math10111894.
35. Yildizdan, G., Baş, E. A Novel Binary Artificial Jellyfish Search Algorithm for Solving 0–1 Knapsack Problems. *Neural Process Lett* **55**, 8605–8671 (2023). https://doi.org/10.1007/s11063-023-11171-x
36. Chou, JS., Molla, A. Recent advances in use of bio-inspired jellyfish search algorithm for solving optimization problems. *Sci Rep* **12**, 19157 (2022). https://doi.org/10.1038/s41598-022-23121-z
37. Tripathy BK, Maddikunta PKR, Pham QV, Gadekallu TR, Dev K, Pandya S, ElHalawany BM. Harris Hawk Optimization: A survey on variants and applications. *Comput Intell Neurosci.* 2022;2022:2218594. doi:10.1155/2022/2218594.
38. Zhou Y, Ling Y, Luo Q. Lévy flight trajectory-based whale optimization algorithm for engineering optimization. *Eng Comput.* 2018 Oct 25;36(1):1-23. doi:10.1108/EC-10-2018-0456.
39. Zhou X, Hu W, Zhang Z, Ye J, Zhao C, Bian X. Adaptive mutation sparrow search algorithm-Elman-AdaBoost model for predicting the deformation of subway tunnels. *Underground Space.* 2024;17:320-360. doi:10.1016/j.undsp.2023.09.014.
40. Braik MS, Hammouri AI, Awadallah MA, Al-Betar MA, Khtatneh K. An improved hybrid chameleon swarm algorithm for feature selection in medical diagnosis. *Biomed Signal Process Control.* 2023;85:105073. doi:10.1016/j.bspc.2023.105073.
41. Han T, Wang H, Li T, Liu Q, Huang Y. MHO: A modified hippopotamus optimization algorithm for global optimization and engineering design problems. *Biomimetics (Basel).* 2025;10(2):90. doi:10.3390/biomimetics10020090.
42. Scheiber IBR, Kotrschal K, Weiß BM. Benefits of family reunions: Social support in secondary greylag goose families. Horm Behav. 2009;55(1):133-138. doi: 10.1016/j.yhbeh.2008.09.006.
43. Månsson J, Liljebäck N, Nilsson L, Olsson C, Kruckenberg H, Elmberg J. Migration patterns of Swedish Greylag geese Anser anser—implications for flyway management in a changing world. Eur J Wildl Res. 2022;68:15. doi: 10.1007/s10344-022-01561-2.
44. El-Kenawy EM, Khodadadi N, Mirjalili S, Abdelhamid AA, Eid MM, Ibrahim A. Greylag Goose Optimization: Nature-inspired optimization algorithm. Expert Syst Appl. 2024 Mar 15;238(Pt E):122147. doi: 10.1016/j.eswa.2023.122147.
45. Xiao B, Wang R, Deng Y, Yang Y, Lu D. Simplified Crayfish Optimization Algorithm. In: 2024 IEEE 7th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC); 2024; Chongqing, China. p. 392-396. doi: 10.1109/IAEAC59436.2024.10503709.
46. Jia H, Rao H, Wen C, et al. Crayfish optimization algorithm. Artif Intell Rev. 2023;56(Suppl 2):1919-1979. doi: 10.1007/s10462-023-10567-4.
47. Altaee MM, Alanezi M. Enhancing cloud computing security by Paillier homomorphic encryption. Int J Electr Comput Eng. 2021;11(2):1771-1779. doi: 10.11591/ijece.v11i2.pp1771-1779.
48. Mohammed SJ, Taha DB. Paillier cryptosystem enhancement for homomorphic encryption technique. Multimed Tools Appl. 2024;83:22567-22579. doi: 10.1007/s11042-023-16301-0.
49. Alqarni AA. A secure approach for data integration in cloud using Paillier homomorphic encryption. Albaha Univ J Basic Appl Sci. 2021;5(2):15-21.
50. Sharma P, Raju S. Metaheuristic optimization algorithms: a comprehensive overview and classification of benchmark test functions. Soft Comput. 2024;28:3123–3186. https://doi.org/10.1007/s00500-023-09276-5.
51. Majid Sohrabi, Amir M. Fathollahi-Fard, Vasilii A. Gromov, Genetic Engineering Algorithm (GEA): An Efficient Metaheuristic Algorithm for Solving Combinatorial Optimization Problems, Neural and Evolutionary Computing, arXiv:2309.16413, 2023 https://doi.org/10.48550/arXiv.2309.16413
52. Yao L, Yuan P, Tsai CY, Zhang T, Lu Y, Ding S. ESO: An enhanced snake optimizer for real-world engineering problems. *Expert Syst Appl.* 2023;225:120594. doi:10.1016/j.eswa.2023.120594.

53. Yuan X, Gao Y, Zeng J. A hybrid grey wolf optimizer and chimp optimization algorithm for global optimization. *arXiv [preprint]*. 2025. arXiv:2501.14769. doi:10.48550/arXiv.2501.14769.
54. Faris H, Aljarah I, Al-Betar MA, Mirjalili S. Grey wolf optimizer: a review of recent variants and applications. *Neural Comput Appl.* 2018;30(2):413–35.
55. Nadimi-Shahraki MH, Asghari Varzaneh Z, Zamani H, Mirjalili S. Binary Starling Murmuration Optimizer algorithm to select effective features from medical data. *Appl Sci.* 2023;13(7):4193. doi:10.3390/app13074193
56. Nadimi-Shahraki MH, Fatahi A, Zamani H, Mirjalili S. Binary approaches of quantum-based avian navigation optimizer to select effective features from high-dimensional medical data. *Mathematics.* 2023;11(4):957. doi:10.3390/math11040957