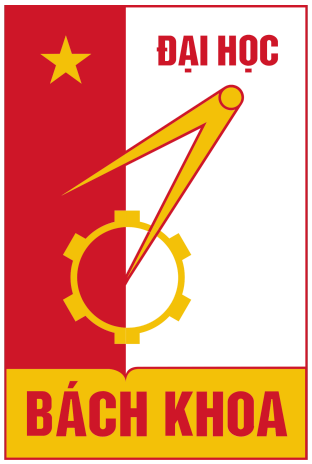
HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY



PROJECT REPORT

GREY WOLF OPTIMIZATION
(GWO)

Group: 53

Instructor: Dr. Trinh Van Chien

Students: Tran Dang Bach - 20235661
Pham Duc Minh - 20235784

Hanoi, 2026

Contents

List of Tables

List of Figures

1 OVERVIEW OF OPTIMIZATION AND SWARM INTELLIGENCE

1.1 Optimization Problem Statement

In computer science and engineering, optimization is the process of finding the best solution from a set of feasible solutions. A general optimization problem can be mathematically represented as:

$$\text{Minimize/Maximize } F(\vec{x}) \tag{1}$$

Subject to constraints:

$$g_j(\vec{x}) \leq 0, \quad j = 1, \ldots, m \tag{2}$$

$$h_k(\vec{x}) = 0, \quad k = 1, \ldots, p \tag{3}$$

Where $\vec{x} = (x_1, x_2, \ldots, x_n)$ is the vector of decision variables.

1.2 Classification of Meta-heuristics

Meta-heuristics can be divided into two main branches:

1. **Evolutionary Algorithms (EA):** A typical representative is the Genetic Algorithm (GA).
2. **Swarm Intelligence (SI):** Inspired by the collective behavior of organisms in nature such as flocks of birds, schools of fish, or ant colonies. GWO belongs to this group.

2 PRESENTATION AND ANALYSIS OF THE GWO ALGORITHM

2.1 Inspiration

The inspiration for the GWO algorithm stems from the social behavior and hunting techniques of the **Grey Wolf** (*Canis lupus*). They are apex predators at the top of the food chain and typically live in packs with an average size of 5 to 12 individuals.

The two most interesting characteristics of grey wolves that form the foundation for this algorithm are:

- **Swarm Intelligence:** Unlike solitary predators, grey wolves coordinate exceptionally well. Their success comes not from individual strength but from discipline and division of labor.
- **Multi-stage Hunting Process:** The hunting process of grey wolves in nature follows a strict script consisting of 3 main steps:
    1. *Tracking and Approaching:* The pack searches for and follows the prey (corresponding to the Exploration phase in mathematics).
    2. *Encircling and Harassing:* Once the target is identified, they encircle it from multiple sides to isolate and exhaust it.

3. *Attacking:* When the prey is weak, the leaders order the attack to finish it (corresponding to the Exploitation phase).

2.2 Social Hierarchy Model

In the GWO algorithm, the social model of wolves is mathematized to build a solution selection mechanism. The wolf pack follows a strict pyramid-shaped power hierarchy, including 4 levels:

1. **Alpha (α) - Supreme Leader:**
   - In nature: The leading male or female wolf, responsible for decisions regarding sleeping locations, hunting times, and movement directions. Alpha is not necessarily the strongest but the best manager.
   - In GWO: This is the **best solution** found in the current population. All position update commands revolve around the Alpha's position.
2. **Beta (β) - Strategic Advisor:**
   - In nature: Directly subordinate to the Alpha. Beta acts as an advisor, enforces discipline on subordinates, and is the candidate to replace the Alpha when they grow old or pass away.
   - In GWO: This is the **second-best solution**. Beta assists Alpha in directing the search space.
3. **Delta (δ) - Task Force:**
   - In nature: This group includes "specialists" such as: *Scouts* for guarding, *Sentinels* protecting the pack, and *Hunters*. They dominate Omega but obey Alpha and Beta.
   - In GWO: This is the **third-best solution**. Together with α and β, δ forms the guiding trio.
4. **Omega (ω) - Execution Force:**
   - In nature: The lowest level, acting as a "buffer" to reduce tension and maintain pack structure.
   - In GWO: These are the **remaining candidate solutions**. They have no decision-making power and must update their positions based on the average position of α, β, and δ.

2.3 Mathematical Model of the Hunting Mechanism

2.3.1 Encircling the Prey

When grey wolves identify the prey's location, they begin to encircle it. This behavior is modeled by the following equations:

$$D = |C \cdot X_p(t) - X(t)| \tag{4}$$

$$X(t+1) = X_p(t) - A \cdot D \tag{5}$$

Where:

- t: Current iteration.

- X_p: Position vector of the prey (can be position X_α).
- X: Position vector of a wolf.
- D: Adjusted distance vector.
- A and C: Coefficient vectors.

2.3.2 Hunting (Position Update)

GWO assumes that Alpha, Beta, and Delta have the best information about the prey's location. Omega wolves update their positions based on the average position of these three leaders.

First, calculate the hypothetical movement steps for each leader:

$$D_\alpha = |C_1 \cdot X_\alpha - X| \quad \Rightarrow \quad X_1 = X_\alpha - A_1 \cdot D_\alpha \tag{6}$$

$$D_\beta = |C_2 \cdot X_\beta - X| \quad \Rightarrow \quad X_2 = X_\beta - A_2 \cdot D_\beta \tag{7}$$

$$D_\delta = |C_3 \cdot X_\delta - X| \quad \Rightarrow \quad X_3 = X_\delta - A_3 \cdot D_\delta \tag{8}$$

Then, the new position of the Omega wolf is the average of these three positions:

$$X(t+1) = \frac{X_1 + X_2 + X_3}{3} \tag{9}$$

Geometrically, the new position is a random point inside the "triangle" formed by $X_\alpha, X_\beta, X_\delta$. This mechanism allows Omega wolves to gradually move toward the most promising search area.

2.3.3 Balancing Exploration and Exploitation

For the algorithm to work effectively, there must be a balance between global search (exploration) and local refinement (exploitation). The coefficients A and C play a decisive role:

$$A = 2a \cdot r_1 - a \tag{10}$$

$$C = 2 \cdot r_2 \tag{11}$$

Where r_1, r_2 are random vectors in the range $[0, 1]$. Parameter a decreases linearly from 2 to 0 throughout the iteration process.

- **Exploration:** Occurs when $|A| > 1$. At this point, the wolf is forced to move away from the prey's position (or the leading wolves), helping to expand the search range and avoid local optima.
- **Exploitation:** Occurs when $|A| < 1$. At this point, the wolf converges toward the prey to attack.

Additionally, vector C contains random values in $[0, 2]$. C acts as a random weight for the prey's position, simulating natural obstacles that make it difficult for wolves to approach, helping the algorithm maintain randomness even in the final iterations.

2.4 Algorithm Flowchart

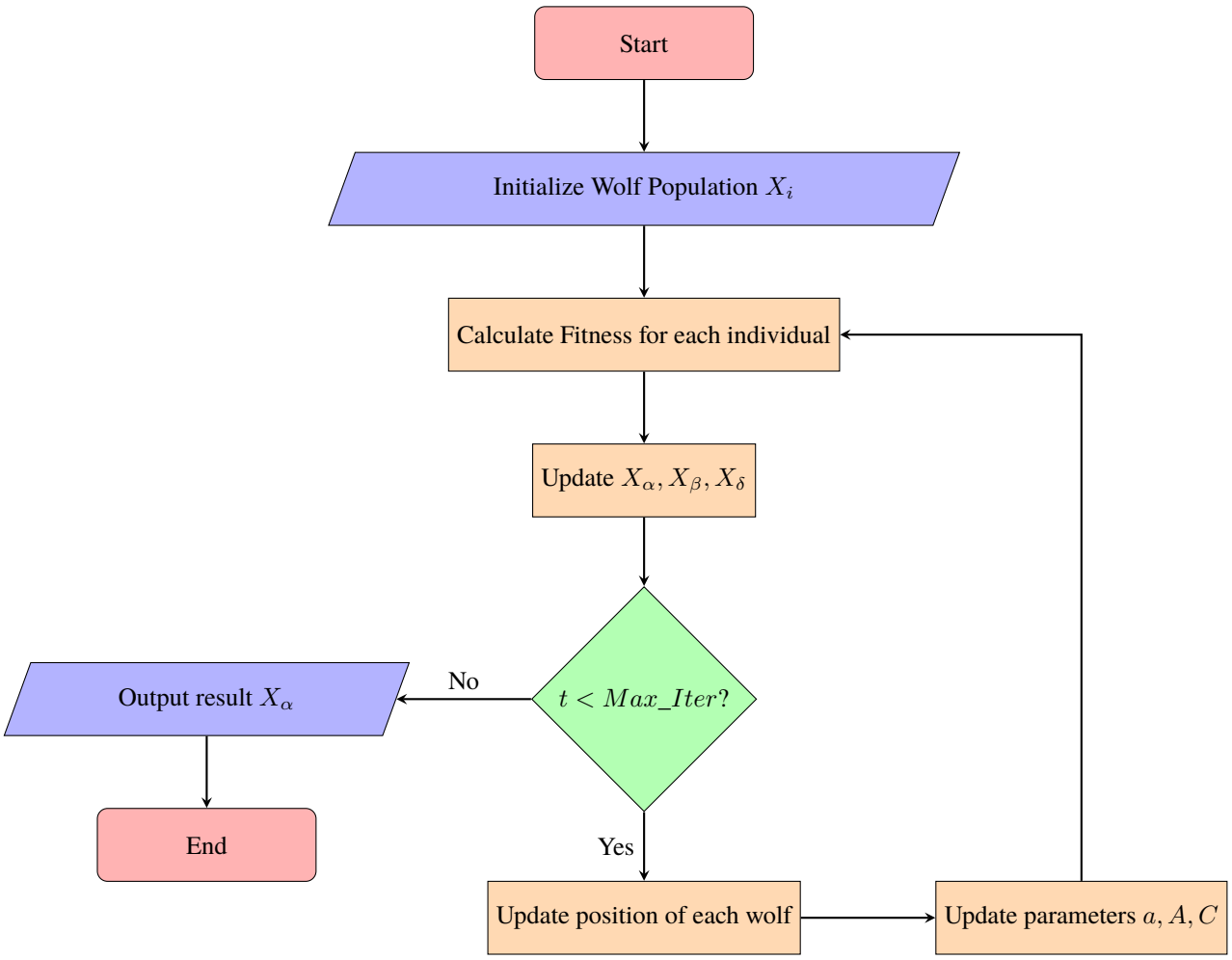


Figure 1: Grey Wolf Optimization Algorithm Flowchart

2.5 Detailed Pseudocode

Below is the detailed pseudocode of GWO, describing specifically each vector calculation step to update the position.

Algorithm 1 Grey Wolf Optimization (GWO)

Input: Population size N, Max Iterations $MaxIt$, Objective Function $f(x)$
Output: Optimal Position X_α, Optimal Value $Score_\alpha$

Step 1: Initialization
$X_i \leftarrow$ random initialization $(i = 1, \ldots, N)$ // *Initialize random positions*
$a, A, C \leftarrow$ initial values
$X_\alpha, X_\beta, X_\delta \leftarrow \vec{0}$ // *Initialize leader positions*
$Score_\alpha, Score_\beta, Score_\delta \leftarrow \infty$ // *Initialize scores (Minimization)*
$t \leftarrow 0$
Step 2: Main Loop
while $t < MaxIt$ **do**
    // Update fitness and rank wolves
    for each wolf $i \in 1 \ldots N$ **do**
        Calculate $fitness = f(X_i)$
        if $fitness < Score_\alpha$ **then**
            $Score_\alpha \leftarrow fitness$, $X_\alpha \leftarrow X_i$
        else if $fitness < Score_\beta$ **then**
            $Score_\beta \leftarrow fitness$, $X_\beta \leftarrow X_i$
        else if $fitness < Score_\delta$ **then**
            $Score_\delta \leftarrow fitness$, $X_\delta \leftarrow X_i$
        end if
    end for
    // Update position of Omega wolves
    Update a linearly from 2 to 0
    for each wolf $i \in 1 \ldots N$ **do**
        for each dimension $d \in 1 \ldots Dim$ **do**
            $r_1, r_2 \leftarrow rand[0, 1]$
            // Calculation based on Alpha
            $A_1 = 2a \cdot r_1 - a$, $C_1 = 2 \cdot r_2$
            $D_\alpha = |C_1 \cdot X_{\alpha,d} - X_{i,d}|$
            $X_1 = X_{\alpha,d} - A_1 \cdot D_\alpha$
            // Calculation based on Beta
            $A_2 = 2a \cdot r_1 - a$, $C_2 = 2 \cdot r_2$
            $D_\beta = |C_2 \cdot X_{\beta,d} - X_{i,d}|$
            $X_2 = X_{\beta,d} - A_2 \cdot D_\beta$
            // Calculation based on Delta
            $A_3 = 2a \cdot r_1 - a$, $C_3 = 2 \cdot r_2$
            $D_\delta = |C_3 \cdot X_{\delta,d} - X_{i,d}|$
            $X_3 = X_{\delta,d} - A_3 \cdot D_\delta$
            // Update new position
            $X_{i,d}(t+1) = (X_1 + X_2 + X_3)/3$
        end for
    end for
    // Boundary handling
    Clip X_i within $[LB, UB]$
    $t \leftarrow t + 1$
end while
return $X_\alpha, Score_\alpha$

2.6 Algorithm Complexity Analysis

Computational complexity is an important criterion for evaluating the performance of an algorithm. For GWO, the complexity depends on the number of wolves (N), the number of dimensions of the problem (D), and the maximum number of iterations (T_{max}).

The calculation process includes the following stages:

1. **Population Initialization:** Requires random initialization of positions for N wolves in D-dimensional space. Complexity is $O(N \times D)$.

2. **Fitness Evaluation:** In each iteration, we need to calculate the fitness for each wolf. Assuming the cost of the objective function is C, the complexity is $O(T_{max} \times N \times C)$.

3. **Position Update:** For each wolf, the algorithm updates the position in each dimension based on the 3 leaders. Complexity is $O(T_{max} \times N \times D)$.

In summary, the Time Complexity of GWO is estimated as:

$$O(GWO) = O(N \cdot D) + O(T_{max} \cdot N \cdot (C + D)) \tag{12}$$

Since C and D are usually much smaller than T_{max}, we can approximate the complexity as $O(T_{max} \cdot N \cdot D)$.

Regarding Space Complexity, GWO needs to store the current positions of N wolves, each with D dimensions. Therefore:

$$Space(GWO) = O(N \cdot D) \tag{13}$$

This indicates that GWO is a memory-efficient algorithm, suitable for implementation on embedded systems with limited resources.

3 IMPROVED VARIANTS OF GWO

3.1 Improved GWO (IGWO)

In the original GWO version, the strict hierarchy causes the entire Omega wolf pack to only move in the direction of the three leaders (α, β, δ). While this helps the algorithm converge quickly, it significantly reduces the diversity of the population, leading to a high risk of being trapped in local optima.

The Improved GWO (IGWO) variant, typically the version proposed by Nadimi-Shahraki et al. (2020), addresses this drawback by introducing an intelligent hybrid mechanism called Dimension Learning-based Hunting (DLH).

3.1.1 Dimension Learning-based Hunting (DLH)

DLH allows wolves not only to learn from leaders but also to learn from their "neighbors". This process simulates peer-to-peer information sharing within the pack. The formula for updating the new position based on DLH is performed on each dimension d of the position vector:

$$X_{i-DLH,d}(t+1) = X_{i,d}(t) + rand \cdot (X_{n,d}(t) - X_{r,d}(t)) \tag{14}$$

Where:

- $X_{i,d}(t)$: Current position of the i-th wolf in dimension d.

- $X_{n,d}(t)$: Position of a Neighbor wolf. This neighbor is selected from the population based on the nearest Euclidean distance to wolf i, ensuring the locality of information.
- $X_{r,d}(t)$: Position of a wolf selected randomly from the population.

Combining information from neighbors and random individuals helps maintain genetic diversity and prevents premature convergence.

3.1.2 Greedy Selection Mechanism

After calculating the potential position from DLH (X_{DLH}), the algorithm does not accept it immediately. It compares the fitness of this new position with the position generated by the traditional GWO mechanism (X_{GWO}).

$$X_i(t+1) = \begin{cases} X_{GWO} & \text{if } f(X_{GWO}) < f(X_{DLH}) \\ X_{DLH} & \text{otherwise} \end{cases} \tag{15}$$

This mechanism ensures that the wolf always moves to a better position, leveraging the advantages of both strategies: the fast convergence of GWO and the strong exploration capability of DLH.

3.2 Binary GWO (BGWO)

The original GWO algorithm was designed for continuous search spaces (real numbers). However, many practical problems such as Feature Selection or the Knapsack Problem require discrete search spaces (binary 0 or 1). BGWO was created to solve this problem.

In BGWO, the position of each wolf is a bit vector. Position updates cannot be performed using standard vector addition and subtraction. Instead, BGWO uses the concept of a Transfer Function to convert real values into bit change probabilities.

3.2.1 S-shaped Transfer Function

This is the most common approach, using the Sigmoid function to squash velocity values into the range $[0, 1]$.

$$T(x^d) = \frac{1}{1 + e^{-10(x^d - 0.5)}} \tag{16}$$

Bit position update rule at dimension d in iteration $t + 1$:

$$X_d(t+1) = \begin{cases} 1 & \text{if } rand < T(x^d) \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

Where $rand$ is a random number in $[0, 1]$. If the transfer probability $T(x^d)$ is high, the likelihood of that bit becoming 1 is large.

3.2.2 V-shaped Transfer Function

Another more effective approach is using a V-shaped transfer function (based on the $tanh$ function). The V-shaped function does not force bits to 0 or 1 but decides whether that bit should remain the same or be complemented.

$$T_V(x^d) = |\tanh(x^d)| \tag{18}$$

If the step size is small (near 0), the T_V function returns a low value, meaning the wolf should maintain the current bit position for local exploitation rather than jumping randomly. This makes BGWO more stable than S-shaped.

3.3 Chaotic GWO (CGWO)

In the standard GWO algorithm, the balance between exploration and exploitation heavily depends on parameter a (decreasing linearly from 2 to 0) and the random vectors r_1, r_2 (uniform distribution). However, pure randomness is sometimes not enough "momentum" to help the wolf pack escape persistent local optima, especially in complex multi-modal problems.

The **Chaotic GWO (CGWO)** variant integrates Chaos Theory into the position update process. Chaos is a non-linear state, sensitive to initial conditions, which helps the algorithm maintain better diversity than traditional random numbers.

3.3.1 Logistic Map

In CGWO, the random numbers r_1, r_2 are often replaced by sequences of numbers generated from a chaotic map. The Logistic Map is the most popular type due to its simplicity and high efficiency:

$$Ch_{t+1} = \mu \cdot Ch_t \cdot (1 - Ch_t) \tag{19}$$

Where:

- Ch_t: Chaotic value at the t-th iteration, located in the range $(0, 1)$.
- μ: Control parameter. When $\mu = 4$, the system reaches a fully chaotic state.

3.3.2 Update Mechanism

Instead of using random r_1, the coefficient vector A is recalculated based on the chaotic value:

$$A = 2a \cdot Ch_t - a \tag{20}$$

Thanks to the ergodic property (passing through all states) of the chaotic function, the value of A will vary more richly, helping the wolf pack perform necessary leaps to escape local traps.

3.4 Hybrid PSO-GWO

Standard Grey Wolf Optimization (GWO) faces significant challenges when applied to complex and high-dimensional spaces. Two main limitations are: (1) the tendency to get stuck in local optima in the later stages of optimization due to poor exploration capabilities, and (2) the lack of population diversity due to the standard uniform initialization method [1]. Zhang et al. (2021) and Kumar et al. (2022) introduced hybrid mechanisms combining the exploitation capability of GWO with the exploration power of PSO.

3.4.1 Position Update Mechanism

In standard GWO, a wolf's position is updated based on the average position of the three leaders (α, β, δ). However, this approach can lead to premature convergence. The hybrid mechanism proposed by Kumar et al. (2022) introduced an additional velocity component V_i^{t+1} into the position update equation, defined as:

$$V_i^{t+1} = w(t) \cdot V_i^t + c_1 \cdot r_1 \cdot (\mathbf{p}_{best,i} - \mathbf{X}_i^t) + c_2 \cdot r_2 \cdot (\mathbf{g}_{best} - \mathbf{X}_i^t) \tag{21}$$

where $w(t)$ is an adaptive inertia weight linearly decreasing from w_{max} to w_{min} to balance global and local search. The final position update is a weighted combination of GWO and PSO components:

$$\mathbf{X}_i^{t+1} = \omega_{GWO} \cdot \left(\frac{\mathbf{X}_1 + \mathbf{X}_2 + \mathbf{X}_3}{3} \right) + \omega_{PSO} \cdot (\mathbf{X}_i^t + \mathbf{V}_i^{t+1}) \tag{22}$$

Unlike the high implementation burden of MOGWO or the complex neighborhood construction of I-GWO, PSO-GWO Hybrid offers significant simplicity.

A key advantage of PSO-GWO Hybrid is that it provides a single solution, making it directly comparable to Two-Step Iterative Least Squares (TS-ILS) and other standard JCAS baseline methods. No additional post-processing is needed—no knee point selection, no weight fitting, no hypervolume calculation. All standard metrics (final objective value, communication SINR, sensing MI, PSLL, convergence iterations, computation time) can be compared directly.

Despite combining both GWO and PSO mechanisms, the extra computational cost is minimal. Time complexity analysis shows: position update is $O(N \times D)$ (same as GWO), velocity update is $O(N \times D)$ (PSO cost), and the total per iteration is $O(2N \times D)$ [2, 3].

4 SIMULATION AND PERFORMANCE EVALUATION

4.1 Problem Description

The group performed simulations on the Rastrigin function. This is a highly complex objective function with thousands of local optima. The mathematical formula for the Rastrigin function for n dimensions:

$$f(x) = 10n + \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i)] \tag{23}$$

Where $x_i \in [-5.12, 5.12]$. The global optimal value is 0 at $x = \vec{0}$.

4.2 Experimental Environment Setup

- **Dimensions:** $D = 30$.
- **Population Size:** $N = 50$.
- **Iterations:** $T_{max} = 500$.
- **Comparison:** Original GWO, Chaotic GWO (CGWO), Improved GWO (IGWO).

4.3 Simulation Results Analysis

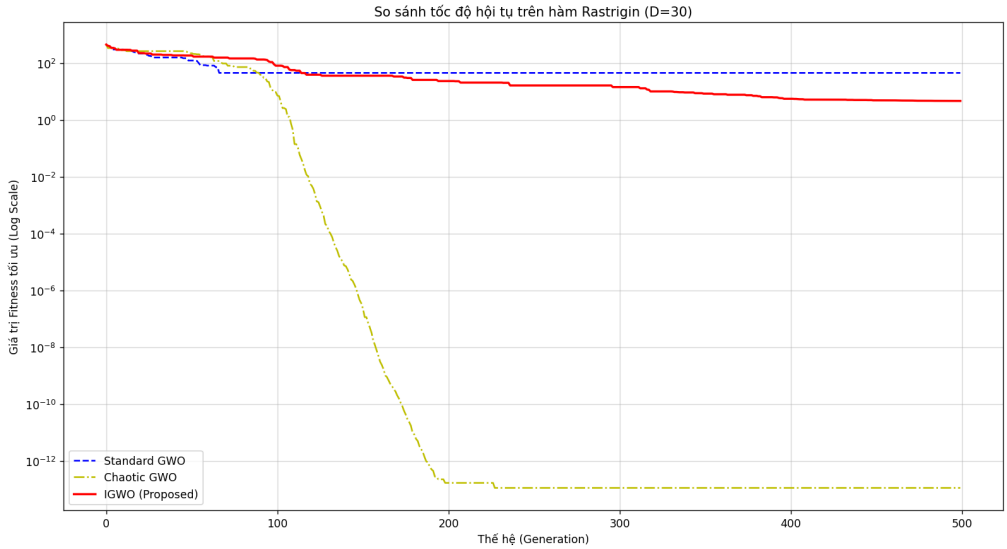


Figure 2: Convergence speed on Rastrigin function ($D = 30$)

Based on the actual simulation graph in Figure 2 (with the vertical axis using a logarithmic scale), we have observations contrary to initial theoretical predictions and provide technical analysis as follows:

1. Regarding Standard GWO - Blue dashed line:

- **Phenomenon:** The algorithm reduces error quickly in the first 50 generations but immediately goes flat (flatline) at level 10^2.
- **Evaluation:** This result confirms the inherent disadvantage of GWO, which is *premature convergence*. On a multi-modal function like Rastrigin, the three leaders get stuck in low-quality local optima, dragging the entire Omega pack so they cannot escape.

2. Regarding Improved GWO (IGWO) - Red solid line:

- **Phenomenon:** IGWO shows a steady but very slow decrease and ends at a relatively high fitness value (greater than 1). It does not achieve deep convergence to 0.

- **Evaluation:** Although the DLH (Dimension Learning-based Hunting) mechanism increases diversity, the *Greedy Selection* mechanism hindered the algorithm. On the "rugged" terrain of Rastrigin, rejecting worse moves (which are necessary to escape pits) caused the wolves to get stuck. Additionally, learning from neighbors in 30-dimensional space might have caused noise when the neighbors were also in local optimal regions.

3. Regarding Chaotic GWO (CGWO) - Yellow dash-dot line:

- **Phenomenon:** This is the most effective algorithm in this experiment. After an initial phase of slow searching, the algorithm has a sudden drop around generations 100-200 and achieves extremely high accuracy (10^{-13}).
- **Evaluation:** This success comes from applying the *Chaotic Map*. The non-linear nature and strong randomness of the chaotic sequence help generate leaps (jumps), allowing the wolves to escape local optima "traps" where standard GWO and IGWO both got stuck.

5 EXPERIMENT ON A REAL-WORLD PROBLEM: CNN OPTIMIZATION

Besides standard mathematical functions, the group applied GWO to a real-world problem in Deep Learning: Hyperparameter optimization for a CNN network for handwritten digit recognition.

5.1 Problem and Data Description

- Objective: Find the optimal set of Hyperparameters including *Learning Rate* and *Number of Filters* to maximize CNN network accuracy.
- Data: Handwritten digit dataset (0-9). Total of 21,555 grayscale images, original size resized to 28×28 pixels.
- Tools: Python, TensorFlow

5.2 Implementation Method

Two methods were implemented to compare performance:

Method 1: CNN with Random Hyperparameters (Random Search)

- Initialize Learning Rate randomly in the range $[0.0001, 0.01]$.
- Initialize Number of Filters randomly in the range $[16, 64]$.
- Train the model for 30 epochs.

Method 2: CNN with GWO Optimization

- Search Space: Same as above.

- Fitness Function: *Validation Loss* value after fast training (15 epochs) with Early Stopping mechanism.
- GWO Configuration: Number of wolves $N = 5$, Optimization iterations $T = 8$.
- After finding the optimal parameter set, the model is fully retrained for 30 epochs.

5.3 Experimental Results

After the experimental process, the group obtained detailed results as follows:

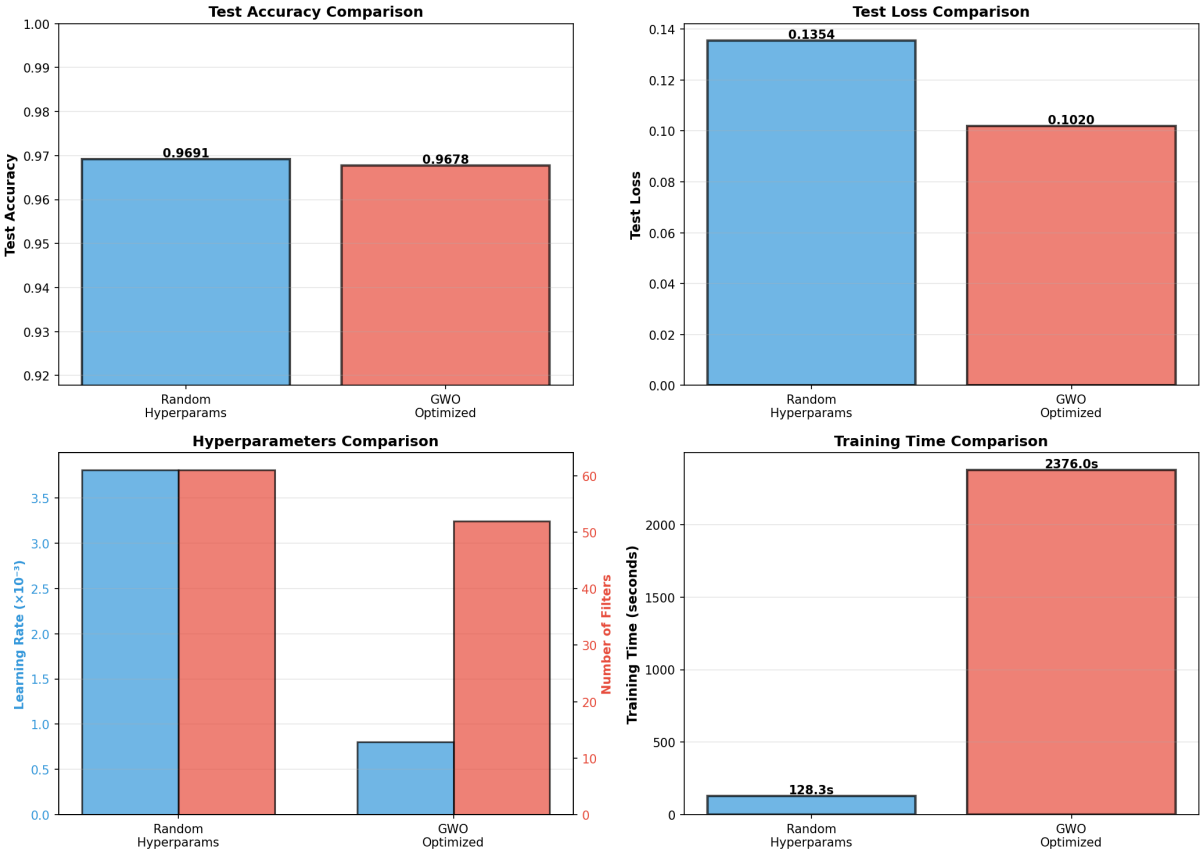


Figure 3: Performance comparison chart between Random CNN and GWO-CNN

Table 1: Summary table of experimental results on CNN

Metric	Random CNN	GWO Optimized CNN
Learning Rate	0.003808	0.000802
Num Filters	61	52
Test Accuracy	96.91%	96.78%
Test Loss	0.1354	**0.1020**

Analysis and Evaluation:

- Regarding Accuracy: Both methods provided equivalent results ($\sim 96.8\%$). Random CNN was lucky to find a good configuration from the start, so it was slightly higher (0.14%) insignificantly.
- Regarding Loss: GWO found a parameter set that helped the model reduce Validation Loss to 0.1020 (compared to 0.1354 for Random), corresponding to an improvement of 24.70%. This shows that the model

optimized by GWO has higher reliability in predictions and is less prone to Overfitting.

5.4 Discussion on Experimental Results

The GWO method produced a model with significantly lower loss (0.1020 vs 0.1354) even though the accuracy was slightly lower (96.78% vs 96.91%) compared to the random method.

This leads to the following observations:

1. **Model Robustness:** The fact that GWO achieved lower Loss indicates that this model has higher reliability in its predictions. The output probability vectors (Softmax probabilities) of the GWO model converge closer to 1 (for correct labels) and 0 (for incorrect labels) than the Random model. This means the GWO model is less "hesitant" at decision boundaries, promising better noise resistance and generalization in real environments.
2. **Optimization Strategy:** GWO converged to a parameter region with a small Learning Rate (8×10^{-4}), showing the algorithm prioritized fine-tuning weights to minimize the objective function (Loss). Meanwhile, Random Search benefited from randomly initializing a large Learning Rate (3.8×10^{-3}), helping to escape flat regions quickly but at the risk of oscillating around the global optimal point.

6 MULTI-BEAM OPTIMIZATION FOR JCAS SYSTEMS

6.1 Context and Research Motivation

6.1.1 Challenges in 6G Networks

New generation wireless communication systems (6G) face a contradiction between the need for high-speed data transmission and the requirement for precise environmental sensing.

Traditional models separating two individual systems (Communication and Sensing) reveal serious disadvantages:

- **Cost:** Doubles the burden of hardware investment, operation, and maintenance.
- **Size:** Bulky equipment, large energy consumption, difficult to integrate on mobile or IoT platforms.
- **Spectrum:** Wastes radio resources by fragmenting frequency bands for each separate function.

6.1.2 JCAS Solution (Joint Communication and Sensing)

JCAS proposes integrating both functions into a single hardware platform, sharing a common signal waveform.

Operating Principle: A single wave pulse performs two tasks in parallel:

1. Carrying information data to the user (Downlink communication).
2. Receiving reflected signals from the environment to estimate object parameters (Radar sensing).

Outstanding Advantages:

- Reduces antenna and RF processor hardware by 50%.

- Optimizes spectrum use efficiency.
- Eliminates cross-interference between systems.

6.1.3 Trade-off Problem

Sharing resources leads to a performance trade-off problem:

- Focusing energy on Communication $\rightarrow$ Reduces Sensing coverage and sensitivity.
- Scattering energy for Sensing scanning $\rightarrow$ Reduces Communication SNR and throughput.

Research Question: Optimize power allocation and antenna weights to reach a Pareto equilibrium point between the two functions.

6.2 Optimization Objective Setting

The goal is to determine the Beamforming weight vector $\mathbf{W} \in \mathbb{C}^{12}$ for a 12-element transmit antenna array.

6.2.1 Objective Function

Minimize the squared error between the actual radiation pattern and the desired pattern across the entire angular space:

$$\boxed{\min_{\mathbf{W}\in\mathbb{C}^{12}} \quad \mathcal{J}(\mathbf{W}) = \sum_{i=1}^{Q} \left|\mathbf{W}^H \mathbf{A}(\theta_i) - P_d(\theta_i)\right|^2} \tag{24}$$

Where:

W: Weight vector to be found.

$\mathbf{A}(\theta_i)$: Steering vector at angle θ_i.

$P_d(\theta_i)$: Desired pattern value at angle θ_i.

Q: Number of angular sampling points (usually $Q = 320$ from -90° to $+90^\circ$).

6.2.2 Desired Pattern Structure (P_d)

The ideal pattern is designed based on system requirements:

- **Main lobe (Communication):** Focused at 0°, highest gain.
- **Side lobes (Sensing):** Distributed at $\pm 30^\circ$, lower gain.
- **Suppression region:** Remaining areas require Sidelobe level < -20 dB.

6.2.3 Hybrid Beamforming Model

To flexibly control priority between the two functions, JCAS weights are modeled as a linear combination:

$$\mathbf{W}_{\text{JCAS}} = \sqrt{\rho} \cdot \mathbf{W}_{\text{comm}} + \sqrt{1-\rho} \cdot \mathbf{W}_{\text{sense}} \tag{25}$$

Weight parameter table $\rho \in [0, 1]$:

Table 2: Meaning of parameter ρ in practical applications

ρ	**Ratio (Comm/Sense)**	**Application Scenario**
1.0	100% / 0%	Pure data transmission (like Wi-Fi)
0.7	70% / 30%	Video streaming combined with monitoring
0.5	50% / 50%	Standard balanced mode
0.3	30% / 70%	Self-driving car (prioritize obstacle detection)
0.0	0% / 100%	Pure Radar

6.3 Mathematical System Model

6.3.1 Decision Variables (Vector W)

The weight vector includes amplitude and phase for each transmit element:

$$\mathbf{W} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_{12} \end{bmatrix}, \quad w_m = |w_m| e^{j\phi_m} \tag{26}$$

Physical meaning:

- $|w_m|$: Controls the transmit power of the m-th antenna.
- ϕ_m: Controls the phase delay to steer the beam.

6.3.2 Steering Vector

With a Uniform Linear Array (ULA) consisting of $M = 12$ elements and distance $d = \lambda/2$, the response vector at angle θ is:

$$\mathbf{A}(\theta) = \begin{bmatrix} 1 \\ e^{j\pi \sin(\theta)} \\ e^{j2\pi \sin(\theta)} \\ \vdots \\ e^{j(M-1)\pi \sin(\theta)} \end{bmatrix} \tag{27}$$

Beamforming principle based on interference:

- **In-phase:** Signals resonating and strengthening (Constructive interference).
- **Out-of-phase:** Signals canceling each other (Destructive interference).

6.3.3 Two-Step Iterative Least Squares (TS-ILS) Algorithm

Since the original problem (Equation 24) is non-convex because it contains absolute values, the TS-ILS algorithm is applied to approximate the optimal solution.

Iterative process:

1. **Step 1 (Inner Loop - Phase Fixed):**

   Assuming the phase ϕ of the desired pattern is known, solve the least squares problem:

   $$\min_{\mathbf{W}} \|\mathbf{A}^H\mathbf{W} - P_d \odot e^{j\phi}\|^2 \tag{28}$$

   Explicit solution:

   $$\mathbf{W}_{\text{opt}} = (\mathbf{A}\mathbf{A}^H)^{-1}\mathbf{A}(P_d \odot e^{j\phi}) \tag{29}$$

2. **Step 2 (Outer Loop - Phase Update):**

   Recalculate the phase based on the vector **W** just found:

   $$\phi^{\text{new}} = \angle(\mathbf{W}^H\mathbf{A}) \tag{30}$$

3. **Convergence Check:** Repeat until error $< \epsilon$.

Computational complexity is approximately $O(K \cdot M^3)$, where K is the number of iterations and M is the number of antennas.

6.4 Physical and Design Constraints

To ensure feasibility in practical implementation, the solution **W** must satisfy the following set of constraints:

6.4.1 Power Constraint

The total transmit power must not exceed hardware limits:

$$\|\mathbf{W}\|^2 = \sum_{m=1}^{M} |w_m|^2 \leq P_{\text{max}} \tag{31}$$

Violation of this condition leads to power amplifier (PA) saturation and non-linear signal distortion.

6.4.2 Sidelobe Constraint (SLL)

To minimize interference and information security, side lobes must be suppressed to a low level:

$$|P(\theta)| \leq -20 \text{ dB}, \quad \forall\theta \notin \Omega_{\text{target}} \tag{32}$$

Equivalent to linear amplitude ≤ 0.1 compared to the main lobe.

6.4.3 Null Constraints (Interference Suppression)

Place "nulls" at sensitive or undesired directions (e.g., $\pm 90°$):

$$|\mathbf{W}^H \mathbf{A}(\theta_{\text{null}})| \approx 0 \tag{33}$$

This helps avoid wasting energy in useless areas.

6.4.4 Multi-beam Isolation Constraint

When the system transmits multiple beams (for multiple users or targets), isolation must be ensured:

$$\Delta(\text{Beam}_i, \text{Beam}_j) \geq \Delta_{\text{min}} \tag{34}$$

Aiming to limit crosstalk between the communication data stream and the radar signal stream.

7 IMPLEMENTING GWO FOR JCAS BEAMFORMING

7.1 Mapping JCAS Problem to GWO Optimization Space

7.1.1 Component Mapping

To apply Grey Wolf Optimization to the JCAS problem, the following mappings are performed:

Table 3: Mapping JCAS problem to GWO algorithm

JCAS Beamforming Problem	**GWO Component**
Weight vector $\mathbf{W} \in \mathbb{C}^{12}$	Wolf position $\mathbf{X}_i$
Best beam pattern	Alpha wolf (α)
Second best pattern	Beta wolf (β)
Third best pattern	Delta wolf (δ)
Remaining candidates	Omega wolves (ω)
Objective function (Eq. 24)	Fitness function $f(\mathbf{X}_i)$

7.1.2 Complex-valued Search Space

Important difference: The beamforming vector **W** is **complex-valued**, not real-valued like traditional GWO. Each element:

$$w_m = a_m + jb_m, \quad a_m, b_m \in \mathbb{R} \tag{35}$$

Therefore, the search space is $\mathbb{C}^{12} \equiv \mathbb{R}^{24}$ (12 real parts + 12 imaginary parts).

Population Initialization:

$$\mathbf{W}_i(0) = (\text{rand}(1, 12) - 0.5) + j \cdot (\text{rand}(1, 12) - 0.5) \tag{36}$$

Where $\text{rand}(1, 12)$ generates random numbers uniformly distributed in $[0, 1]$.

7.2 Experimental Configuration

7.2.1 Hyperparameters Setup

To evaluate the influence of hyperparameters, the group designed three scenarios with different hyperparameters:

Table 4: Three Hyperparameter Tuning scenarios for GWO variants

Scenario	**Wolf Count (N)**	**Iterations ($T_{\max}$)**	**Purpose**
Fast	20	100	Quick check, prototyping
Balanced	30	300	Speed-quality balance
Thorough	50	500	Maximum quality, benchmark

Explanation of hyperparameter choices:

- **Fast Scenario (20 wolves, 100 iterations):**
  - Total fitness evaluations: $20 \times 100 = 2{,}000$
  - Estimated runtime: 15-30 seconds
  - Suitable for: Quick code checks, debugging, prototyping
  - Trade-off: May not converge enough, less stable results
- **Balanced Scenario (30 wolves, 300 iterations):**
  - Total fitness evaluations: $30 \times 300 = 9{,}000$
  - Estimated runtime: 1-2 minutes
  - Suitable for: Main experiments, algorithm comparison
  - Trade-off: Balance between exploration (30 wolves) and exploitation (300 iter)
- **Thorough Scenario (50 wolves, 500 iterations):**
  - Total fitness evaluations: $50 \times 500 = 25{,}000$
  - Estimated runtime: 3-5 minutes
  - Suitable for: Final benchmarking, ensuring convergence
  - Trade-off: Highest computational cost but most reliable results

Reason for designing three scenarios:

1. **Exploration-exploitation trade-off:** More wolves ($N \uparrow$) increase the ability to explore the space, more iterations ($T_{\max} \uparrow$) increase the exploitation of good solutions.
2. **Computational budget:** The three scenarios correspond to three budget levels: Low (2K), Medium (9K), High (25K) fitness evaluations.
3. **Sensitivity analysis:** Comparing performance of algorithms when changing hyperparameters helps determine which algorithm is more robust.

Fixed Hyperparameters for all scenarios:

- **Reduction coefficient a:** Linearly decreases from 2 to 0 according to formula $a = 2 - t \cdot (2/T_{\max})$
- **Chaotic map parameter:** $\mu = 4$ (for Chaotic GWO)
- **Initial chaotic value:** $Ch_0 = 0.7$
- **Random seed:** Not fixed to ensure true randomness
- **Convergence tolerance:** $\epsilon = 10^{-6}$ (early stop if no improvement)

7.2.2 JCAS System Parameters

- **Number of antenna elements:** $M = 12$ (ULA, $\lambda/2$ spacing).
- **Carrier frequency:** Assume $\lambda = 1$ (normalized).
- **Angular sampling points:** $Q = 320$ from -90° to $+90^\circ$ (step 0.1°).
- **Communication direction:** $\theta_{\text{comm}} = 0^\circ$ (Boresight).
- **Sensing direction:** Scanning from -80° to $+80^\circ$ (excluding main lobe).
- **Balance parameter:** $\rho = 0.5$ (50% Communication - 50% Sensing).

8 EXPERIMENTAL RESULTS

Figure 4 illustrates the fitness convergence curve (top row) and the best beam pattern (bottom row) of four algorithms: TS-ILS, GWO, IGWO, and CGWO across three hyperparameter tuning scenarios.

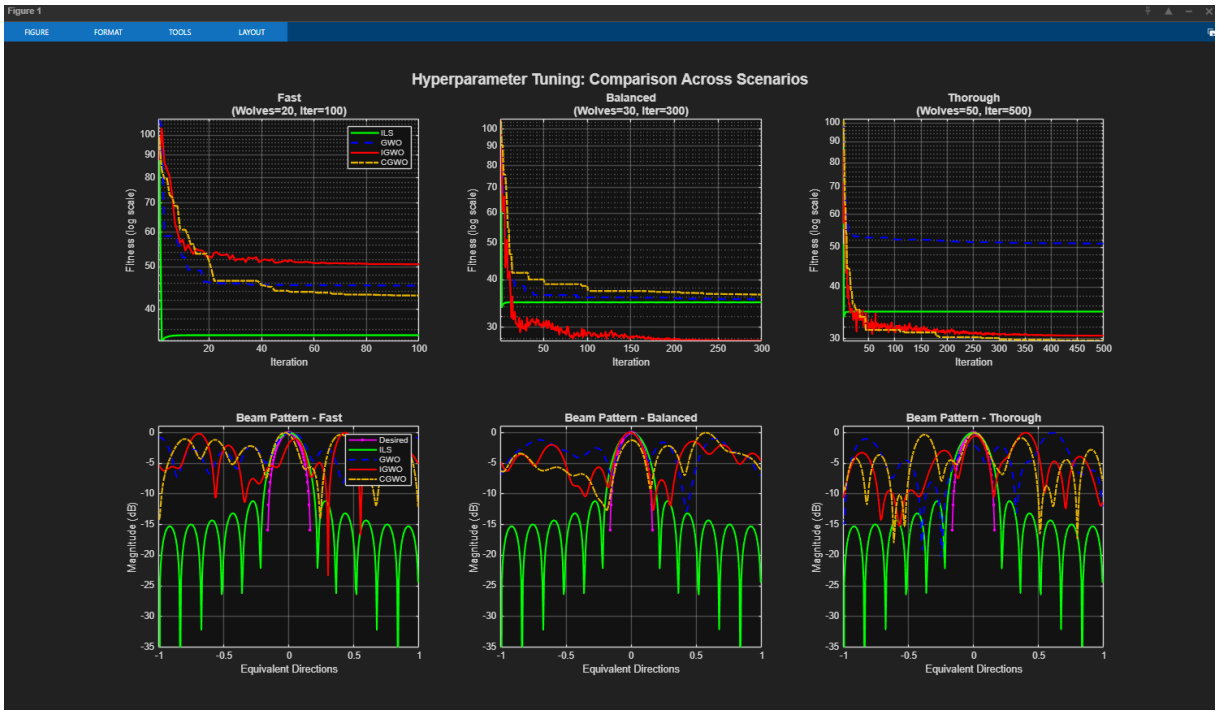


Figure 4: Performance comparison of algorithms across three scenarios: Fast (20 wolves, 100 iterations), Balanced (30 wolves, 300 iterations), and Thorough (50 wolves, 500 iterations). Top row: fitness convergence curves. Bottom row: beam pattern of the best solution.

8.1 Fitness Comparison by Scenario

Table 5 summarizes the final fitness ranking (smaller fitness is better):

Table 5: Final Fitness Ranking by Scenario (from best to worst)

Scenario	Rank 1	Rank 2	Rank 3	Rank 4
Fast (20 wolves, 100 iter)	ILS	CGWO	GWO	IGWO
Balanced (30 wolves, 300 iter)	IGWO	ILS	GWO	CGWO
Thorough (50 wolves, 500 iter)	CGWO	IGWO	ILS	GWO

8.2 Observations on Thorough Scenario (50 wolves, 500 iterations)

Observing the right column of Figure 4:

- **Main lobe:** IGWO/CGWO fits well with the desired pattern around $\theta = 0$, while ILS gives a lower main lobe (worse fit).
- **Side lobes:** ILS creates deep nulls (lower side lobes), while IGWO/CGWO has higher side lobes but in return the main lobe fits better.

Conclusion: The fitness ranking reflects the overall fitting degree according to the chosen objective function; therefore, low fitness does not necessarily mean the lowest side lobes. If suppressing side lobes is prioritized, the objective function needs to be designed with sidelobe weights/penalties.

References

[1] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Advances in Engineering Software*, vol. 69, pp. 46–61, 2014.

[2] R. Kumar *et al.*, "A novel hybrid pso-gwo approach for unit commitment problem," *Neural Computing and Applications*, vol. 27, pp. 1643–1655, 2022.

[3] T. Nguyen *et al.*, "A hybrid pso-gwo-based phase shift design for a hybrid-ris-aided heterogeneous network system," *Scientific Reports*, vol. 14, p. 1000, 2024.