BÁO CÁO PROJECT:
THUẬT TOÁN WHALE OPTIMIZATION ALGORITHM (WOA)
VÀ CÁC BIẾN THỂ (1.1 - 1.9)

Tổng hợp chi tiết từ tài liệu

Ngày 11 tháng 12 năm 2025

Mục lục

1 Thuật toán Whale Optimization Algorithm (WOA)

1.1 Mục đích

Thuật toán Whale Optimization Algorithm (WOA) là một thuật toán tối ưu metaheuristic dùng để tìm nghiệm tối ưu của các bài toán phi tuyến, nhiều cực trị cục bộ và không có đạo hàm.

WOA mô phỏng hành vi săn mồi theo vòng xoắn của cá voi lưng gù (bubble-net feeding).

1.2 Mô tả thuật toán và Các cơ chế cập nhật

Whale Optimization Algorithm (WOA) thực hiện tối ưu thông qua ba nhiệm vụ chính mô phỏng hành vi săn mồi của cá voi lưng gù:

1. **Encircling prey** (Bao vây con mồi).
2. **Bubble-net spiral attack** (Tấn công mạng lưới bong bóng xoắn ốc).
3. **Search for prey** (Tìm kiếm con mồi).

1.2.1 1. Encircling prey (Bao vây con mồi)

WOA giả định rằng nghiệm tốt nhất hiện tại $\vec{X}^*(t)$ chính là vị trí con mồi. Các cá voi cập nhật vị trí bằng cách tiến dần về nghiệm này:

$$\vec{D} = \left| \vec{C} \cdot \vec{X}^*(t) - \vec{X}_i(t) \right| \tag{1}$$

$$\vec{X}_i(t+1) = \vec{X}^*(t) - \vec{A} \cdot \vec{D} \tag{2}$$

Trong đó $\cdot$ là phép nhân từng phần tử.

1.2.2 2. Bubble-net spiral attack (Tấn công xoắn ốc)

Để mô phỏng chuyển động xoắn ốc khi săn mồi, WOA sử dụng phương trình sau:

$$\vec{D}' = \left| \vec{X}^*(t) - \vec{X}_i(t) \right| \tag{3}$$

$$\vec{X}_i(t+1) = \vec{D}' \cdot e^{bl} \cos(2\pi l) + \vec{X}^*(t) \tag{4}$$

trong đó l là số ngẫu nhiên trong khoảng $[-1, 1]$ và b là tham số điều chỉnh độ cong của đường xoắn ốc.

1.2.3 3. Search for prey (Tìm kiếm con mồi)

Khi $|A| > 1$, cá voi không tiến về nghiệm tốt nhất nữa mà tiến về một cá thể được chọn ngẫu nhiên $\vec{X}_{\text{rand}}$, nhằm tăng khả năng khám phá (Exploration):

$$\vec{D} = \left| \vec{C} \cdot \vec{X}_{\text{rand}}(t) - \vec{X}_i(t) \right| \tag{5}$$

$$\vec{X}_i(t+1) = \vec{X}_{\text{rand}}(t) - \vec{A} \cdot \vec{D} \tag{6}$$

1.2.4 Quy tắc lựa chọn cơ chế cập nhật

WOA chọn ngẫu nhiên giữa hai chiến lược bao vây (1) và tấn công xoắn ốc (2) với xác suất $p \in [0, 1]$:

$$\vec{X}_i(t+1) = \begin{cases} \vec{X}^*(t) - \vec{A} \cdot \vec{D} & \text{nếu } p < 0.5 \\ \vec{D}' \cdot e^{bl} \cos(2\pi l) + \vec{X}^*(t) & \text{nếu } p \geq 0.5 \end{cases} \tag{7}$$

Khi $|A| \geq 1$, cơ chế tìm kiếm ngẫu nhiên (3) sẽ được ưu tiên kích hoạt để tăng mức độ khám phá không gian nghiệm.

1.3 Mã giả

```
Khởi tạo quần thể Xi
Đánh giá fitness
X* = cá thể tốt nhất
while (t < MaxIter)
    for mỗi Xi (i=1...N)
        Cập nhật a, A, C, l, p
        if p < 0.5
            if |A| < 1
                bao vây con mồi
            else
                tìm kiếm ngẫu nhiên
        else
            tấn công xoắn ốc
    end for
    Cập nhật X*
    t = t + 1
end while
```

1.4 Độ phức tạp tính toán

Độ phức tạp của WOA:

$$O(T \times N \times D)$$

với T = số vòng lặp, N = số cá voi, D = số chiều nghiệm.

1.5 Hội tụ

WOA hội tụ nhờ tham số a giảm dần từ 2 về 0, giúp chuyển dần từ khám phá sang khai thác nghiệm tốt nhất.

1.6 Nhược điểm

- Có thể rơi vào cực trị cục bộ nếu tham số chưa tối ưu.
- Hội tụ chậm với không gian lớn.
- Cần chạy nhiều lần để đạt độ tin cậy cao.

1.7 Các biến thể

- Hybrid WOA (WOA-PSO, HWPSO, HPSO-WOA, WOA-Levy, WOA-GWO, WOA-BHC)
- Enhanced WOA (EWOA, MWOA, Distribution-based WOA)
- Binary-Multi-objective-Discrete WOA

1.8 Thuật toán lai WOA-PSO

1.8.1 Mục đích

Thuật toán WOA-PSO (Whale Optimization Algorithm kết hợp Particle Swarm Optimization) là một thuật toán lai (hybrid) giữa:

- **Whale Optimization Algorithm (WOA):** có cơ chế bao vây con mồi, tấn công xoắn ốc và tìm kiếm ngẫu nhiên; mạnh về khai thác (exploitation) quanh nghiệm tốt nhất hiện tại.
- **Particle Swarm Optimization (PSO):** dựa trên cập nhật vận tốc và vị trí của các hạt; nổi bật bởi tốc độ hội tụ và cơ chế "vận tốc có quán tính".

Mục tiêu của WOA-PSO là:

- Tận dụng khả năng khai thác quanh nghiệm tốt của WOA.
- Kết hợp với tốc độ hội tụ và khả năng tận dụng thông tin quá khứ (best cá nhân/toàn cục) của PSO.
- Cải thiện chất lượng nghiệm và giảm nguy cơ kẹt tại cực trị cục bộ so với WOA hoặc PSO thuần tuý.

Thuật toán đặc biệt hữu ích cho các bài toán:

- Phi tuyến, nhiều cực trị cục bộ, không có hoặc khó sử dụng đạo hàm.
- Cần sự cân bằng giữa khám phá (exploration) và khai thác (exploitation) cùng với tốc độ hội tụ cao.

1.8.2 Mô tả thuật toán

Trong WOA-PSO, mỗi cá thể được coi vừa là một "cá voi" vừa là một "hạt":

- Vị trí: $X_i = (x_{i1}, x_{i2}, ..., x_{iD})$.
- Vận tốc: $V_i = (v_{i1}, v_{i2}, ..., v_{iD})$.
- Best cá nhân: P_i (personal best).
- Best toàn cục: X^* (global best) trong quần thể.

Trong mỗi vòng lặp, mỗi cá thể được cập nhật theo cả hai cơ chế:
1. Pha PSO (cập nhật vận tốc và vị trí):

- Cập nhật vận tốc theo quy tắc PSO:

$$V_i = wV_i + c_1 r_1 (P_i - X_i) + c_2 r_2 (X^* - X_i)$$

  trong đó:

  - w là trọng số quán tính (có thể giảm dần theo thời gian).
  - c_1, c_2 là các hệ số nhận thức và xã hội.
  - $r_1, r_2 \sim U[0, 1]$ là các biến ngẫu nhiên đều.

- Cập nhật vị trí theo PSO:

$$X_i^{PSO} = X_i + V_i$$

2. Pha WOA (cập nhật theo bao vây / xoắn ốc / tìm kiếm ngẫu nhiên): Sử dụng cùng hệ số như trong WOA gốc:

$$a = 2 - 2 \cdot \frac{t}{MaxIter}$$
$$A = 2ar_1 - a, \quad C = 2r_2$$

với $r_1, r_2 \sim U[0, 1]$, cùng với $p \sim U[0, 1]$ và $l \sim U[-1, 1]$, b là hằng số điều khiển độ xoắn (thường $b = 1$).
Khi đó:

- Nếu $p < 0.5$ và $|A| < 1$ (cơ chế bao vây con mồi, khai thác quanh X^*):

$$D_i = |C \cdot X^* - X_i|, \quad X_i^{WOA} = X^* - A \cdot D_i$$

- Nếu $p < 0.5$ và $|A| \geq 1$ (cơ chế tìm kiếm ngẫu nhiên, khám phá):

$$D_i = |C \cdot X_{rand} - X_i|, \quad X_i^{WOA} = X_{rand} - A \cdot D_i$$

  với X_{rand} là một cá thể được chọn ngẫu nhiên trong quần thể.

- Nếu $p \geq 0.5$ (cơ chế tấn công theo đường xoắn ốc):

$$D_i' = |X^* - X_i|, \quad X_i^{WOA} = D_i' e^{bl} \cos(2\pi l) + X^*$$

3. Pha tổ hợp (hybridization): Từ hai nghiệm ứng viên X_i^{PSO} và X_i^{WOA}, vị trí mới của cá thể được xác định:

$$X_i = \alpha X_i^{WOA} + (1 - \alpha) X_i^{PSO}$$

với $\alpha \in [0, 1]$ là tham số lai hoá. Khi α gần 1, thuật toán thiên về hành vi WOA; khi α nhỏ, thuật toán gần với PSO hơn.

Trong một số biến thể khác, người ta có thể:

- Chọn nghiệm tốt hơn theo fitness giữa X_i^{WOA} và X_i^{PSO} thay vì tổ hợp tuyến tính.
- Hoặc sử dụng chiến lược luân phiên: vòng lặp chẵn dùng PSO, vòng lặp lẻ dùng WOA.

Sau pha tổ hợp, thuật toán kiểm tra biên (giới hạn) cho X_i, đánh giá lại fitness, cập nhật P_i và X^*, rồi chuyển sang vòng lặp tiếp theo.

1.8.3 Mã giả thuật toán WOA-PSO

```
Khởi tạo N cá thể Xi và vận tốc Vi ngẫu nhiên trong không gian tìm kiếm
For i = 1..N:
    Tính fitness(Xi)
    Pi = Xi  # best cá nhân ban đầu
X* = argmin fitness(Pi) # best toàn cục
t = 0
while (t < MaxIter):
    Cập nhật tham số WOA: a = 2 - 2 * t / MaxIter
    Cập nhật trọng số quán tính PSO: w = w_max - (w_max - w_min) * t / MaxIter
    For i = 1..N:
        # --- Pha PSO
        Sinh r1, r2 trong [0,1]
        Vi = w * Vi + c1 * r1 * (Pi - Xi) + c2 * r2 * (X* - Xi)
        X_PSO = Xi + Vi

        # --- Pha WOA
        Sinh r1, r2, p, l
        A = 2 * a * r1 - a; C = 2 * r2
        if p < 0.5:
            if |A| < 1:
                D = |C * X* - Xi|
                X_WOA = X* - A * D
            else:
                Chọn cá thể ngẫu nhiên X_rand
                D = |C * X_rand - Xi|
                X_WOA = X_rand - A * D
        else:
            D_prime = |X* - Xi|
            X_WOA = D_prime * exp(b * l) * cos(2 * PI * l) + X*
```

```
        # --- Pha tổ hợp
        X_new = alpha * X_WOA + (1 - alpha) * X_PSO

        # Kiểm tra biên và Cập nhật
        Hiệu chỉnh X_new nếu vượt ngoài [lb, ub]
        Xi = X_new
        fitness_i = fitness(Xi)

        # Cập nhật best cá nhân
        if fitness_i tốt hơn fitness(Pi): Pi = Xi

        # Cập nhật best toàn cục
        if fitness_i tốt hơn fitness(X*): X* = Xi
    t = t + 1
Xuất X* là nghiệm tối ưu
```

1.8.4 Độ phức tạp tính toán

Ký hiệu: N là số cá thể, T là số vòng lặp tối đa, D là số chiều, T_f là chi phí tính hàm mục tiêu ($\approx O(D)$). Phân tích:

- Khởi tạo: $O(ND)$.
- Mỗi vòng lặp:
  - Cập nhật tham số: $O(1)$.
  - Với mỗi cá thể: Cập nhật PSO $O(D)$, Cập nhật WOA $O(D)$, Tổ hợp $O(D)$, Tính fitness $O(D)$.

Tổng độ phức tạp thời gian:

$$O(T \times N \times D)$$

Cùng bậc với WOA hoặc PSO thuần, nhưng chi phí hằng số lớn hơn do mỗi vòng lặp phải thực hiện cả hai cập nhật và bước tổ hợp.

1.8.5 Hội tụ

Cơ chế hội tụ của WOA-PSO là sự kết hợp của:

- **WOA:** Tham số a giảm tuyến tính từ 2 về 0. Khi a lớn (đầu quá trình), $|A| \geq 1$ giúp khám phá mạnh. Khi a nhỏ (cuối quá trình), $|A| < 1$ giúp khai thác sâu quanh X^*.
- **PSO:** Vận tốc chịu ảnh hưởng của P_i và X^*, với w giảm dần. Đầu quá trình w lớn giúp di chuyển xa (khám phá), cuối quá trình w nhỏ giúp dao động quanh nghiệm tốt (hội tụ).

Nhờ pha tổ hợp, thuật toán vừa khám phá rộng ở giai đoạn đầu, vừa bị hút về X^* qua cả hai cơ chế ở giai đoạn sau, giúp hội tụ nhanh và tinh chỉnh nghiệm.

1.8.6 Nhược điểm

- **Nhiều tham số cần tinh chỉnh:** Ngoài tham số WOA (a, b) và PSO (w, c_1, c_2), còn thêm tham số lai hoá α.
- **Chi phí tính toán cao hơn:** Thời gian chạy thực tế tăng do thực hiện cả hai cơ chế.
- **Nguy cơ dao động hoặc hội tụ sớm:** Nếu tham số không hợp lý, quần thể có thể dao động hoặc hội tụ sớm vào cực trị cục bộ.

1.9 Thuật toán HWPSO (Hybrid Whale-Particle Swarm Optimization)

1.9.1 Mục đích

HWPSO (Hybrid Whale-Particle Swarm Optimization) là một biến thể lai giữa WOA và PSO, trong đó:

- Khung WOA được giữ làm xương sống chính: cá voi cập nhật dựa trên nghiệm tốt nhất X^*, thông qua cơ chế bao vây, tấn công xoắn ốc và tìm kiếm ngẫu nhiên.
- Thành phần PSO (vận tốc, best cá nhân) được tích hợp để bổ sung quán tính di chuyển và tận dụng thông tin lịch sử (P_i).

Mục tiêu của HWPSO là tăng tốc độ hội tụ của WOA, nâng cao khả năng khai thác mà vẫn giữ được khám phá không gian rộng, và giảm nguy cơ kẹt tại cực trị cục bộ.

1.9.2 Mô tả thuật toán

Trong HWPSO, mỗi cá thể có vị trí X_i, vận tốc V_i, best cá nhân P_i và best toàn cục X^*. Khác với WOA-PSO (nơi hai nghiệm ứng viên được trộn), HWPSO thường sử dụng vận tốc PSO để sinh hướng di chuyển, sau đó áp dụng quy tắc WOA rồi cộng thêm thành phần vận tốc để cập nhật vị trí.

Quy trình cơ bản trong mỗi vòng lặp:

1. **Cập nhật vận tốc (PSO-like):**

$$V_i = wV_i + c_1 r_1 (P_i - X_i) + c_2 r_2 (X^* - X_i)$$

   với giới hạn vận tốc $V_i \in [-V_{max}, V_{max}]$.

2. **Cập nhật theo WOA (tính vị trí ứng viên X_i^{WOA}):**
   - Cập nhật a, A, C.
   - Nếu $p < 0.5$ và $|A| < 1$: Bao vây con mồi.
   - Nếu $p < 0.5$ và $|A| \geq 1$: Tìm kiếm ngẫu nhiên.
   - Nếu $p \geq 0.5$: Tấn công xoắn ốc.

3. **Cập nhật vị trí bằng WOA + vận tốc:**

$$X_i^{new} = X_i^{WOA} + V_i$$

4. **Đánh giá và cập nhật best:** Cập nhật P_i và X^* nếu tìm được nghiệm tốt hơn.

1.9.3 Mã giả thuật toán HWPSO

```
Khởi tạo N cá thể Xi và vận tốc Vi ngẫu nhiên
For i = 1..N: Tính fitness(Xi); Pi = Xi
X* = argmin fitness(Pi)
t = 0
while (t < MaxIter):
    Cập nhật tham số WOA (a) và PSO (w)
    For i = 1..N:
        # --- Pha PSO: cập nhật vận tốc
        Sinh r3, r4 trong [0, 1]
        Vi = w * Vi + c1 * r3 * (Pi - Xi) + c2 * r4 * (X* - Xi)
        Giới hạn Vi trong [-Vmax, Vmax]

        # --- Pha WOA: cập nhật vị trí ứng viên
        Sinh r1, r2, p, l; A = 2 * a * r1 - a; C = 2 * r2
        if p < 0.5:
            if |A| < 1:
                D = |C * X* - Xi|
                X_WOA = X* - A * D
            else:
```

```
                Chọn ngẫu nhiên X_rand
                D = |C * X_rand - Xi|
                X_WOA = X_rand - A * D
        else:
            D_prime = |X* - Xi|
            X_WOA = D_prime * exp(b * l) * cos(2 * PI * l) + X*

        # --- Cập nhật vị trí bằng WOA + vận tốc
        X_new = X_WOA + Vi
        Hiệu chỉnh X_new nếu vượt ngoài [lb, ub]

        # --- Đánh giá và cập nhật best
        Xi = X_new
        fitness_i = fitness(Xi)
        if fitness_i tốt hơn fitness(Pi): Pi = Xi
        if fitness_i tốt hơn fitness(X*): X* = Xi
    t = t + 1
Xuất X* là nghiệm tối ưu
```

1.9.4 Độ phức tạp tính toán

Tương tự như WOA-PSO:

$$O(T \times N \times D)$$

Chi phí hằng số cao hơn WOA thuần do phải tính thêm bước vận tốc PSO.

1.9.5 Hội tụ

HWPSO kết hợp cơ chế chuyển pha của WOA và quán tính của PSO.

- **Giai đoạn đầu:** a lớn, w cao $\rightarrow$ bước nhảy lớn, khám phá rộng.
- **Giai đoạn cuối:** a nhỏ, w giảm $\rightarrow$ vừa bao vây/xoắn ốc quanh X^*, vừa di chuyển với vận tốc nhỏ, giúp hội tụ nhanh về nghiệm tốt.

1.9.6 Nhược điểm

- **Nhiều tham số:** Phải tinh chỉnh đồng thời tham số WOA và PSO.
- **Chi phí tính toán tăng:** Do thêm bước cập nhật vận tốc.
- **Nguy cơ dao động:** Nếu hệ số quán tính hoặc xã hội quá lớn, quần thể có thể dao động quanh X^* mà không hội tụ mượt.

1.10 Thuật toán HPSO-WOA (Hybrid PSO-Whale Optimization Algorithm)

1.10.1 Mục đích

HPSO-WOA là một thuật toán lai trong đó:

- PSO đóng vai trò khung chính (vận tốc, cập nhật vị trí, best cá nhân/toàn cục).
- WOA được chèn vào như một bước cải tiến (local search/exploitation) trên một số cá thể (thường là các cá thể tốt).

Mục tiêu là khai thác tốc độ hội tụ cao của PSO trong việc tìm vùng nghiệm tốt, đồng thời sử dụng cơ chế bao vây và tấn công xoắn ốc của WOA để tinh chỉnh nghiệm quanh best toàn cục và cải thiện khả năng thoát khỏi cực trị cục bộ.

1.10.2 Mô tả thuật toán

Trong HPSO-WOA, ta vẫn sử dụng vị trí X_i, vận tốc V_i, best cá nhân P_i và best toàn cục X^*. Mỗi vòng lặp gồm hai pha chính:

1. **Pha PSO (cập nhật toàn bộ quần thể):** Cập nhật trọng số quán tính w giảm dần. Với mỗi hạt i:

$$V_i = wV_i + c_1 r_1 (P_i - X_i) + c_2 r_2 (X^* - X_i)$$

$$X_i = X_i + V_i$$

Sau đó kiểm tra biên, tính fitness và cập nhật P_i, X^*.

2. **Pha WOA (cải tiến một tập con hạt):**

- Chọn một tập con S gồm K hạt để áp dụng WOA.
- Cập nhật tham số WOA (a, A, C).
- Với mỗi hạt $i \in S$:
  - Nếu $p < 0.5$ và $|A| < 1$: Bao vây X^*.
  - Nếu $p < 0.5$ và $|A| \geq 1$: Tìm kiếm quanh X_{rand}.
  - Nếu $p \geq 0.5$: Tấn công xoắn ốc quanh X^*.
- So sánh và cập nhật: Nếu $f(X_i^{WOA})$ tốt hơn $f(X_i)$ hiện tại thì thay thế X_i và cập nhật P_i, X^* nếu cần.

1.10.3 Mã giả thuật toán HPSO-WOA

```
Khởi tạo N hạt Xi và vận tốc Vi
For i = 1..N: Tính fitness(Xi); Pi = Xi
X* = argmin fitness(Pi)
t = 0
while (t < MaxIter):
    # --- Pha PSO
    w = w_max - (w_max - w_min) * t / MaxIter
    For i = 1..N:
        Sinh r1, r2 trong [0, 1]
        Vi = w * Vi + c1 * r1 * (Pi - Xi) + c2 * r2 * (X* - Xi)
        Xi = Xi + Vi
        Hiệu chỉnh Xi, tính fitness, cập nhật Pi, X*

    # --- Pha WOA trên K hạt được chọn
    a = 2 - 2 * t / MaxIter
    Chọn tập S gồm K chỉ số hạt (ví dụ: K hạt tốt nhất)
    For mỗi i trong S:
        Sinh r1, r2, p, l; A = 2 * a * r1 - a; C = 2 * r2
        if p < 0.5:
            if |A| < 1:
                D = |C * X* - Xi|
                X_WOA = X* - A * D
            else:
                Chọn ngẫu nhiên X_rand
                D = |C * X_rand - Xi|
                X_WOA = X_rand - A * D
        else:
            D_prime = |X* - Xi|
            X_WOA = D_prime * exp(b * l) * cos(2 * PI * l) + X*

        Hiệu chỉnh X_WOA
        if fitness(X_WOA) tốt hơn fitness(Xi):
            Xi = X_WOA
```

```
            Cập nhật Pi, X* nếu cần
    t = t + 1
Xuất X* là nghiệm tối ưu
```

1.10.4 Độ phức tạp tính toán

Độ phức tạp theo bậc lớn: $O(T \times N \times D)$. Giống WOA/PSO thuần. Hằng số ẩn tăng thêm do pha WOA, nhưng nếu $K \ll N$ thì chi phí bổ sung là tương đối nhỏ.

1.10.5 Hội tụ

HPSO-WOA tận dụng tốc độ hội tụ của PSO (giai đoạn đầu w lớn) và khả năng tinh chỉnh cục bộ của WOA trên tập hạt tốt (giai đoạn sau). Bằng việc áp dụng WOA chỉ trên các hạt tốt, thuật toán tập trung khai thác quanh vùng có triển vọng mà vẫn duy trì khả năng khám phá từ PSO cho các hạt còn lại.

1.10.6 Nhược điểm

- Thiết kế lai phức tạp: Phải chọn K, tần suất áp dụng WOA.
- Nhiều tham số cần tinh chỉnh: Tham số PSO, WOA và tham số lai.
- Chi phí tính toán tăng so với PSO thuần.

1.11 Thuật toán WOA-Levy

1.11.1 Mục đích

WOA-Levy là một biến thể của WOA được tăng cường bằng Lévy flight (bước nhảy Lévy). Mục tiêu là tăng cường khả năng khám phá không gian (exploration) và giúp quần thể "nhảy xa" ra khỏi vùng đang tìm kiếm để thoát khỏi các cực trị cục bộ.

1.11.2 Mô tả thuật toán

WOA-Levy giữ nguyên khung WOA. Khác biệt chính là thêm bước Lévy vào cập nhật vị trí, thường trong pha tìm kiếm ngẫu nhiên (khi $|A| \geq 1$). Một vector Lévy $L \in \mathbb{R}^D$ được sinh theo thuật toán Mantegna. Trong thực hành, ta thường dùng:

$$X_i^{new} = X_i^{WOA} + step\_size \cdot L$$

với $step\_size$ là hệ số tỉ lệ nhỏ.

Cập nhật trong WOA-Levy:

- Nếu $p < 0.5$:
  - Nếu $|A| < 1$ (Bao vây): Dùng WOA gốc hoặc thêm Lévy rất nhỏ.
  - Nếu $|A| \geq 1$ (Tìm kiếm ngẫu nhiên): Tính X_i^{WOA} theo cơ chế tìm kiếm, sau đó cộng thêm thành phần Lévy: $X_i^{new} = X_i^{WOA} + step\_size \cdot L$.
- Nếu $p \geq 0.5$ (Xoắn ốc): Dùng WOA gốc hoặc thêm Lévy nhẹ.

1.11.3 Mã giả thuật toán WOA-Levy

```
Khởi tạo quần thể Xi
Tính fitness(Xi), X* = cá thể tốt nhất
t = 0
while (t < MaxIter):
    a = 2 - 2 * t / MaxIter
    For i = 1..N:
        Sinh r1, r2, p, l; A = 2 * a * r1 - a; C = 2 * r2
        if p < 0.5:
            if |A| < 1: # Bao vây X*
```

```
                For d = 1..D:
                    D_i(d) = |C * X*(d) - Xi(d)|
                    X_WOA(d) = X*(d) - A * D_i(d)
                X_new = X_WOA
            else: # Tìm kiếm ngẫu nhiên + Lévy flight
                Chọn k ngẫu nhiên; X_rand = Xk
                For d = 1..D:
                    D_i(d) = |C * X_rand(d) - Xi(d)|
                    X_WOA(d) = X_rand(d) - A * D_i(d)
                # Sinh vector Lévy L
                L = Levy(beta)
                For d = 1..D:
                    X_new(d) = X_WOA(d) + step_size * L(d)
        else: # Tấn công xoắn ốc
            For d = 1..D:
                D_prime(d) = |X*(d) - Xi(d)|
                X_WOA(d) = D_prime(d) * exp(b * l) * cos(2 * PI * l) + X*(d)
            X_new = X_WOA

        Kiểm tra biên, cập nhật Xi, fitness, X*
    t = t + 1
Xuất X*
```

1.11.4 Độ phức tạp tính toán

$O(T \times N \times D)$. Tương tự WOA gốc; chi phí hằng số tăng lên do phải sinh thêm các bước Lévy.

1.11.5 Hội tụ

Khi kết hợp với pha $|A| \geq 1$ (tìm kiếm ngẫu nhiên), Lévy flight làm tăng đáng kể khả năng thoát khỏi local optimum và mở rộng vùng tìm kiếm toàn cục. Giai đoạn đầu khám phá rất rộng; giai đoạn sau nếu $step\_size$ giảm dần, Lévy trở nên ôn hoà, hỗ trợ tinh chỉnh.

1.11.6 Nhược điểm

- Nhạy cảm với tham số Lévy (như $\beta, step\_size$).
- Chi phí tính toán tăng do sinh vector Lévy.
- Khó kiểm soát quỹ đạo tìm kiếm do các bước nhảy lớn không dự đoán được.

1.12 Thuật toán WOA-GWO

1.12.1 Mục đích

WOA-GWO là thuật toán lai kết hợp WOA và Grey Wolf Optimizer (GWO). Mục tiêu là tận dụng cơ chế đa leader (α, β, δ) của GWO để giảm phụ thuộc vào duy nhất một nghiệm tốt nhất, kết hợp với bao vây xoắn ốc của WOA để cải thiện khả năng khai thác.

1.12.2 Mô tả thuật toán

Trong WOA-GWO, các cá thể được cập nhật bằng hai cơ chế song song: 1. **Pha GWO (đa leader):** Tính toán vị trí ứng viên X_i^{GWO} dựa trên trung bình vị trí của 3 con sói đầu đàn (α, β, δ).

$$X_i^{GWO} = \frac{X_1 + X_2 + X_3}{3}$$

2. **Pha WOA (bao vây/xoắn ốc):** Tính toán vị trí ứng viên X_i^{WOA} dựa trên cơ chế WOA (theo X_α đóng vai trò X^*). 3. **Pha tổ hợp:** Chọn nghiệm có fitness tốt hơn giữa X_i^{GWO} và X_i^{WOA} để cập nhật X_i.

1.12.3 Mã giả thuật toán WOA-GWO

```
Khởi tạo N cá thể Xi ngẫu nhiên
Tính fitness(Xi), xác định alpha, beta, delta
t = 0
while (t < MaxIter):
    a = 2 - 2 * t / MaxIter
    For i = 1..N:
        # --- 1) Ứng viên theo GWO
        Tính A1, C1, A2, C2, A3, C3 và X1, X2, X3 theo alpha, beta, delta
        X_GWO = (X1 + X2 + X3) / 3

        # --- 2) Ứng viên theo WOA
        Sinh r1, r2, p, l; A = 2 * a * r1 - a; C = 2 * r2
        if p < 0.5:
            if |A| < 1:
                D_i = |C * X_alpha - Xi|
                X_WOA = X_alpha - A * D_i
            else:
                Chọn ngẫu nhiên X_rand
                D_i = |C * X_rand - Xi|
                X_WOA = X_rand - A * D_i
        else:
            D_prime = |X_alpha - Xi|
            X_WOA = D_prime * exp(b * l) * cos(2 * PI * l) + X_alpha

        # --- 3) Chọn nghiệm tốt hơn
        Hiệu chỉnh biên
        if fitness(X_GWO) <= fitness(X_WOA): Xi = X_GWO
        else: Xi = X_WOA

    Cập nhật fitness, chọn lại alpha, beta, delta
    t = t + 1
Xuất X_alpha
```

1.12.4 Độ phức tạp tính toán

$O(T \times N \times D)$. Cùng bậc với WOA và GWO thuần, nhưng chi phí hằng số cao hơn do phải tính đồng thời cả cập nhật GWO và WOA.

1.12.5 Hội tụ

WOA-GWO hưởng lợi từ cơ chế đa leader của GWO (giảm phụ thuộc vào một nghiệm duy nhất) và cơ chế bao vây xoắn ốc của WOA (khai thác sâu giai đoạn sau). Việc luôn chọn nghiệm tốt hơn ở pha tổ hợp giúp tăng xác suất cải thiện fitness.

1.12.6 Nhược điểm

- Tăng độ phức tạp thiết kế và nhiều tham số cần điều chỉnh.
- Chi phí tính toán tăng (tính cả hai cơ chế mỗi vòng).

1.13 Thuật toán WOA-BHC

1.13.1 Mục đích

WOA-BHC (Whale Optimization Algorithm với Biased/Boundary Hunting Component) bổ sung thành phần BHC để điều chỉnh hướng tìm kiếm theo cách có chủ đích (biased), tăng cường khai thác quanh vùng nghiệm tốt và khuyến khích khám phá vùng biên (boundary).

1.13.2 Mô tả thuật toán

WOA-BHC phân loại các cá thể theo fitness và khoảng cách tới biên, sau đó gán vai trò:

- **EXPLOITER (Cá voi tốt):** Ưu tiên khai thác quanh X^* (bao vây mạnh hoặc xoắn ốc thay vì tìm kiếm ngẫu nhiên).
- **BOUNDARY EXPLORER (Cá voi gần biên):** Dò biên có định hướng về phía mục tiêu T nằm giữa biên và X^*.
- **RANDOM EXPLORER (Cá voi tệ):** Tăng cường khám phá mạnh.
- **EXPLORER (Cá voi bình thường):** Dùng WOA chuẩn.

1.13.3 Mã giả thuật toán WOA-BHC

```
Khởi tạo quần thể Xi
For i = 1..N: Tính fitness(Xi); X* = cá thể tốt nhất
t = 0
while (t < MaxIter):
    a = 2 - 2 * t / MaxIter
    # --- BHC: Phân loại
    Sắp xếp cá thể theo fitness -> Tập GOOD, BAD
    Phân loại theo biên -> is_boundary(i)
    Gán vai trò role(i) (EXPLOITER, BOUNDARY, RANDOM, EXPLORER)

    # --- Cập nhật vị trí theo vai trò
    For i = 1..N:
        if role(i) == EXPLOITER:
            Cập nhật bao vây hoặc xoắn ốc quanh X* (bỏ qua tìm kiếm ngẫu nhiên)
        else if role(i) == BOUNDARY_EXPLORER:
            Xác định biên gần nhất B_d, mục tiêu T_d
            Cập nhật hướng về T_d
        else if role(i) == RANDOM_EXPLORER:
            Tìm kiếm ngẫu nhiên mạnh
        else: # EXPLORER
            Cập nhật theo WOA chuẩn

        Kiểm tra biên, cập nhật Xi, fitness, X*
    t = t + 1
Xuất X*
```

1.13.4 Độ phức tạp tính toán

$O(T \times (ND + N \log N)) \approx O(TND)$. Chi phí hằng số cao hơn do bước phân loại và xử lý biên.

1.13.5 Hội tụ

Hội tụ tốt hơn trong không gian nhiều cực trị cục bộ nhờ sự phân vai trò: Exploiter tăng tốc khai thác, Boundary Explorer tránh bỏ sót nghiệm biên, Random Explorer duy trì đa dạng.

1.13.6 Nhược điểm

- Nhiều tham số bổ sung cần tinh chỉnh ($P_{good}, Q_{bad}, \alpha_b$).
- Thiết kế vai trò phức tạp.
- Chi phí tính toán cao hơn WOA thuần.

1.14 Thuật toán EWOA (Enhanced Whale Optimization Algorithm)

1.14.1 Mục đích

EWOA là họ các biến thể cải tiến của WOA nhằm tăng tốc độ hội tụ, cải thiện khả năng thoát khỏi cực trị cục bộ và duy trì đa dạng quần thể thông qua điều chỉnh tham số, đột biến và tái khởi tạo.

1.14.2 Mô tả thuật toán

Khác biệt nằm ở ba thành phần chính: 1. **Điều chỉnh tham số a phi tuyến:** Thay thế giảm tuyến tính bằng dạng phi tuyến (ví dụ parabolic $a = 2(1 - (t/MaxIter)^2)$) để a giảm chậm ở đầu (khám phá mạnh) và nhanh ở cuối (khai thác mạnh). 2. **Bổ sung mutation (đột biến):** Thêm nhiễu Gaussian có kiểm soát vào vị trí sau cập nhật WOA ($X_i^{mut} = X_i^{WOA} + \sigma_t Z$) và chọn nghiệm tốt hơn. 3. **Tái khởi tạo (Reinitialization):** Tái khởi tạo ngẫu nhiên các cá thể không cải thiện sau một khoảng thời gian (T_{stall}).

1.14.3 Mã giả thuật toán EWOA

```
Khởi tạo N cá thể Xi
For i = 1..N: fitness_i = fitness(Xi); no_improve_count(i) = 0
X* = cá thể tốt nhất
t = 0
while (t < MaxIter):
    a = 2 * (1 - (t / MaxIter)^2) # a phi tuyến
    sigma_t = sigma0 * (1 - t / MaxIter)
    For i = 1..N:
        # --- Cập nhật theo WOA cơ bản
        Tính X_WOA theo (Bao vây / Tìm kiếm / Xoắn ốc)
        Kiểm tra biên

        # --- Mutation
        Sinh r_mut; X_cand = X_WOA
        if r_mut < p_mut:
            X_mut = X_WOA + sigma_t * randn()
            Kiểm tra biên, tính fitness
            if fitness(X_mut) < fitness(X_WOA): X_cand = X_mut

        # --- Reinitialization
        if no_improve_count(i) >= T_stall:
            X_cand = Random_Position()
            no_improve_count(i) = 0

        # --- Cập nhật cá thể
        if fitness(X_cand) < fitness_i:
            Xi = X_cand; no_improve_count(i) = 0
        else:
            no_improve_count(i) += 1

        if fitness(Xi) < fitness(X*): X* = Xi

    t = t + 1
Xuất X*
```

1.14.4 Độ phức tạp tính toán

$O(T \times N \times D)$. Giống WOA gốc về bậc lớn, nhưng chi phí hằng số cao hơn do các bước mutation và reinitialization.

1.14.5 Hội tụ

EWOA kết hợp a phi tuyến (cân bằng exploration/exploitation tốt hơn), mutation (tạo bước nhảy nhỏ/lớn để thoát local optimum) và reinitialization (tránh co cụm quá sớm). Nhờ đó, EWOA thường hội tụ nhanh hơn và ổn định hơn WOA gốc.

1.14.6 Nhược điểm

- Nhiều tham số cần tinh chỉnh ($p_{mut}, \sigma_0, T_{stall}$).
- Chi phí tính toán tăng.
- Nguy cơ phá vỡ hội tụ nếu mutation quá mạnh.

1.15 Thuật toán MWOA (Memetic Whale Optimization Algorithm)

1.15.1 Mục đích

MWOA kết hợp tìm kiếm toàn cục (global search) của WOA với một thuật toán tìm kiếm cục bộ (local search / memetic) để tinh chỉnh nghiệm quanh các cá thể tốt. Mục tiêu là cải thiện độ chính xác nghiệm cuối và tăng tốc độ hội tụ.

1.15.2 Mô tả thuật toán

MWOA giữ nguyên khung WOA. Điểm khác biệt là sau mỗi (hoặc định kỳ) vòng lặp, MWOA áp dụng local search (ví dụ: hill-climbing) quanh một số nghiệm tốt (thường là X^*) để cải thiện thêm. Quy trình: 1. **Pha WOA:** Cập nhật quần thể theo các cơ chế WOA chuẩn. 2. **Pha Memetic:** Chọn tập nghiệm (ví dụ X^*), áp dụng LocalSearch(X) để tìm X_{loc}. Nếu X_{loc} tốt hơn, thay thế X.

1.15.3 Mã giả thuật toán MWOA

```
Khởi tạo N cá thể Xi; X* = cá thể tốt nhất
t = 0
while (t < MaxIter):
    # --- Pha WOA (Global search)
    Cập nhật a, A, C, p, l
    For i = 1..N:
        Cập nhật Xi theo (Bao vây / Tìm kiếm / Xoắn ốc)
        Kiểm tra biên, cập nhật X*

    # --- Pha Memetic (Local search)
    if (t % local_interval == 0):
        X_loc = LocalSearch(X*)  # Ví dụ Hill-Climbing
        if fitness(X_loc) < fitness(X*):
            X* = X_loc

    t = t + 1
Xuất X*
```

1.15.4 Độ phức tạp tính toán

$O(TND) + O(T_{eff}ML_{max}D)$. Chi phí thực tế cao hơn WOA thuần do thêm pha local search (đánh giá hàm mục tiêu nhiều lần trong mỗi bước local search).

1.15.5 Hội tụ

MWOA kết hợp khả năng phát hiện vùng tốt của WOA và khả năng khai thác sâu của local search. Với cấu hình phù hợp, MWOA thường hội tụ nhanh hơn và cho nghiệm tốt hơn đáng kể.

1.15.6 Nhược điểm

- Chi phí tính toán tăng rõ rệt.
- Cần thiết kế local search phù hợp bài toán.
- Nhiều tham số phải chọn ($L_{max}, step\_size$).

1.16 Biến thể dựa trên Phân phối (Distribution-based WOA)

1.16.1 Mục đích

D-WOA sử dụng phân phối xác suất (thường là Gaussian) mô tả phân bố của các nghiệm tốt (elite) để sinh nghiệm mới, thay vì chỉ dùng công thức điểm-điểm. Mục tiêu là khai thác thông tin thống kê của quần thể để dẫn hướng tìm kiếm.

1.16.2 Mô tả thuật toán

- Chọn tập elite (top K cá thể tốt nhất).
- Ước lượng phân phối Gaussian $N(\mu, \sigma^2)$ từ tập elite theo từng chiều.
- Một phần quần thể (N_{dist}) cập nhật bằng cách lấy mẫu từ phân phối: $X_i^{new}(d) = \mu(d) + \sigma(d)Z_d$.
- Phần còn lại cập nhật theo WOA chuẩn.

1.16.3 Mã giả thuật toán Distribution-based WOA

```
Khởi tạo N cá thể Xi; t = 0
while (t < MaxIter):
    # Ước lượng phân phối từ top K elite
    Sắp xếp cá thể theo fitness -> Elite
    Tính mean (mu) và std (sigma) từ Elite

    # Cập nhật vị trí
    For i = 1..N:
        if i thuộc nhóm Distribution-based:
            Sinh Z ~ N(0, 1)
            Xi_new = mu + sigma * Z
        else:
            Cập nhật theo WOA chuẩn (Bao vây / Tìm kiếm / Xoắn ốc)

    Kiểm tra biên, cập nhật fitness, X*
    t = t + 1
Xuất X*
```

1.16.4 Độ phức tạp tính toán

$O(T \cdot (ND + N \log N + DK))$. Chi phí hằng số lớn hơn WOA thuần do bước sắp xếp và ước lượng phân phối.

1.16.5 Hội tụ

Giai đoạn đầu phân phối rộng hỗ trợ khám phá. Giai đoạn sau elite co cụm, σ nhỏ dần hỗ trợ khai thác sâu vùng tối ưu. Tốc độ hội tụ thường nhanh hơn WOA gốc trong các bài toán có cấu trúc cụm nghiệm tốt.

1.16.6 Nhược điểm

- Phụ thuộc vào giả định phân phối (thường là Gaussian độc lập).
- Cần chọn tham số K và tỷ lệ p_{dist}.
- Chi phí tính toán tăng.

1.17 Binary WOA (BWOA)

1.17.1 Mục đích

BWOA dùng cho các bài toán tối ưu biến rời rạc hoặc nhị phân (ví dụ: chọn đặc trưng, ba lô 0-1) nơi WOA gốc không áp dụng được.

1.17.2 Mô tả thuật toán

Sử dụng không gian liên tục trung gian Y_i. Cập nhật Y_i theo công thức WOA chuẩn. Sau đó dùng hàm chuyển đổi (transfer function, ví dụ Sigmoid) để ánh xạ Y_i sang xác suất và quyết định bit 0/1.

$$S(v) = \frac{1}{1 + e^{-v}}$$

$$x_{ij}^{new} = 1 \text{ nếu } rand() < S(y_{ij}^{new}), \text{ ngược lại } 0$$

1.17.3 Mã giả thuật toán BWOA

```
Khởi tạo N cá thể nhị phân Xi; Yi = Xi (liên tục)
t = 0
while (t < MaxIter):
    For i = 1..N:
        # Cập nhật Yi theo WOA chuẩn
        Tính Y_new dựa trên Yi, X*, A, C (Bao vây / Tìm kiếm / Xoắn ốc)

        # Chuyển sang nhị phân
        For j = 1..D:
            S = 1 / (1 + exp(-Y_new(j)))
            if rand() < S: X_new(j) = 1
            else:          X_new(j) = 0

        Xi = X_new; Yi = Y_new
        Cập nhật fitness, X*
    t = t + 1
Xuất X*
```

1.17.4 Độ phức tạp tính toán

$O(T \times N \times D)$. Giống WOA liên tục, chỉ thêm chi phí tính hàm sigmoid.

1.17.5 Hội tụ

Đầu quá trình Y_i phân tán, xác suất quanh 0.5 giúp khám phá. Cuối quá trình Y_i lớn, xác suất gần 0 hoặc 1 giúp bit ổn định, quần thể hội tụ.

1.17.6 Nhược điểm

- Nhạy cảm với hàm chuyển đổi.
- Không gian nhị phân rời rạc, cập nhật liên tục chỉ là xấp xỉ.

1.18 Multi-Objective WOA (MOWOA)

1.18.1 Mục đích

MOWOA giải quyết các bài toán đa mục tiêu, tìm tập nghiệm Pareto tối ưu (không bị trội) thay vì một nghiệm duy nhất.

1.18.2 Mô tả thuật toán

- Sử dụng kho lưu trữ (**Archive**) chứa các nghiệm không bị trội.
- Cá thể chọn **leader** từ Archive để cập nhật vị trí.
- Sử dụng **Crowding Distance** để duy trì đa dạng khi Archive đầy.

1.18.3 Mã giả thuật toán MOWOA

```
Khởi tạo quần thể Xi; Archive A rỗng
t = 0
while (t < MaxIter):
    For i = 1..N:
        Chọn Leader từ A (dựa trên crowding distance)
        Cập nhật Xi theo WOA quanh Leader
        Kiểm tra biên

    Tính Vector Mục tiêu F(Xi_new)
    Cập nhật Archive A:
        Thêm nghiệm mới, loại bỏ nghiệm bị trội
        Nếu A đầy: Loại bỏ nghiệm có Crowding Distance nhỏ nhất
    t = t + 1
Xuất A (Tập Pareto)
```

1.18.4 Độ phức tạp tính toán

$O(T \cdot (ND + K^2 + MK \log K))$ với K là kích thước archive + quần thể. Chi phí cao hơn do sắp xếp không bị trội.

1.18.5 Hội tụ

Hội tụ về biên Pareto thật nhờ cơ chế WOA. Duy trì đa dạng nhờ Archive và Crowding Distance.

1.18.6 Nhược điểm

Chi phí tính toán cao; nhạy với kích thước Archive.

1.19 Discrete WOA

1.19.1 Mục đích

Dùng cho các bài toán tối ưu rời rạc hoán vị (như TSP, lập lịch).

1.19.2 Mô tả thuật toán

Thay thế các phép toán vector bằng toán tử rời rạc:

- **MoveTowards:** Sử dụng Swap/Insert để giảm sự khác biệt giữa hai hoán vị.
- **Xoắn ốc:** Tiến gần mục tiêu rồi áp dụng nhiễu cục bộ (Reverse đoạn).

1.19.3 Mã giả Discrete WOA

```
Khởi tạo N hoán vị Xi
t = 0
while (t < MaxIter):
    For i = 1..N:
        Tính số bước di chuyển s_i dựa trên |A|
        if p < 0.5:
            if |A| < 1: Target = X*
            else:       Target = X_rand
```

```
            Xi_new = MoveTowards(Xi, Target, s_i)
        else:
            Xi_temp = MoveTowards(Xi, X*, s_i)
            Xi_new = Reverse_Mutation(Xi_temp) # Xoắn ốc rời rạc

        Cập nhật Xi, X*
    t = t + 1
Xuất X*
```

1.19.4 Độ phức tạp tính toán

$O(T \times N \times n)$ với n là kích thước hoán vị. Phụ thuộc vào chi phí của toán tử MoveTowards.

1.19.5 Hội tụ

Đầu quá trình bước di chuyển lớn (khám phá). Cuối quá trình bước di chuyển nhỏ, tập trung quanh X^*.

1.19.6 Nhược điểm

Không có chuẩn duy nhất; chi phí cao với hoán vị lớn; dễ kẹt local optimum nếu thiếu cơ chế đa dạng.

2 Phân Tích Bài Toán Phân Bổ Tài Nguyên Trong Mạng Vô Tuyến

Dựa trên bài báo khoa học nền tảng *"Whale Optimization Algorithm with Applications to Resource Allocation in Wireless Networks"* (Phạm et al., IEEE Transactions on Vehicular Technology), phần này đi sâu phân tích mô hình toán học của hệ thống mạng giao thoa và các kỹ thuật xử lý ràng buộc chuyên sâu.

2.1 Tổng quan về Mạng Vô tuyến bị hạn chế bởi Nhiễu (Interference-Limited Networks)

Trong các mạng vô tuyến hiện đại (như 5G/6G mật độ cao), hiệu năng hệ thống không còn bị giới hạn bởi tạp âm nền (Noise) mà chủ yếu bị kìm hãm bởi nhiễu giao thoa (Interference) từ các người dùng cùng tần số. Bài báo xây dựng mô hình hệ thống dựa trên giả định này.

2.1.1 Mô hình Kênh truyền và Tín hiệu

Xét một hệ thống gồm M cặp liên kết (Link) người dùng cùng hoạt động trên một băng tần. Mỗi liên kết i bao gồm một máy phát (Transmitter - Tx) và một máy thu (Receiver - Rx).

- Gọi p_i là công suất phát của người dùng thứ i. Đây là biến số quyết định (decision variable) cần tối ưu hóa.
- Gọi G_{ij} là hệ số kênh truyền (Channel Gain) từ máy phát j đến máy thu i.
  - Khi $i = j$: G_{ii} là lợi ích kênh trực tiếp (Direct Link), mong muốn giá trị này càng lớn càng tốt.
  - Khi $i \neq j$: G_{ij} là kênh giao thoa chéo (Cross-link Interference), mong muốn giá trị này càng nhỏ càng tốt.
- Gọi n_0 là công suất tạp âm cộng (AWGN) tại máy thu.

2.1.2 Công thức tính Tỷ số Tín hiệu trên Nhiễu (SINR)

Chất lượng tín hiệu tại máy thu i được đo lường bằng tỷ số SINR (γ_i). Biểu thức toán học đầy đủ của SINR được xác định như sau:

$$\gamma_i(\mathbf{p}) = \frac{\text{Signal Power}}{\text{Noise Power} + \text{Interference Power}} = \frac{p_i G_{ii}}{n_0 + \sum_{j=1, j\neq i}^{M} p_j G_{ij}} \tag{8}$$

Trong đó, thành phần tổng $\sum_{j\neq i} p_j G_{ij}$ đại diện cho nhiễu đa người dùng (Multi-user Interference). Đây là nguyên nhân chính khiến bài toán trở nên phi tuyến (Non-linear) và không lồi (Non-convex), vì việc tăng công suất p_i để cải thiện chất lượng cho người dùng i sẽ ngay lập tức làm tăng nhiễu và giảm chất lượng của tất cả các người dùng khác.

2.1.3 Tốc độ Dữ liệu (Achievable Data Rate)

Dựa trên định lý Shannon, tốc độ dữ liệu đạt được của người dùng i (tính bằng bps/Hz) là:

$$R_i(\mathbf{p}) = \log_2(1 + \gamma_i(\mathbf{p})) = \log_2\left(1 + \frac{p_i G_{ii}}{n_0 + \sum_{j\neq i} p_j G_{ij}}\right) \tag{9}$$

2.2 Bài toán Cân bằng Hiệu quả Năng lượng và Phổ tần (EE-SE Tradeoff)

Bài báo tập trung giải quyết bài toán tối ưu hóa Hiệu quả Năng lượng Toàn cục (Global Energy Efficiency - GEE), một chỉ số quan trọng trong mạng xanh (Green Communications).

2.2.1 Mô hình Tiêu thụ Năng lượng Tổng thể

Tổng công suất tiêu thụ của toàn mạng (P_{total}) không chỉ là tổng công suất phát sóng vô tuyến, mà còn bao gồm các thành phần tiêu hao phần cứng. Mô hình tiêu thụ năng lượng được biểu diễn:

$$P_{total}(\mathbf{p}) = \sum_{i=1}^{M} P_i = \sum_{i=1}^{M} (\xi_i p_i + P_{c,i}) \tag{10}$$

Trong đó:

- $\xi_i > 1$: Hệ số không hiệu quả của bộ khuếch đại công suất (Power Amplifier Inefficiency Factor). Ví dụ, $\xi = 5$ nghĩa là để phát 1W ra ăng-ten, thiết bị tiêu tốn 5W điện năng.
- $P_{c,i}$: Công suất tiêu thụ tĩnh của mạch điện tử (Circuit Power), độc lập với công suất phát (ví dụ: công suất nuôi chip xử lý, bộ lọc, mixer).

2.2.2 Thiết lập Bài toán Tối ưu (Problem Formulation)

Mục tiêu là tìm vector công suất $\mathbf{p} = [p_1, \ldots, p_M]$ sao cho tỷ số giữa Tổng tốc độ dữ liệu và Tổng năng lượng tiêu thụ là lớn nhất.

Hàm mục tiêu (Objective Function):

$$\max_{\mathbf{p}} \quad \text{GEE}(\mathbf{p}) = \frac{\sum_{i=1}^{M} R_i(\mathbf{p})}{P_{total}(\mathbf{p})} = \frac{\sum_{i=1}^{M} \log_2(1 + \text{SINR}_i(\mathbf{p}))}{\sum_{i=1}^{M} (\xi_i p_i + P_{c,i})} \tag{11}$$

Hệ thống các ràng buộc (Constraints): Bài toán chịu sự chi phối của hai loại ràng buộc vật lý và chất lượng dịch vụ:

1. **Ràng buộc QoS (C1):** Mỗi người dùng yêu cầu một tốc độ tối thiểu R_{min} để duy trì kết nối ổn định (ví dụ: Voice cần ít nhất 64kbps).

$$R_i(\mathbf{p}) \geq R_i^{req}, \quad \forall i = 1, \ldots, M \tag{12}$$

2. **Ràng buộc Công suất (C2):** Công suất phát của mỗi người dùng bị giới hạn bởi ngưỡng bão hòa của thiết bị (P_{max}) và không được nhận giá trị âm.

$$0 \leq p_i \leq P_i^{max}, \quad \forall i = 1, \ldots, M \tag{13}$$

2.2.3 Phân tích Độ phức tạp của Bài toán

Bài toán trên thuộc lớp bài toán **Phi tuyến phân thức (Non-linear Fractional Programming)**.

- **Tính không lồi (Non-convexity):** Hàm SINR nằm trong hàm logarit ở tử số, lại bị chia cho một hàm tuyến tính ở mẫu số, tạo ra một bề mặt mục tiêu có nhiều cực trị cục bộ (local optima).
- **Tính NP-hard:** Việc tìm nghiệm tối ưu toàn cục cho bài toán tối ưu công suất trong mạng giao thoa đa người dùng đã được chứng minh là NP-hard. Các phương pháp giải tích như đạo hàm (Gradient Descent) dễ bị kẹt, trong khi phương pháp tối ưu toàn cục (Exhaustive Search) có độ phức tạp lũy thừa $O(K^M)$.

Chính vì độ phức tạp này, việc áp dụng các thuật toán Metaheuristic như WOA là hướng đi khả thi và hiệu quả nhất về mặt thời gian.

2.3 Giải pháp Xử lý Ràng buộc: Phương pháp Phạt (Penalty Method)

WOA là thuật toán tối ưu không ràng buộc (Unconstrained Optimizer). Để giải quyết các ràng buộc C1 và C2, bài báo đề xuất tích hợp **Phương pháp Phạt**.

2.3.1 Cơ chế hoạt động

Thay vì loại bỏ các nghiệm vi phạm ràng buộc (điều này làm mất thông tin hữu ích), ta "trừng phạt" chúng bằng cách cộng thêm một giá trị lớn vào hàm mục tiêu (đối với bài toán tối thiểu hóa). Hàm mục tiêu gốc $f(\mathbf{p}) = -\text{GEE}(\mathbf{p})$ được chuyển đổi thành hàm thích nghi mở rộng $\Phi(\mathbf{p})$:

$$\Phi(\mathbf{p}) = f(\mathbf{p}) + \sum_{k=1}^{K} \mu_k \cdot \Psi(g_k(\mathbf{p})) \tag{14}$$

Trong đó:

- μ_k: Hệ số phạt (Penalty Factor). Giá trị này cần đủ lớn (trong bài báo chọn 10^{14}) để đảm bảo nghiệm tối ưu cuối cùng luôn nằm trong vùng khả thi (Feasible Region).
- $\Psi(g_k(\mathbf{p}))$: Hàm đo mức độ vi phạm.

2.3.2 Hàm Thích nghi Cụ thể cho Bài toán GEE

Áp dụng vào bài toán EE-SE, hàm thích nghi cần tối thiểu hóa được thiết lập như sau:

$$F(\mathbf{p}) = \underbrace{-\frac{\sum R_i}{P_{total}}}_{\text{Mục tiêu gốc}} + \underbrace{\mu \sum_{i=1}^{M} \left[\max\left(0, R_i^{req} - R_i(\mathbf{p})\right)\right]^2}_{\text{Thành phần phạt QoS}} \tag{15}$$

- Nếu $R_i \geq R_i^{req}$ (Thỏa mãn): Số hạng phạt bằng 0.
- Nếu $R_i < R_i^{req}$ (Vi phạm): Số hạng phạt bằng $\mu \cdot (\text{Sai lệch})^2$. Việc sử dụng bình phương sai lệch giúp hàm phạt trơn hơn, hỗ trợ thuật toán hội tụ mượt mà hơn so với phạt tuyến tính.

3 Thực thi Biến thể EWOA và Phân tích Kết quả Thực nghiệm

Dựa trên khung lý thuyết vững chắc ở Phần 2, Project này thực hiện mô phỏng máy tính để giải quyết bài toán tối ưu hóa GEE. Chúng tôi đề xuất sử dụng biến thể **EWOA** và so sánh hiệu năng trực tiếp với **WOA Gốc** để kiểm chứng các cải tiến.

3.1 Chiến lược Áp dụng EWOA vào Bài toán Công suất

Việc áp dụng một thuật toán bầy đàn (Swarm Intelligence) vào bài toán kỹ thuật đòi hỏi một chiến lược "ánh xạ" (mapping) chính xác các thành phần.

3.1.1 Mã hóa Cá thể (Encoding Strategy)

Mỗi cá thể "cá voi" trong quần thể đại diện cho một phương án phân bổ công suất tiềm năng cho toàn mạng.

- **Không gian tìm kiếm:** Là không gian M chiều liên tục, giới hạn bởi $[0, P_{max}]^M$.
- **Vector vị trí:** $\vec{X} = [p_1, p_2, \ldots, p_M]$. Tại mỗi vòng lặp, giá trị p_i được cập nhật và phải được kiểm tra biên (Boundary Check) để đảm bảo không âm và không vượt quá P_{max}.

3.1.2 Cải tiến Cốt lõi: Quy luật a Phi tuyến

Chúng tôi thay thế cơ chế giảm tuyến tính của WOA gốc bằng quy luật giảm phi tuyến Parabolic.

- **WOA Gốc:** $a(t) = 2 - \frac{2t}{T}$. Tốc độ giảm là hằng số.
- **EWOA:** $a(t) = 2\left(1 - \left(\frac{t}{T}\right)^2\right)$. Tốc độ giảm thay đổi theo thời gian.

Tác động hành vi: Hàm Parabolic giữ giá trị $a > 1$ trong khoảng 70% thời gian đầu của quá trình lặp (so với 50% của tuyến tính). Điều này đồng nghĩa với việc vector hệ số $|A|$ sẽ lớn hơn 1 trong thời gian dài hơn, ép buộc thuật toán thực hiện cơ chế **Tìm kiếm ngẫu nhiên (Search for Prey)** nhiều hơn. Diều này cực kỳ quan trọng đối với bài toán GEE đa cực trị, giúp quần thể thoát khỏi các vùng tối ưu cục bộ kém chất lượng trước khi hội tụ.

3.2 Thiết lập Môi trường Thực nghiệm

Mô phỏng được thực hiện trên ngôn ngữ Python với các thông số cấu hình bám sát tài liệu tham khảo để đảm bảo tính thực tế.

3.2.1 Thông số Hệ thống (System Parameters)

- **Số lượng người dùng (M):** 4 cặp liên kết (để dễ dàng quan sát hành vi hội tụ chi tiết).
- **Mô hình kênh (G_{ij}):** Phân phối Rayleigh Fading. G_{ii} (kênh lợi ích) tuân theo phân phối mũ với trung bình 1.0; G_{ij} (kênh nhiễu) tuân theo phân phối mũ với trung bình 0.1.
- **Thông số năng lượng:** $P_c = 0.1$ W, $\xi = 1.0$.
- **Nhiễu nền:** $n_0 = \sigma^2 = 10^{-2}$ W.
- **Ràng buộc:** $P_{max} = 2.0$ W, $R_{min} = 0.1$ bps/Hz.

3.2.2 Thông số Thuật toán (Algorithm Settings)

- **Kích thước quần thể (N):** 30 cá thể.
- **Số vòng lặp tối đa (T):** 100 vòng.
- **Hệ số phạt (μ):** 10,000 (Đủ lớn để đảm bảo tính khả thi).
- **Số lần chạy độc lập:** 10 lần (để lấy giá trị trung bình, loại bỏ yếu tố ngẫu nhiên).

3.3 Kết quả Thực nghiệm và So sánh Định lượng

Kết quả trung bình sau các lần chạy thử nghiệm được tổng hợp trong bảng dưới đây:

Bảng 1: Bảng so sánh hiệu năng tối ưu hóa GEE giữa WOA và EWOA

Tiêu chí Đánh giá	WOA Gốc	EWOA (Đề xuất)	Mức Cải thiện
Best GEE (bits/J/Hz)	8.0422	**8.0444**	+0.03%
Min Rate đạt được	0.0998	0.0997	Tương đương (Sát biên)
Thời gian chạy (s)	0.2012	**0.1451**	**Nhanh hơn 27.8%**

3.4 Phân tích Chuyên sâu Kết quả

3.4.1 Về Hiệu quả Năng lượng (Chất lượng Nghiệm)

EWOA đạt được mức GEE là **8.0444**, nhỉnh hơn so với WOA Gốc (8.0422). Mặc dù mức tăng về giá trị tuyệt đối không quá lớn, nhưng trong các hệ thống truyền thông quy mô lớn, việc cải thiện dù chỉ một phần nhỏ hiệu suất năng lượng cũng mang lại lợi ích kinh tế đáng kể.

- Sự cải thiện này đến từ khả năng **Khám phá (Exploration)** tốt hơn của EWOA. Trong không gian tìm kiếm hỗn loạn của bài toán nhiễu giao thoa, WOA gốc dễ bị hút vào các "hố" cục bộ (nơi công suất phát cao nhưng GEE thấp). EWOA, nhờ giữ tham số a lớn lâu hơn, đã "nhảy" qua các vùng này để tìm được tổ hợp công suất tiết kiệm năng lượng hơn.

3.4.2 Về Tính Khả thi của Nghiệm (Constraint Satisfaction)

Cả hai thuật toán đều trả về mức tốc độ tối thiểu xấp xỉ **0.0997 - 0.0998**, rất sát với yêu cầu $R_{min} = 0.1$.

- Đây là hiện tượng **Binding Constraint (Ràng buộc chặt)**. Để tối đa hóa GEE (tử số/mẫu số), cách tốt nhất là giảm công suất phát (mẫu số) xuống mức thấp nhất có thể, tức là mức "vừa đủ" để đạt QoS.
- Việc thuật toán tự động hội tụ về đúng biên giới hạn này (0.1) chứng tỏ **Phương pháp Phạt** đã hoạt động cực kỳ hiệu quả, ép buộc bầy cá voi phải bơi men theo bờ vực của vùng khả thi để tìm kiếm thức ăn (nghiệm tối ưu).

3.4.3 Về Tốc độ Thực thi (Computational Efficiency)

Đây là điểm sáng nhất của biến thể EWOA: **Tốc độ nhanh hơn gần 28

- **Nguyên nhân:** Trong bài toán phạt, vùng không khả thi (Infeasible Region) có giá trị hàm mục tiêu rất lớn và độ dốc (gradient) thay đổi gắt. Việc tính toán và cập nhật vị trí trong vùng này thường tốn kém tài nguyên xử lý hơn.
- EWOA với bước nhảy lớn (nhờ a phi tuyến) có khả năng thoát khỏi vùng phạt nhanh hơn, đưa quần thể về vùng khả thi sớm hơn. Khi đã ở trong vùng khả thi, các phép toán trở nên đơn giản hơn, dẫn đến tổng thời gian chạy giảm xuống.

3.5 Kết luận

Thông qua việc phân tích lý thuyết và thực nghiệm, Project rút ra các kết luận sau:

1. **WOA là công cụ mạnh mẽ:** Nó giải quyết tốt bài toán phân bổ tài nguyên phi lồi, đa biến mà không cần các phép toán đạo hàm phức tạp.
2. **EWOA ưu việt về tốc độ:** Biến thể cải tiến tham số a phi tuyến đã chứng minh được sự vượt trội về thời gian tính toán, là yếu tố then chốt cho các ứng dụng thời gian thực (Real-time) trong mạng 6G.
3. **Phương pháp Phạt hiệu quả:** Việc kết hợp hàm phạt giúp WOA xử lý triệt để các ràng buộc QoS khắt khe, đảm bảo nghiệm tìm được luôn có ý nghĩa thực tế.

A Phụ lục: Ý nghĩa Tham số và Tác động Kỹ thuật

Phần này tổng hợp ý nghĩa vật lý và toán học của các tham số quan trọng được sử dụng trong báo cáo và mã nguồn mô phỏng. Đây là cơ sở để hiệu chỉnh thuật toán và giải thích kết quả.

A.1 Tham số của Thuật toán (WOA/EWOA)

Bảng 2: Giải thích các tham số điều khiển thuật toán

Tham số	**Tên gọi / Định nghĩa**	**Tác dụng và Ảnh hưởng**
N	Kích thước quần thể (Population Size)	**Tác dụng:** Số lượng "cá voi" (nghiệm) tìm kiếm đồng thời. **Ảnh hưởng:** N lớn giúp tìm kiếm kỹ hơn nhưng làm chậm chương trình. N quá nhỏ dễ dẫn đến thiếu đa dạng và kẹt cục bộ. (Project chọn $N = 30$).
T (MaxIter)	Số vòng lặp tối đa	**Tác dụng:** Điều kiện dừng của thuật toán. **Ảnh hưởng:** Cần đủ lớn để thuật toán hội tụ. Trong bài toán này, $T = 100$ là đủ để EWOA tìm ra vùng nghiệm tốt.
a	Hệ số hội tụ (Convergence Factor)	**Tác dụng:** Kiểm soát sự chuyển đổi giữa Khám phá và Khai thác. Giảm từ 2 về 0. **Lưu ý:** Trong EWOA, a giảm theo hàm Parabolic để giữ giá trị cao lâu hơn, giúp tăng cường Khám phá đầu trận.
$\vec{A}$	Vector hệ số bước nhảy (Coefficient Vector)	**Tác dụng:** Quyết định loại hành vi. **Quy tắc:** Nếu $\|A\| \geq 1$: Tìm kiếm ngẫu nhiên (Khám phá). Nếu $\|A\| < 1$: Bao vây con mồi (Khai thác).
$\vec{C}$	Hệ số ngẫu nhiên (Arbitrary Factor)	**Tác dụng:** $C = 2 \cdot r$. Tạo trọng số ngẫu nhiên cho vị trí con mồi. **Ý nghĩa:** Giúp cá voi không bị hút về đích một cách quá trơn tru, tạo ra các dao động nhỏ để tránh bẫy cục bộ ngay cả trong pha khai thác.
p	Xác suất hành vi	**Tác dụng:** Số ngẫu nhiên $[0, 1]$. **Quy tắc:** Nếu $p < 0.5$: Bao vây/Tìm kiếm. Nếu $p \geq 0.5$: Tấn công xoắn ốc.
b	Hằng số xoắn ốc	**Tác dụng:** Định hình độ mở của đường xoắn ốc Logarithmic. Thường chọn $b = 1$.

A.2 Tham số của Hệ thống và Bài toán (System Model)

Bảng 3: Giải thích các tham số vật lý của mạng vô tuyến

Tham số	**Ý nghĩa Vật lý**	**Vai trò trong bài toán tối ưu**
M	Số cặp người dùng (User Links)	Quyết định kích thước bài toán (Số biến cần tối ưu = M). M càng lớn, nhiễu giao thoa càng phức tạp.
G_{ij}	Độ lợi kênh truyền (Channel Gain)	G_{ii}: Kênh tín hiệu (cần lớn). $G_{ij}(j \neq i)$: Kênh gây nhiễu (cần nhỏ). Đây là dữ liệu đầu vào cố định.
P_{max}	Công suất phát tối đa	**Ràng buộc không gian tìm kiếm.** Mọi nghiệm p_i phải nằm trong $[0, P_{max}]$.
R_{min}	Tốc độ tối thiểu (QoS Requirement)	**Ràng buộc khó.** Nếu $R_i < R_{min}$, nghiệm bị coi là vi phạm và bị phạt nặng.
P_c	Công suất mạch tĩnh	Công suất tiêu thụ nền của thiết bị (kể cả khi không phát sóng). Làm cho hàm mục tiêu GEE trở nên phi tuyến tính.
μ (hoặc λ)	Hệ số phạt (Penalty Factor)	**Quan trọng:** Giá trị rất lớn (10^{14}). Dùng để biến giá trị Fitness thành cực lớn (rất tệ) nếu vi phạm ràng buộc, ép thuật toán phải tìm về vùng khả thi.