TẠP CHÍ KHOA HỌC ĐẠI HỌC ĐỒNG THÁP
Dong Thap University Journal of Science
Chuyên san Khoa học Tự nhiên
ISSN 0866-7675 | e-ISSN 2815-567X

DOI: https://doi.org/10.52714/dthu.14.8.2025.1551

NÂNG CAO CHẤT LƯỢNG GIẢI PHÁP CỦA THUẬT TOÁN LAI SÓI XÁM VÀ DI TRUYỀN THÔNG QUA TÌM KIẾM CỤC BỘ LỐC XOÁY

Phạm Nguyễn[1], Nguyễn Thị Thu Hiền và Đinh Nguyễn Trọng Nghĩa[*]

Khoa Công nghệ thông tin,
Trường Đại học Công Thương Thành phố Hồ Chí Minh, Việt Nam

[*]Tác giả liên hệ, Email: nghiadnt@huit.edu.vn

Lịch sử bài báo

Ngày nhận: 21/11/2024; Ngày nhận chỉnh sửa: 07/3/2025; Ngày duyệt đăng: 12/3/2025

Tóm tắt

Bài báo này nghiên cứu việc nâng cao hiệu suất của thuật toán Sói xám (GWO) bằng cách kết hợp với thuật toán di truyền (GA) và tích hợp thêm bước tìm kiếm cục bộ lốc xoáy nhằm cải thiện khả năng tối ưu hóa. GWO là một phương pháp dựa trên hành vi săn mồi của Sói xám, trong khi GA là thuật toán tiến hóa mô phỏng quá trình chọn lọc tự nhiên. Việc kết hợp này tận dụng khả năng khai thác của GWO, khả năng khám phá của GA và sự tinh chỉnh bằng phương pháp tìm kiếm cục bộ lốc xoáy. Thực hiện các thí nghiệm trên các hàm kiểm tra chuẩn cho thấy phương pháp GWO-GA cải thiện đáng kể độ chính xác và tốc độ hội tụ so với các phương pháp truyền thống. Phương pháp này mở ra một cách tiếp cận mới trong lĩnh vực tối ưu hóa, với tiềm năng ứng dụng rộng rãi trong nhiều lĩnh vực khác nhau. Các hướng phát triển tương lai bao gồm tối ưu hóa tham số thuật toán và thử nghiệm trên các bài toán tối ưu hóa thực tế phức tạp hơn.

Từ khóa: *GA, GWO, hiệu suất tối ưu, tìm kiếm cục bộ lốc xoáy, tối ưu hóa kết hợp.*

Trích dẫn: Phạm, N., Nguyễn, T. T. H., & Đinh, N. T. N. (2025). Nâng cao chất lượng giải pháp của thuật toán lai sói xám và di truyền thông qua tìm kiếm cục bộ lốc xoáy. *Tạp chí Khoa học Đại học Đồng Tháp, 14*(8), 50-66. https://doi.org/10.52714/dthu.14.8.2025.1551

IMPROVING THE QUALITY OF THE GREY WOLF ALGORITHM AND GENETIC ALGORITHM THROUGH TORNADO LOCAL SEARCH

Pham Nguyen, Nguyen Thi Thu Hien, and Dinh Nguyen Trong Nghia*

Department of Information Technology,
Ho Chi Minh City University of Industry and Trade, VietNam

**Corresponding author, Email: nghiadnt@huit.edu.vn*

Article history

Received: 21/11/2024; Received in revised form: 07/3/2025; Accepted: 12/3/2025

Abstract

This paper investigates the enhancement of the Grey Wolf Optimizer (GWO) by integrating it with the Genetic Algorithm (GA) and an additional tornado local search step to improve optimization performance. GWO is inspired by the hunting behavior of grey wolves, while GA is an evolutionary algorithm that simulates the process of natural selection. This hybrid approach leverages the exploitation capability of GWO, the exploration strength of GA, and the refinement potential of local search. Experiments on standard benchmark functions demonstrate that the GWO-GA approach significantly improves both accuracy and convergence speed compared to traditional methods. This novel approach to optimization promises applications across various fields. Future research includes optimizing algorithm parameters and testing on more complex, real-world optimization problems.

Keywords: *Combination optimization*, *genetic algorithm*, *Grey Wolf Optimizer*, *optimal performance*, *tornado local search.*

1. Đặt vấn đề

Trong bối cảnh phát triển nhanh chóng của khoa học và công nghệ, các bài toán tối ưu hóa phức tạp xuất hiện ngày càng nhiều trong các lĩnh vực như kỹ thuật, y học, kinh tế và quản lý. Những bài toán này không chỉ đòi hỏi các giải pháp chính xác mà còn cần phải được tìm kiếm trong thời gian hợp lý, điều này đặt ra thách thức lớn đối với các phương pháp tối ưu hóa truyền thống (Mirjalili, 2014).

Thuật toán metaheuristic nổi bật với khả năng tìm kiếm lời giải tối ưu gần đúng bằng cách mô phỏng các hiện tượng tự nhiên và sử dụng tính ngẫu nhiên, thay vì dựa vào một quy trình xác định. Các thuật toán này thường sử dụng quần thể và quá trình tiến hóa để khám phá không gian lời giải (Tomar & cs., 2023).

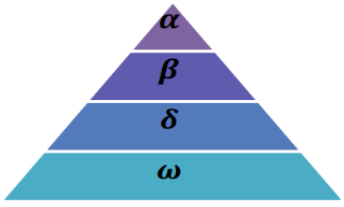
Trong số các thuật toán metaheuristic, thuật toán Sói xám và thuật toán di truyền đã chứng tỏ hiệu quả đáng kể. GWO được biết đến với khả năng giải quyết hiệu quả các bài toán tối ưu đa mục tiêu, trong khi GA với sự linh hoạt và mạnh mẽ đã trở thành một công cụ quan trọng trong việc tìm kiếm và tối ưu hóa lời giải (Abdel-Basset & cs., 2018).

Bài báo cáo này sẽ tập trung vào việc nâng cao hiệu suất và tối ưu hóa thuật toán GWO thông qua sự kết hợp với thuật toán GA và tích hợp thêm bước tìm kiếm cục bộ lốc xoáy (Abd & Oliva 2018). Bằng cách kết hợp hai phương pháp mạnh mẽ này cùng với tìm kiếm cục bộ, kỳ vọng sẽ đạt được hiệu quả tối ưu hóa cao hơn và cải thiện khả năng tìm kiếm lời giải cho các bài toán phức tạp.

2. Cơ sở lý thuyết

2.1. Thuật toán Tối ưu Sói xám (GWO)

Thuật toán GWO được phát triển vào năm 2014 bởi Mirjalili và cộng sự. Lấy cảm hứng từ hành vi xã hội của loài sói, hệ thống phân cấp xã hội của bầy sói xám thường là bốn nhóm được gọi tên theo thứ tự lần lượt từ cao đến thấp Alpha (α), Beta (β), Delta (δ) và Omega (ω) như thể hiện ở hình 1 (Mirjalili, 2015) (Mirjalili & cs., 2014).



Hình 1. Phân cấp xã hội của đàn sói trong tự nhiên

Mỗi con sói sẽ nhận một nhiệm vụ riêng: Alpha (α) là con đầu đàn, đưa ra các quyết định chiến lược về săn mồi và phân chia thức ăn. Beta (β) hỗ trợ Alpha trong việc ra quyết định. Delta (δ) đảm nhận vai trò đào tạo và bảo vệ bầy đàn (Mirjalili & cs., 2016). Omega (ω) là nhóm có vị thế thấp nhất, tuân theo lệnh của các nhóm trên. Sói xám săn mồi theo chiến lược hợp tác, một vài con đuổi theo con mồi trong khi các con khác đảm bảo không để mồi thoát.Chiến lược săn mồi của bầy sói bao gồm:

2.1.1. Bao vây con mồi

Các bước bao vây trong GWO bằng các phương trình sau đây:

$$\overrightarrow{D} = \left|\overrightarrow{C}.\overrightarrow{X_p}(t) - \overrightarrow{X}(t)\right| \quad (1)$$

$$\overrightarrow{X}(t+1) = \overrightarrow{X_p}(t) - \overrightarrow{A}.\overrightarrow{D} \quad (2)$$

Trong đó t cho biết sự lặp lại hiện tại, $\overrightarrow{X_p}$ là vectơ vị trí của con mồi và $\overrightarrow{X}$ biểu thị vectơ vị trí của một con sói. $\overrightarrow{A}$ và $\overrightarrow{C}$ là các vectơ hệ số được tính như sau:

$$\overrightarrow{A} = 2.\overrightarrow{a}.\overrightarrow{r_1} - \overrightarrow{a} \quad (3)$$

$$\overrightarrow{C} = 2.\overrightarrow{r_2} \quad (4)$$

Trong đó $\overrightarrow{a}$ sẽ được giảm tuyến tính từ 2 xuống 0 trong quá trình lặp lại và *r*1, *r*2 là các vectơ ngẫu nhiên trong [0, 1]. Vì vậy, một con sói có thể cập nhật vị trí của nó theo tọa độ (X, Y) trong không gian xung quanh con mồi ở bất kỳ vị trí ngẫu nhiên.

2.1.2. Săn mồi

Sói Xám có khả năng nhận biết và bao vây con mồi trong quá trình săn. Trong tối ưu hóa, Alpha (α) hướng dẫn các con sói khác. Khi không có thông tin rõ ràng về vị trí tối ưu, Beta (β) và Delta (δ) đóng vai trò như các giải pháp ứng cử viên có kiến thức tốt hơn về vị trí tiềm năng của con mồi. Thuật toán GWO lưu trữ ba giải pháp tốt nhất (α, β, δ) và yêu cầu Omega (ω) cập nhật vị trí dựa trên vị trí của các con sói có khả năng tìm kiếm tốt nhất (Fu & cs., 2021). Công thức được đề xuất cho vấn đề này như sau:

$$\overrightarrow{D_\alpha} = \left|\overrightarrow{C_1}.\overrightarrow{X_\alpha} - \overrightarrow{X}\right|, \overrightarrow{D_\beta} = \left|\overrightarrow{C_2}.\overrightarrow{X_\beta} - \overrightarrow{X}\right|, \overrightarrow{D_\delta} = \left|\overrightarrow{C_3}.\overrightarrow{X_\beta} - \overrightarrow{X}\right| \quad (5)$$

$$\overrightarrow{X_1} = \overrightarrow{X_\alpha} - \overrightarrow{A_1}.(\overrightarrow{D_\alpha}), \overrightarrow{X_2} = \overrightarrow{X_\beta} - \overrightarrow{A_2}.(\overrightarrow{D_\beta}), \overrightarrow{X_3} = \overrightarrow{X_\delta} - \overrightarrow{A_3}.(\overrightarrow{D_\delta}) \quad (6)$$

Với các phương trình trên cho thấy alpha(α), beta(β) và delta(δ) sẽ xác định vị trí của con mồi và con sói omega(ω) sẽ cập nhật vị trí của mình xung quanh con mồi theo cách ngẫu nhiên (Zhang & cs., 2017).

2.1.3. Tấn công con mồi (khai thác)

Đàn sói chỉ kết thúc cuộc săn bằng cách tấn công con mồi khi nó ngừng di chuyển. Thuật toán GWO mô phỏng quá trình này bằng cách giảm giá trị $\overrightarrow{a}$ từ 2 xuống 0 . Biên độ dao động của $\overrightarrow{A}$ cũng giảm theo $\overrightarrow{a}$ với $\overrightarrow{A}$ là một giá trị ngẫu nhiên trong khoảng [-2a,2a]. Khi các giá trị ngẫu nhiên của $\overrightarrow{A}$ nằm trong [-1,1], vị trí tiếp theo của con sói tìm kiếm có thể ở giữa vị trí hiện tại của nó và vị trí của con mồi.

2.1.4. Tìm kiếm con mồi (thăm dò)

Quá trình tìm kiếm bắt đầu khi các sói alpha(α), beta(β) và delta(δ) ước tính vị trí của con mồi và cập nhật khoảng cách. Các con sói tách ra để tìm kiếm và hội tụ để tấn công. Để tạo sự đa dạng, giá trị $\overrightarrow{A}$ ngẫu nhiên lớn hơn 1 hoặc nhỏ hơn -1 được sử dụng để bắt buộc các con sói phải chuyển hướng khỏi con mồi. Tham số $\overrightarrow{a}$ được giảm dần để thăm dò và khai thác. Khi $\overrightarrow{|A|} > 1$ sói di chuyển xa con mồi, khi $\overrightarrow{|A|} < 1$ sói di chuyển gần con mồi. Thuật toán sẽ kết thúc khi thỏa mãn một tiêu chí cuối cùng (Panda & Das, 2019).

2.2. Thuật giải di truyền (GA)

GA là một kỹ thuật tối ưu hóa dựa trên di truyền học và sinh học tiến hóa, tương tự với cấu trúc di truyền và hành vi của nhiễm sắc thể trong quần thể lấy cảm hứng từ học thuyết Darwin (Tuhus & Krarti, 2010). GA mô phỏng quá trình thích nghi của quần thể sinh học để tìm giải pháp tối ưu cho các vấn đề phức tạp. Sau đây là quá trình thực thi của GA dựa trên các nguyên lý này:

- Các cá thể trong quần thể tranh giành tài nguyên. Khi có những cá thể khỏe mạnh nhất sẽ giao phối để sinh ra nhiều cá thể con hơn những cá thể không khỏe mạnh.

- Các gen từ bố mẹ "khỏe nhất" được truyền qua nhiều thế hệ, nghĩa là đôi khi bố mẹ

tạo ra con cái tốt hơn bố hoặc mẹ sẽ phù hợp hơn với môi trường.

Sau khi đã hình thành ý tưởng về cách thuật toán này mô phỏng sự tiến hóa tự nhiên, các bước trong quy trình tối ưu hóa của GA bao gồm (Mondal & cs., 2023):

2.2.1. Khởi tạo

Bước đầu tiên trong thuật toán di truyền là khởi tạo quần thể ban đầu, mỗi cá thể được tạo ra ngẫu nhiên hoặc bằng các phương pháp heuristic. Mỗi thiết kế trong quần thể được đại diện bởi một nhiễm sắc thể. Công thức biểu diễn quần thể khởi tạo là:

$$P(0) = \{x_1, x_2, \dots, x_N\} \quad (7)$$

Trong đó, $P(0)$ là quần thể ban đầu và x_i là các cá thể trong quần thể với N là kích thước quần thể (Yao & Xu, 2024).

2.2.2. Đánh giá hàm mục tiêu và mức độ phù hợp

Sau khi khởi tạo quần thể, hàm mục tiêu được sử dụng để đánh giá mức độ thích nghi của từng cá thể trong quần thể.

2.2.3. Lựa chọn

Ý tưởng là ưu tiên những cá thể có điểm thể lực tốt và có thể truyền gen của mình cho các thế hệ kế tiếp. Trong đó xác suất chọn một cá thể nhất định tỷ lệ thuận với fitness của nó.

2.2.4. Trao đổi chéo

Quá trình trao đổi chéo thực hiện sự giao phối giữa các cá thể. Hai cá thể được chọn bằng cách sử dụng toán tử lựa chọn và các vị trí trao đổi chéo được chọn ngẫu nhiên. Các gen tại các vị trí trao đổi chéo này được trao đổi để tạo ra các cá thể mới Rafaely & Bennell, 2006).

2.2.5. Đột biến

Quá trình đột biến thực hiện các thay đổi ngẫu nhiên nhỏ để tăng sự đa dạng cho quần thể và ngăn chặn sự hội tụ sớm vào các giải pháp kém.

2.2.6. Thay thế

Quá trình thay thế diễn ra bằng cách thay thế thế hệ cũ bằng thế hệ mới, bao gồm các thiết kế ban đầu, các thế hệ con lai và các thiết kế đã trải qua đột biến. Công thức cho bước này là:

$$P(t+1) = \{M_1, M_2, \dots, M_N\} \quad (8)$$

Trong đó, $P(t+1)$ là quần thể mới sau khi thay thế.

2.2.7. Tiêu chí kết thúc

Quá trình lặp lại từ bước lựa chọn đến thay thế cho đến khi đạt điều kiện kết thúc, chẳng hạn như số thế hệ tối đa hoặc khi không còn thấy sự cải thiện đáng kể nhằm tìm ra giải pháp tối ưu hoặc gần tối ưu cho bài toán (Shial & cs., 2023).

3. Phương pháp kết hợp thuật toán GWO-GA tích hợp tìm kiếm cục bộ lốc xoáy

3.1. Lý do lựa chọn kết hợp

GWO và GA là hai thuật toán tối ưu hóa metaheuristic phổ biến, mỗi thuật toán có những ưu điểm và nhược điểm riêng. Việc kết hợp chúng nhằm mục đích khai thác hiệu quả các ưu điểm và giảm thiểu các nhược điểm của từng thuật toán. GWO nổi bật với khả năng khai thác chi tiết không gian tìm kiếm, mô phỏng hành vi săn mồi của bầy sói xám, giúp duy

trì sự cân bằng giữa việc khám phá và khai thác không gian tìm kiếm. Tuy nhiên, GWO dễ mắc kẹt tại các cực trị cục bộ do thiếu khả năng khám phá rộng. GA mô phỏng quá trình chọn lọc tự nhiên, nổi bật với khả năng khám phá không gian tìm kiếm rộng lớn thông qua các quá trình lai ghép và đột biến, giúp duy trì đa dạng di truyền và tăng khả năng thoát khỏi các cực trị cục bộ. Tuy nhiên, GA có khả năng khai thác chi tiết không cao và quá trình hội tụ có thể chậm hơn.

Việc kết hợp GWO và GA tận dụng điểm mạnh của cả hai phương pháp, tăng khả năng khám phá và khai thác, tránh mắc kẹt tại các cực trị cục bộ và nâng cao hiệu suất tối ưu hóa. GWO đảm nhận vai trò khai thác chi tiết các vùng tìm kiếm tiềm năng, trong khi GA mở rộng khả năng khám phá, giúp tìm kiếm các giải pháp tối ưu toàn cục hiệu quả hơn. Tuy nhiên, để khai thác tối đa tiềm năng của GWO và GA, phương pháp kết hợp này còn tích hợp thêm kỹ thuật tìm kiếm cục bộ theo quỹ đạo lốc xoáy nhằm tăng cường khả năng tìm kiếm chi tiết trong các vùng cục bộ. Cách tiếp cận này giúp nâng cao khả năng thoát khỏi cực trị cục bộ và cải thiện hiệu suất tối ưu hóa tổng thể.

3.2. Quá trình kết hợp GWO và GA tích hợp tìm kiếm cục bộ

3.2.1. Quá trình kết hợp

Quá trình tối ưu hóa bắt đầu bằng việc tạo ra một quần thể ngẫu nhiên, trong đó các cá thể được phân bố trong không gian tìm kiếm D-chiều với các giới hạn xác định. Độ thích nghi của mỗi cá thể sẽ được đánh giá dựa trên hàm mục tiêu. Sau khi đánh giá, vị trí các cá thể được cập nhật theo cơ chế săn mồi của thuật toán GWO. Trong GWO, các cá thể dẫn đầu alpha, beta và delta đóng vai trò điều hướng và tấn công mục tiêu, mô phỏng hành vi bao vây con mồi từ nhiều hướng khác nhau.

3.2.2. Tìm kiếm cục bộ lốc xoáy

Sau khi vị trí các cá thể được cập nhật, bước tìm kiếm cục bộ theo quỹ đạo xoắn ốc được triển khai để tăng cường chất lượng của từng cá thể. Phương pháp này tìm kiếm sâu hơn trong không gian lân cận, đặc biệt hiệu quả trong việc khám phá các điểm lân cận đa hướng, giúp giảm nguy cơ mắc kẹt tại các cực trị cục bộ.

Trong tìm kiếm cục bộ lốc xoáy, mỗi cá thể sẽ di chuyển quanh vị trí hiện tại theo một quỹ đạo xoắn ốc, với bán kính dịch chuyển thu hẹp dần qua mỗi bước, tạo điều kiện mở rộng phạm vi tìm kiếm trong không gian. Công thức mô tả quỹ đạo xoắn ốc là:

$$x_{new} = x_{current} + r \cdot \begin{bmatrix} \cos(0) \\ \sin(0) \end{bmatrix} \quad (9)$$

Trong đó, x_{new} là vị trí mới của cá thể, $x_{current}$ là vị trí hiện tại. Bán kính r của quỹ đạo xoắn ốc và 0 là góc xoay ngẫu nhiên trong khoảng từ 0 đến 2π Bán kính r giảm dần qua các bước, được tính toán dựa trên công thức:

$$r = r_{init} \cdot (shrink\_factor)^{\text{step}} \quad (10)$$

Trong đó r_{init} là bán kính ban đầu và $shrink\_factor$ là hệ số điều chỉnh tốc độ giảm dần của bán kính. Góc xoay ngẫu nhiên 0 trong từng bước di chuyển giúp tăng tính linh hoạt của quá trình tìm kiếm, đảm bảo việc tiếp cận không gian lân cận một cách toàn diện.

Quá trình này cho phép các cá thể tiếp cận không gian lân cận một cách toàn diện, từ đó tăng khả năng khai thác tối đa xung quanh vị trí hiện tại, đồng thời linh hoạt trong việc thăm dò (exploration) theo nhiều hướng khác nhau. Việc giảm dần bán kính và tạo góc ngẫu nhiên giúp cá thể duy trì sự linh hoạt trong quá trình tìm kiếm, từ đó tăng hiệu quả đạt giải pháp tối ưu và tránh tình trạng bị kẹt trong các cục bộ.

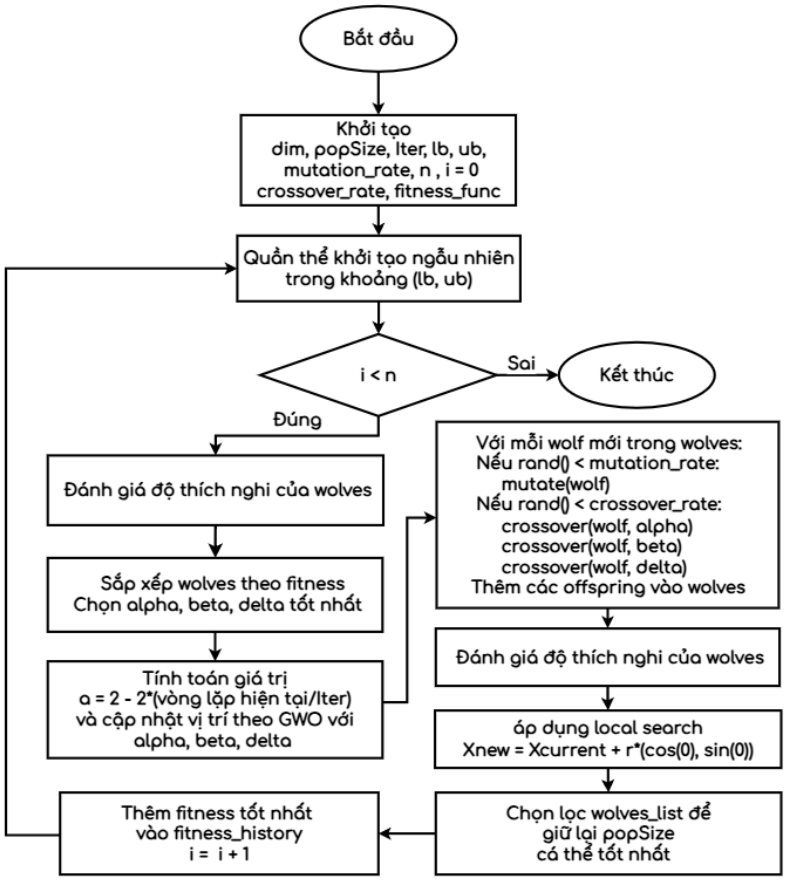
3.2.3. Áp dụng thuật toán GA

Sau quá trình tìm kiếm cục bộ, thuật toán di truyền được áp dụng để duy trì sự đa dạng trong quần thể và đảm bảo khả năng khám phá toàn cục. Quá trình đột biến được thực hiện với một tỷ lệ xác định trước, giúp tạo ra các cá thể mới với biến thể khác biệt, nhờ đó tăng khả năng khám phá không gian tìm kiếm rộng hơn (Emary & cs., 2016). Các cá thể con sau đột biến sẽ được lai ghép với các cá thể alpha, beta và delta trong GWO nhằm kết hợp những đặc điểm tốt nhất từ các cá thể ưu tú. Việc lai ghép này giúp cải thiện chất lượng của quần thể, nâng cao khả năng tìm ra giải pháp tối ưu.

3.2.4. Quá trình chọn lọc và duy trì quần thể

Sau khi thực hiện đột biến và lai ghép, các cá thể trong quần thể sẽ trải qua quá trình chọn lọc. Chỉ những cá thể có độ thích nghi cao nhất sẽ được giữ lại, giúp đảm bảo rằng quần thể tiếp tục phát triển theo hướng tìm kiếm giải pháp tốt hơn. Số lượng cá thể trong quần thể được duy trì ổn định theo quy mô ban đầu, đồng thời các bước từ đánh giá, cập nhật, tìm kiếm cục bộ và GA sẽ được lặp lại liên tục cho đến khi đạt đến điều kiện dừng của thuật toán (ví dụ: số vòng lặp tối đa hoặc độ chính xác tối ưu).

3.2.5. Sơ đồ khối



Hình 2. Sơ đồ mô tả quá trình thực thi của thuật toán kết hợp

Thuật toán bắt đầu với các tham số đầu vào như số chiều bài toán (dim), kích thước quần thể (popSize), số vòng lặp (Iter), giới hạn miền tìm kiếm (lb, ub), tỷ lệ đột biến

(mutation_rate), tỷ lệ lai ghép (crossover_rate) và hàm độ thích nghi (fitness_func). Quần thể ban đầu được tạo ngẫu nhiên trong miền giới hạn để đảm bảo tính đa dạng.

Mỗi vòng lặp, thuật toán đánh giá độ thích nghi của quần thể, sắp xếp và chọn ba cá thể tốt nhất (alpha, beta, delta) để dẫn dắt quá trình tìm kiếm theo cơ chế GWO. Sau đó, các toán tử đột biến và lai ghép của GA được áp dụng nhằm tăng cường khai thác không gian tìm kiếm. Ngoài ra, thuật toán tích hợp bước tìm kiếm cục bộ (local search) để tinh chỉnh nghiệm, cải thiện độ chính xác.

Cuối cùng, thuật toán chọn lọc các cá thể có độ thích nghi cao nhất để duy trì kích thước quần thể, tiếp tục lặp lại cho đến khi đạt số vòng lặp tối đa. Kết quả thu được là lịch sử tiến hóa của các cá thể tối ưu theo từng vòng lặp. Hình trên minh họa sơ đồ khối của quá trình này. Quá trình lặp lại đến khi đạt số vòng lặp tối đa (n), kết quả là lịch sử độ thích nghi của các cá thể tốt nhất. Hình 2 minh họa sơ đồ khối của quá trình này.

3.2.6. Mã giả

```
Khởi tạo(dim, popSize, Iter, lb, ub, mutation_rate,
crossover_rate, fitness_func):
  Hàm initialize_population():
    Tạo quần thể ngẫu nhiên trong khoảng (lb, ub)
    Trả về quần thể ban đầu
  Hàm calculate_fitness(wolves):
    Đánh giá độ thích nghi của mỗi cá thể trong wolves
    Trả về danh sách độ thích nghi
  Hàm mutate(wolf):
    Nếu rand() < mutation_rate:
      Thực hiện đột biến gene cho wolf
    Trả về wolf đã đột biến
  Hàm crossover(wolf1, wolf2):
    Tạo offspring bằng cách lai ghép wolf1 và wolf2
    Trả về offspring
  Hàm local_search(wolf):
    Tạo biến ngẫu nhiên r để điều chỉnh vị trí
    Cập nhật vị trí mới của wolf:
      Xnew = Xcurrent + r * [cos(θ), sin(θ)]
    Trả về wolf với vị trí đã được điều chỉnh
  Hàm optimize():
    wolves = initialize_population()
    fitness_history = []
    Cho mỗi vòng lặp từ 1 đến Iter:
      fitness = calculate_fitness(wolves)
      Sắp xếp wolves theo fitness
      Chọn alpha, beta, delta là ba cá thể có fitness tốt nhất
      a = 2 - 2 * (vòng lặp hiện tại / Iter)
      wolves_list = []
      Cho mỗi wolf trong wolves:
        Cập nhật vị trí của wolf theo chiến lược của GWO:
```

```
          wolf.position = vị trí mới theo alpha, beta, delta
và a
        Nếu rand() < mutation_rate:
          wolf = mutate(wolf)
        Nếu rand() < crossover_rate:
          Chọn ngẫu nhiên một trong ba cá thể alpha, beta,
delta để lai
          wolf = crossover(wolf, cá thể đã chọn)
        Thực hiện tìm kiếm cục bộ:
          wolf = local_search(wolf)
        Thêm wolf vào wolves_list
      Chọn lọc wolves_list để giữ lại popSize cá thể tốt nhất
      Cập nhật quần thể wolves = wolves_list
      Thêm độ thích nghi tốt nhất vào fitness_history
    Trả về fitness_history
```

4. Thực nghiệm và đánh giá trên các hàm thử nghiệm

Để đánh giá hiệu quả của thuật toán GWO-GA, nhóm thực hiện thực nghiệm trên một loạt các hàm thử nghiệm (Jamil & Yang, 2013) phổ biến bao gồm F1, F2, F3, F4, F5, F6, F7, F8 và F9. Các hàm này được lựa chọn nhằm kiểm tra khả năng của thuật toán trong việc tìm kiếm cực trị toàn cục trong các không gian tìm kiếm phức tạp và đa dạng.

Dưới đây là bảng mô tả chi tiết các hàm thử nghiệm bao gồm công thức, phạm vi biến và số lượng biến sử dụng trong các thí nghiệm. Tất cả các kết quả kiểm chứng trong bài báo này đều được thực hiện trên máy tính có cấu hình AMD Ryzen 5 3550H, 16GB RAM và tốc độ xử lý 2.10 GHz, sử dụng phiên bản Python 3.11.2. Các thử nghiệm được tiến hành trong môi trường này nhằm đảm bảo tính nhất quán và độ tin cậy của các kết quả đạt được từ phương pháp GWO-GA.

Bảng 1. Công thức toán học và thông số của các hàm thử nghiệm được sử dụng trong thực nghiệm

Hàm	Công thức	Phạm vi	Số chiều	Mục tiêu
F1	$\sum x_i^2$	$[-2,2]$	10,30,50	0
F2	$\sum \lvert x_i \rvert + \prod \lvert x_i \rvert$	$[-10,10]$	10,30,50	0
F3	$\sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2$	$[-5.12,5.12]$	10,30,50	0
F4	$max\lvert x_i \rvert$	$[-30,30]$	10,30,50	0

Hàm	Công thức	Phạm vi	Số chiều	Mục tiêu
F5	$\sum_{i=1}^{n-1}\left[100\left(x_{i+1}-x_i^2\right)^2+(x_i-1)^2\right]$	$[-30,30]$	10,30,50	0
F6	$-20exp-0.2\sqrt{\frac{1}{n}\sum x_i^2}-exp\frac{1}{n}\sum cos2\pi x_i+20+exp1$	$[-32,32]$	10,30,50	0
F7	$\frac{\pi}{n}\left[10sin^2\pi\left(1+\frac{x_1+1}{4}\right)+\sum_{i=1}^{n-1}\left(\frac{x_i+1}{4}\right)^2\left(1+10sin^2\pi\left(1+\frac{x_{i+1}+1}{4}\right)\right)+\left(\frac{x_n+1}{4}\right)^2\right]+\sum U(x,10,100,4)$	$[-50,50]$	10,30,50	0
F8	$0.1\left[sin^2(3\pi x_1)+\sum_{i=1}^{n-1}(x_i-1)^2\left(1+sin^2(3\pi x_{i+1})\right)+(x_n-1)^2(1+sin^2(2\pi x_n))\right]+\sum U(x,5,100,4)$	$[-50,50]$	10,30,50	0
F9	$\sum_{i=1}^{n}\left(x_i^2-10\cos(2\pi x_i)\right)+10n$	$[-5.12,5.12]$	10,30,50	0

4.1. Thiết lập thực nghiệm

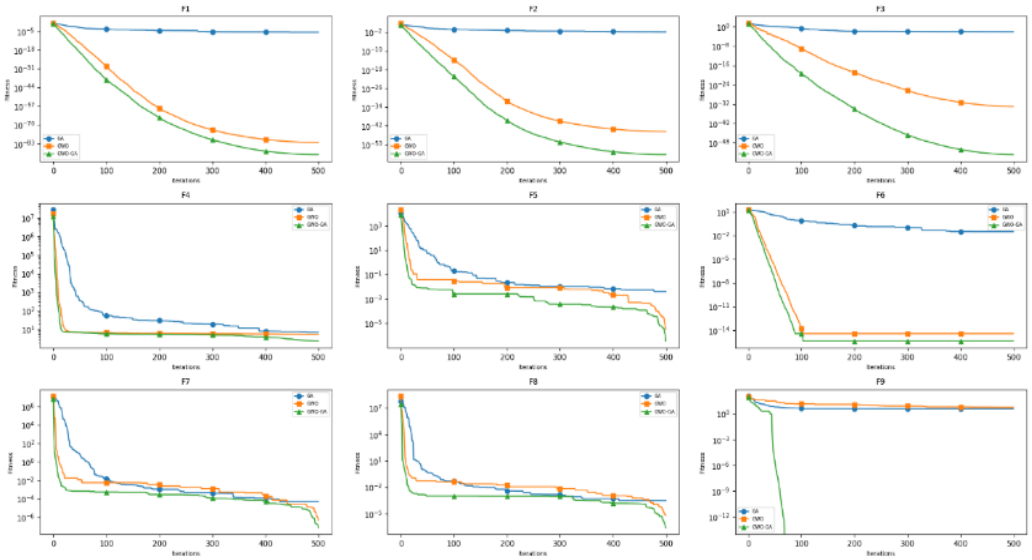
Các hàm thử nghiệm được chọn bao gồm chín hàm phổ biến với độ phức tạp và tính chất khác nhau, được kiểm tra trên các số chiều khác nhau (10, 30, 50) để đảm bảo tính tổng quát của kết quả. Các tham số của thuật toán được thiết lập như sau: kích thước quần thể (popSize) là 50, số chiều (dim) được thử nghiệm với các giá trị 10, 30 và 50, số vòng lặp (Iter) là 500, tỷ lệ đột biến (mutation_rate) là 0.1 và tỷ lệ lai ghép (crossover_rate) là 0.8. Mỗi hàm thử nghiệm được chạy 10 lần để giảm thiểu ảnh hưởng của yếu tố ngẫu nhiên và đảm bảo tính khách quan của kết quả. Kết quả độ thích nghi trung bình qua các lần chạy được sử dụng để đánh giá hiệu quả của thuật toán.

4.2. Kết quả thực nghiệm

Trong phần này, chúng tôi trình bày kết quả thực nghiệm của thuật toán GWO-GA-CLS

trên tập các hàm thử nghiệm chuẩn F1–F9 với số chiều lần lượt là 10, 30 và 50. Ngoài việc báo cáo các giá trị fitness đạt được theo số vòng lặp (được thể hiện trong các Bảng 2, 3, 4), chúng tôi còn bổ sung phần diễn giải nhằm giúp người đọc dễ dàng nắm bắt ý nghĩa của các số liệu và hình minh họa (Hình 3–5).

4.2.1. Thực nghiệm với số chiều là 10



Hình 3. Biểu đồ kết quả thực nghiệm với số chiều là 10

Bảng 2. Bảng thống kê kết quả sau mỗi 100 vòng lặp với số chiều là 10

	Algorithm	100	200	300	400	500
F1	**GA**	5.09E+00	1.67E-04	1.92E-05	5.93E-06	3.32E-06
	GWO	6.79E+00	8.53E-30	3.07E-58	1.38E-72	1.75E-79
	GWO-GA	2.89E+00	6.00E-40	1.28E-67	1.80E-83	2.64E-91
F2	**GA**	6.14E+01	1.89E-01	5.22E-02	2.72E-02	2.28E-02
	GWO	3.54E+02	1.52E-14	2.29E-31	2.53E-40	5.95E-44
	GWO-GA	1.17E+01	1.10E-21	8.70E-40	3.58E-49	2.55E-53
F3	**GA**	4.64E+01	3.19E-01	6.65E-02	1.91E-02	5.42E-03
	GWO	4.19E+01	4.02E-10	1.39E-24	1.81E-34	5.96E-40
	GWO-GA	1.59E+01	2.97E-21	4.71E-36	4.70E-46	5.39E-52
F4	**GA**	1.23E+07	6.67E+01	1.51E+01	5.78E+00	2.75E+00
	GWO	5.22E+06	6.88E+00	6.50E+00	6.34E+00	6.23E+00
	GWO-GA	5.14E+06	3.41E+00	1.48E+00	1.39E+00	7.33E-01
F5	**GA**	1.10E+04	2.49E-01	3.23E-02	1.13E-02	1.06E-02
	GWO	1.21E+04	4.66E-02	1.39E-02	7.44E-03	1.95E-03
	GWO-GA	3.15E+03	8.49E-03	3.03E-03	2.05E-04	1.82E-04
F6	**GA**	1.94E+01	4.16E-01	1.32E-01	7.56E-02	4.85E-02
	GWO	2.02E+01	2.38E-13	4.00E-15	4.00E-15	4.00E-15
	GWO-GA	1.82E+01	4.00E-15	4.00E-15	4.44E-16	4.44E-16

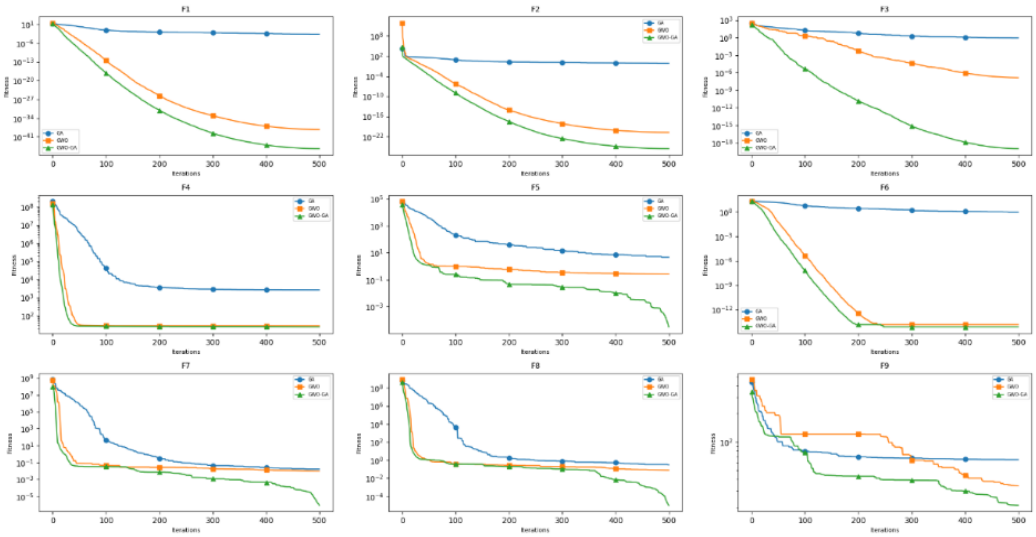
	Algorithm	100	200	300	400	500
F7	GA	4.56E+07	5.84E-03	4.93E-04	7.19E-05	5.07E-05
	GWO	9.32E+07	6.52E-03	3.63E-03	6.08E-04	3.19E-04
	GWO-GA	1.07E+06	3.56E-04	1.63E-04	8.00E-05	4.41E-05
F8	GA	1.15E+08	9.54E-02	1.90E-02	9.37E-04	2.90E-04
	GWO	3.05E+07	4.74E-02	1.47E-02	7.12E-03	2.03E-03
	GWO-GA	2.43E+06	1.38E-03	1.24E-03	1.08E-03	1.06E-04
F9	GA	8.40E+01	7.21E+00	7.05E+00	6.98E+00	6.97E+00
	GWO	1.19E+02	1.51E+01	2.19E+00	2.06E+00	2.03E+00
	GWO-GA	7.38E+01	0.00E+00	0.00E+00	0.00E+00	0.00E+00

Bảng 2 thể hiện giá trị thích nghi trung bình của các thuật toán GA, GWO và GWO-GA sau mỗi 100 vòng lặp trên các hàm kiểm thử F1 đến F9. Quan sát số liệu cho thấy thuật toán GWO-GA có tốc độ hội tụ nhanh hơn đáng kể so với GA và GWO, đồng thời đạt giá trị tối ưu thấp hơn, chứng tỏ khả năng khai thác và tìm kiếm cực trị toàn cục hiệu quả hơn.

Cụ thể, trên hàm F1, giá trị thích nghi ban đầu của GWO-GA là 2.89E+00 và giảm xuống 2.64E-91 sau 500 vòng lặp, trong khi GWO đạt 1.75E-79 và GA chỉ giảm đến 3.32E-06. Điều này phản ánh rõ ràng khả năng kết hợp của hai thuật toán đã cải thiện hiệu suất tìm kiếm đáng kể. Hình 3 minh họa quá trình hội tụ của ba thuật toán, cho thấy GWO-GA không chỉ đạt giá trị tối ưu nhanh hơn mà còn duy trì sự ổn định trong hội tụ, đặc biệt trên các hàm có nhiều cực trị cục bộ như F4 và F9.

Trên các hàm F5 và F6, GWO-GA tiếp tục cho thấy hiệu suất vượt trội khi đạt giá trị thích nghi cuối cùng thấp hơn ít nhất một bậc số mũ so với GWO và GA. Điều này chứng minh rằng việc lai ghép giữa GWO và GA không chỉ giúp khai thác tốt hơn trong giai đoạn đầu mà còn hỗ trợ quá trình tinh chỉnh giải pháp ở các giai đoạn sau của quá trình tối ưu hóa.

4.2.2. Thực nghiệm với số chiều là 30



Hình 4. Biểu đồ kết quả thực nghiệm với số chiều là 30

Khi tăng số chiều lên 30, độ phức tạp của bài toán tối ưu hóa gia tăng, gây khó khăn hơn cho các thuật toán tìm kiếm. Tuy nhiên, kết quả ở Bảng 3 tiếp tục khẳng định rằng GWO-GA duy trì được tốc độ hội tụ nhanh hơn so với GA và GWO, đặc biệt trên các hàm F1, F3 và

F5. Điều này cho thấy thuật toán lai có khả năng thích nghi tốt với không gian tìm kiếm có số chiều cao.

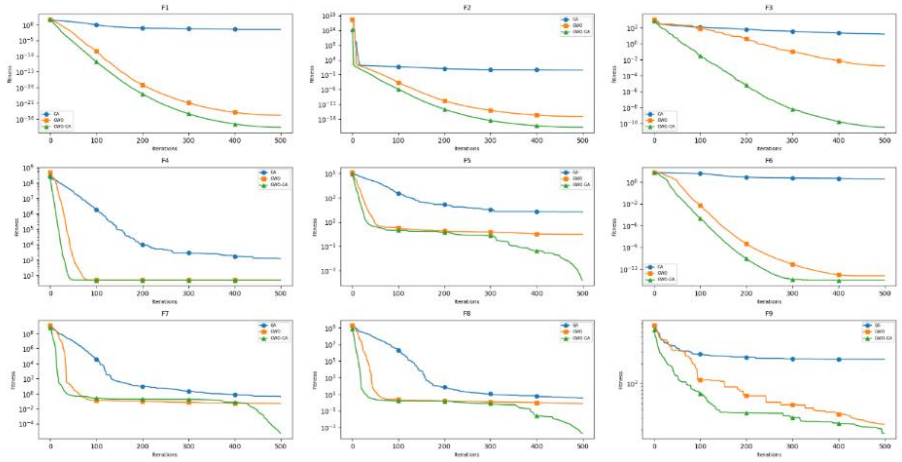
Ví dụ, trên hàm F3, giá trị thích nghi cuối cùng của GA là 1.16E+00, trong khi GWO giảm xuống 9.22E-07, và GWO-GA tiếp tục giảm mạnh xuống 1.37E-18. Đây là một sự cải thiện đáng kể, cho thấy sự kết hợp giữa khả năng tìm kiếm toàn cục của GWO và cơ chế khai thác cục bộ của GA giúp thuật toán hội tụ đến nghiệm tốt hơn.

Hình 4 minh họa sự khác biệt rõ rệt trong tốc độ hội tụ của các thuật toán. Đường hội tụ của GWO-GA cho thấy sự suy giảm giá trị thích nghi ổn định và mạnh mẽ ngay từ giai đoạn đầu của quá trình tối ưu hóa. Điều này là nhờ vào cơ chế lai ghép giúp thuật toán cân bằng giữa khai thác và khám phá, tránh bị mắc kẹt tại các cực trị cục bộ.

Bảng 3. Bảng thống kê kết quả sau mỗi 100 vòng lặp với số chiều là 30

	Algorithm	100	200	300	400	500
F1	**GA**	1.82E+01	6.12E-02	1.24E-02	7.07E-03	3.46E-03
	GWO	3.21E+01	2.86E-13	2.24E-26	8.45E-34	1.05E-37
	GWO-GA	1.50E+01	5.73E-18	8.20E-32	2.60E-40	9.61E-45
F2	**GA**	2.06E+04	7.58E+00	1.88E+00	1.27E+00	9.31E-01
	GWO	8.54E+11	5.74E-07	8.29E-15	6.53E-19	7.02E-21
	GWO-GA	5.25E+04	1.09E-09	2.83E-18	2.44E-23	1.20E-25
F3	**GA**	3.27E+02	1.68E+01	5.74E+00	1.80E+00	1.16E+00
	GWO	3.03E+02	2.23E+00	5.64E-03	4.03E-05	9.22E-07
	GWO-GA	1.66E+02	4.46E-06	1.50E-11	7.55E-16	1.37E-18
F4	**GA**	2.04E+08	4.19E+04	3.51E+03	2.86E+03	2.63E+03
	GWO	1.56E+08	2.77E+01	2.71E+01	2.71E+01	2.71E+01
	GWO-GA	1.36E+08	2.54E+01	2.49E+01	2.43E+01	2.41E+01
F5	**GA**	6.10E+04	1.97E+02	3.85E+01	1.40E+01	7.25E+00
	GWO	6.67E+04	9.83E-01	5.76E-01	3.46E-01	2.82E-01
	GWO-GA	3.74E+04	2.44E-01	4.74E-02	2.81E-02	9.70E-03
F6	**GA**	2.04E+01	6.06E+00	2.80E+00	1.62E+00	1.15E+00
	GWO	2.07E+01	4.25E-06	3.42E-13	1.47E-14	1.47E-14
	GWO-GA	1.98E+01	6.37E-08	1.47E-14	7.55E-15	7.55E-15
F7	**GA**	6.84E+08	4.24E+01	3.11E-01	4.29E-02	2.64E-02
	GWO	6.26E+08	4.85E-02	2.69E-02	1.76E-02	1.32E-02
	GWO-GA	1.00E+08	3.33E-02	7.14E-03	1.22E-03	4.57E-04
F8	**GA**	6.10E+08	4.04E+03	1.83E+00	7.90E-01	5.20E-01
	GWO	8.60E+08	3.78E-01	2.77E-01	2.01E-01	1.18E-01
	GWO-GA	4.28E+08	3.68E-01	1.99E-01	1.04E-01	7.12E-03
F9	**GA**	4.34E+02	7.90E+01	6.92E+01	6.70E+01	6.53E+01
	GWO	4.64E+02	1.21E+02	1.21E+02	6.35E+01	4.39E+01
	GWO-GA	3.44E+02	7.72E+01	4.30E+01	3.90E+01	3.00E+01

4.2.3. Thực nghiệm với số chiều là 50



Hình 5. Biểu đồ kết quả thực nghiệm với số chiều là 50

Bảng 4. Bảng thống kê kết quả sau mỗi 100 vòng lặp với số chiều là 50

	Algorithm	100	200	300	400	500
F1	**GA**	4,55E+01	8,97E-01	8,54E-02	4,98E-02	3,64E-02
	GWO	4,71E+01	3,73E-09	6,32E-20	1,35E-25	1,23E-28
	GWO-GA	3,98E+01	1,85E-12	9,43E-23	4,60E-29	2,56E-32
F2	**GA**	4,75E+17	5,00E+01	1,14E+01	5,20E+00	4,38E+00
	GWO	4,30E+17	2,56E-04	1,52E-10	9,07E-14	2,23E-15
	GWO-GA	2,12E+14	1,30E-06	2,30E-13	3,33E-17	5,94E-19
F3	**GA**	7,29E+02	1,18E+02	6,23E+01	3,65E+01	2,15E+01
	GWO	1,11E+03	7,48E+01	4,13E+00	9,78E-02	7,29E-03
	GWO-GA	7,22E+02	2,92E-02	6,21E-06	6,20E-09	1,69E-10
F4	**GA**	2,50E+08	1,74E+06	9,74E+03	2,86E+03	1,66E+03
	GWO	4,82E+08	4,83E+01	4,78E+01	4,78E+01	4,78E+01
	GWO-GA	3,07E+08	4,69E+01	4,69E+01	4,69E+01	4,69E+01
F5	**GA**	2,50E+08	1,74E+06	9,74E+03	2,86E+03	1,66E+03
	GWO	4,82E+08	4,83E+01	4,78E+01	4,78E+01	4,78E+01
	GWO-GA	3,07E+08	4,69E+01	4,69E+01	4,69E+01	4,69E+01
F6	**GA**	2,07E+01	1,41E+01	4,66E+00	3,52E+00	3,04E+00
	GWO	2,08E+01	5,70E-04	3,19E-09	4,53E-12	1,67E-13
	GWO-GA	2,07E+01	9,44E-06	2,76E-11	3,95E-14	2,89E-14
F7	**GA**	9,34E+08	3,69E+04	9,33E+00	2,00E+00	7,36E-01
	GWO	1,19E+09	1,33E-01	9,63E-02	6,85E-02	5,53E-02
	GWO-GA	6,11E+08	2,41E-01	1,84E-01	1,69E-01	6,67E-02
F8	**GA**	1,61E+09	1,90E+06	6,84E+01	1,07E+01	6,02E+00
	GWO	1,89E+09	2,17E+00	1,46E+00	1,17E+00	9,45E-01
	GWO-GA	7,46E+08	1,55E+00	1,39E+00	7,57E-01	2,66E-02

	Algorithm	100	200	300	400	500
F9	GA	7,36E+02	2,75E+02	2,48E+02	2,35E+02	2,32E+02
	GWO	7,52E+02	1,13E+02	6,54E+01	4,73E+01	3,42E+01
	GWO-GA	6,49E+02	6,98E+01	3,57E+01	3,04E+01	2,47E+01

Với số chiều tăng lên 50, không gian tìm kiếm mở rộng đáng kể, khiến các thuật toán phải đối mặt với nhiều thách thức hơn trong việc hội tụ đến nghiệm tối ưu. Bảng 4 cho thấy GA có dấu hiệu chậm hội tụ hơn so với hai thuật toán còn lại, trong khi GWO-GA vẫn duy trì được tốc độ tìm kiếm hiệu quả.

Trên hàm F2, GA chỉ đạt giá trị thích nghi 4.38E+00 sau 500 vòng lặp, trong khi GWO giảm xuống 2.23E-15 và GWO-GA tiếp tục cải thiện đáng kể với giá trị 5.94E-19. Điều này cho thấy rằng GWO-GA không chỉ tối ưu hóa nhanh hơn mà còn duy trì được khả năng tìm kiếm hiệu quả trong các không gian tìm kiếm phức tạp.

Hình 5 minh họa rõ quá trình hội tụ của ba thuật toán trên không gian có số chiều cao. GWO-GA duy trì tốc độ suy giảm giá trị thích nghi nhanh hơn và ổn định hơn, trong khi GA có dấu hiệu dừng lại ở giá trị cao hơn do khả năng khai thác hạn chế. Đây là bằng chứng thuyết phục cho thấy thuật toán lai giúp cân bằng giữa quá trình khám phá và khai thác, đồng thời tận dụng được điểm mạnh của cả GWO và GA để cải thiện chất lượng nghiệm tối ưu.

4.3. Đánh giá thuật toán từ kết quả thực nghiệm

Nghiên cứu này đánh giá hiệu quả của thuật toán GWO-GA thông qua các hàm kiểm thử tiêu chuẩn F1 đến F9, với kích thước 10, 30 và 50 chiều và thuật toán được chạy 500 lần trên mỗi trường hợp để đảm bảo độ tin cậy. Thuật toán cũng tích hợp thêm bước tìm kiếm cục bộ lốc xoáy, giúp cải thiện khả năng hội tụ và tối ưu hóa toàn cục. Kết quả thực nghiệm cho thấy:

4.3.1. Khả năng hội tụ và tìm kiếm cực trị toàn cục

GWO-GA thể hiện tốc độ hội tụ nhanh và nhất quán, đặc biệt trên các hàm F1, F2, F3 và F5. Thuật toán đạt được các giá trị tối ưu hoặc gần tối ưu trên tất cả các kích thước biến, nhờ vào bước tìm kiếm cục bộ giúp tinh chỉnh và cải thiện độ chính xác trong quá trình tối ưu hóa.

4.3.2. Khả năng xử lý các hàm có nhiều cực trị cục bộ

Với các hàm chứa nhiều cực trị cục bộ như F4 và F9, GWO-GA đã đạt các giá trị tối ưu tốt hơn nhờ sự kết hợp của GWO, GA và tìm kiếm cục bộ cho phép thuật toán thoát khỏi các cực trị cục bộ và tăng cường khả năng khám phá không gian giải pháp.

4.3.3. So sánh với các thuật toán GWO và GA đơn lẻ

So với GWO và GA riêng lẻ, GWO-GA cho thấy độ chính xác và tốc độ hội tụ vượt trội, đặc biệt trên các hàm F5 và F7. Thuật toán này nhanh chóng đạt được các giá trị tốt nhất, nhờ vào vai trò của tìm kiếm cục bộ lốc xoáy trong việc tinh chỉnh vị trí các cá thể và giảm số vòng lặp cần thiết để đạt đến tối ưu.

4.3.4. Khả năng duy trì sự đa dạng của quần thể

GWO-GA duy trì tốt sự đa dạng trong quần thể, giúp thuật toán tránh hội tụ sớm và không bị mắc kẹt tại các cực trị cục bộ. Sự đa dạng này cải thiện khả năng khám phá không gian giải pháp, đặc biệt quan trọng với các bài toán có nhiều cực trị cục bộ.

Tóm lại, GWO-GA là một phương pháp tối ưu hóa mạnh mẽ, có khả năng cân bằng giữa khai thác và khám phá. Kết quả thực nghiệm cho thấy thuật toán này là một phương pháp hiệu quả và đáng tin cậy cho các bài toán tối ưu hóa phức tạp.

5. Kết luận và phương hướng phát triển

Nghiên cứu này đã giới thiệu và đánh giá thuật toán GWO-GA, một sự kết hợp giữa GWO và GA, tích hợp thêm bước tìm kiếm cục bộ để cải thiện khả năng tìm kiếm chi tiết và hội tụ đến cực trị toàn cục. Kết quả thực nghiệm cho thấy GWO-GA vượt trội hơn các phiên bản GWO và GA riêng lẻ về tốc độ hội tụ, độ ổn định và hiệu quả tìm kiếm cực trị trong các không gian tối ưu hóa phức tạp.

Tìm kiếm cục bộ lốc xoáy góp phần nâng cao hiệu suất của GWO-GA, đặc biệt khi kích thước biến tăng lên, giúp thuật toán hoạt động hiệu quả ngay cả trong các không gian tìm kiếm lớn và nhiều cực trị cục bộ. Điều này khẳng định GWO-GA là một công cụ mạnh mẽ và tiềm năng cho các bài toán tối ưu hóa thực tế phức tạp.

Kết quả nghiên cứu cho thấy sự kết hợp giữa GWO, GA và tìm kiếm cục bộ lốc xoáy trong GWO-GA giúp khai thác tối đa ưu điểm của từng thành phần, mang lại một giải pháp tối ưu hóa hiệu quả. Các hướng phát triển tiếp theo bao gồm tinh chỉnh thêm thuật toán GWO-GA và thử nghiệm trên các bài toán phức tạp hơn, ứng dụng trong đa lĩnh vực để khẳng định giá trị và khả năng ứng dụng của thuật toán này.

Tài liệu tham khảo

Abdel-Basset, M., Abdel-Fatah, L., & Sangaiah, A. K. (2018). Metaheuristic algorithms: A comprehensive review. In *Computational Intelligence for Multimedia Big Data on the Cloud with Engineering Applications: Intelligent Data-Centric Systems* (pp. 185–231). https://doi.org/10.1016/B978-0-12-813314-9.00010-4

Abd Elaziz, M., & Oliva, D. (2018). Parameter estimation of solar cells diode models by an improved opposition-based whale optimization algorithm. *Energy Conversion and Management, 171*, 1843–1859. https://doi.org/10.1016/j.enconman.2018.05.062

Emary, E., Zawbaa, H. M., & Hassanien, A. E. (2016). Binary grey wolf optimization approaches for feature selection. *Neurocomputing, 172*, 371–381. https://doi.org/10.1016/j.neucom.2015.06.083

Fu, Y., Xiao, H., Lee, L. H., & Huang, M. (2021). Stochastic optimization using grey wolf optimization with optimal computing budget allocation. *Applied Soft Computing, 103*, 107154. https://doi.org/10.1016/j.asoc.2021.107154

Jamil, M., & Yang, X.-S. (2013). A literature survey of benchmark functions for global optimization problems. *International Journal of Mathematical Modelling and Numerical Optimisation, 4*(2), 150–194. https://doi.org/10.48550/arXiv.1308.4008

Mirjalili, S. (2015). How effective is the grey wolf optimizer in training multi-layer perceptrons. *Applied Intelligence, 43*(1), 150–161. https://doi.org/10.1007/s10489-014-0645-7

Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. *Advances in Engineering Software, 69*, 46–61. https://doi.org/10.1016/j.advengsoft.2013.12.007

Mirjalili, S., Saremi, S., Mirjalili, S. M., & Coelho, L. D. S. (2016). Multi-objective grey wolf optimizer: A novel algorithm for multi-criterion optimization. *Expert Systems with Applications, 47*, 106–119. https://doi.org/10.1016/j.eswa.2015.10.039

Mondal, M., Srivastava, D., & Chitkara, U. (2023). A genetic algorithm-based approach to solve a new time-limited travelling salesman problem. *International Journal of Distributed Systems Technology, 14*(2), 1–14. https://doi.org/10.4018/IJDST.317377

Panda, M., & Das, B. (2019). Grey wolf optimizer and its applications: A survey. In *Proceedings of the Third International Conference on Microelectronics, Computing and Communication Systems* (Lecture Notes in Electrical Engineering, pp. 179–194). https://doi.org/10.1007/978-981-13-7091-5_17

Rafaely, B., & Bennell, J. A. (2006). Optimisation of FTSE 100 tracker funds: A comparison of genetic algorithms and quadratic programming. *Management Finance, 32*(6), 477–492. https://doi.org/10.1108/03074350610666210

Shial, G., Sahoo, S., & Panigrahi, S. (2023). An enhanced GWO algorithm with improved explorative search capability for global optimization and data clustering. *Applied Artificial Intelligence, 37*(1), e2166232. https://doi.org/10.1080/08839514.2023.2166232

Tomar, V., Bansal, M., & Singh, P. (2023). Metaheuristic algorithms for optimization: A brief review. *Engineering Proceedings, 59*(1), 238. https://doi.org/10.3390/engproc2023059238

Tuhus-Dubrow, D., & Krarti, M. (2010). Genetic-algorithm based approach to optimize building envelope design for residential buildings. *Building and Environment, 45*(7), 1574–1581. https://doi.org/10.1016/j.buildenv.2010.01.005

Yao, Z., & Xu, Y. (2024). An improved genetic algorithm for robot path planning. *Journal of Computational Methods in Science and Engineering, 24*(3), 1331–1340. https://doi.org/10.3233/JCM-247133

Zhang, S., Luo, Q., & Zhou, Y. (2017). Hybrid grey wolf optimizer using elite opposition-based learning strategy and simplex method. *International Journal of Computational Intelligence Applications, 16*(02), 1750012. https://doi.org/10.1142/S1469026817500122