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.ns.2399.1873***

PHÂN TÍCH SO SÁNH CÁC THUẬT TOÁN TỐI ƯU HÓA LAI KẾT HỢP THUẬT TOÁN THỎ NHÂN TẠO VÀ THUẬT TOÁN TỐI ƯU HÓA CHIM HẢI ÂU BẮC CỰC

Lê Hoàng Minh Nhật, Bạch Ngọc Vy, Phùng Thị Ly 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: 15/5/2025; Ngày nhận chỉnh sửa: 07/8/2025; Ngày duyệt đăng: 25/8/2025

Tóm tắt

Bài báo mang đến một bước đột phá trong lĩnh vực tối ưu hóa bằng cách kết hợp hai thuật toán lấy cảm hứng từ tự nhiên: Arctic Puffin Optimization (APO) và Artificial Rabbits Optimization (ARO). Nghiên cứu đề xuất năm chiến lược lai như tuần tự, luân phiên và song song, nhằm tận dụng khả năng thăm dò toàn cục của APO và khai thác cục bộ hiệu quả của ARO. Các phương án kết hợp được đánh giá qua sáu hàm benchmark chuẩn với số chiều bài toán khác nhau. Kết quả cho thấy các chiến lược kết hợp, đặc biệt là luân phiên và song song, vượt trội hơn ARO và APO độc lập về tốc độ hội tụ, độ chính xác và khả năng tránh cực trị cục bộ. Sự kết hợp này tận dụng khả năng thăm dò toàn cục của APO và khai thác cục bộ của ARO, mang lại giải pháp hiệu quả cho các bài toán tối ưu hóa đa chiều. Hướng phát triển tương lai tập trung vào tự động điều chỉnh chiến lược lai và ứng dụng thực tiễn như tối ưu tuyến đường hoặc quản lý năng lượng.

Từ khóa: *APO, ARO, Thuật toán lai ghép, Tối ưu hóa.*

Trích dẫn: Lê, H. M. N., Bạch, N. V., Phùng, T. L., & Đinh, N. T. N. (2026). Phân tích so sánh các thuật toán tối ưu hóa lai kết hợp thuật toán thỏ nhân tạo và thuật toán tối ưu hóa chim hải âu bắc cực. *Tạp chí Khoa học Đại học Đồng Tháp, 15*(8), 96-109. https://doi.org/10.52714/dthu.ns.2399.1873

COMPARATIVE ANALYSIS OF HYBRID OPTIMIZATION ALGORITHMS COMBINED ARTIFICIAL RABBITS OPTIMIZATION AND ARCTIC PUFFIN OPTIMIZATION

Le Hoang Minh Nhat, Bach Ngoc Vy, Phung Thi Ly, 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: 15/5/2025; Received in revised form: 07/8/2025; Accepted: 25/8/2025

Abstract

The paper delivers a breakthrough in the optimization field by combining two nature-inspired algorithms: Arctic Puffin Optimization (APO) and Artificial Rabbits Optimization (ARO). The research proposes five hybrid strategies, which include sequential, alternating, and parallel, to take advantage of the global exploration capability of APO and exploit the local efficiency of ARO. These strategies are evaluated by using six standard Benchmark funtions with different problem dimensions. The results show that the hybrid strategies, especially the alternating and parallel ones, outperform ARO and APO independently in terms of convergence speed, accuracy, and local extreme avoidance. This combination deploys the global exploration capabilities of APO and the local exploitation capabilities of ARO, providing effective solutions to multidimensional optimization problems. Future development direction should focuse on automatic tuning of hybrid strategies and practical applications, such as route optimization or energy management.

Keywords: *APO, ARO, Hybrid Algorithm, Optimization.*

1. Đặt vấn đề

Trong bối cảnh phát triển nhanh chóng của khoa học và công nghệ, các bài toán tối ưu phức tạp mà các phương pháp truyền thống không thể xử lý hiệu quả do không gian tìm kiếm quá lớn, phi tuyến tính hoặc có nhiều cực trị cục bộ. Trong nhiều trường hợp, việc tìm lời giải tối ưu tuyệt đối là không khả thi trong thời gian hợp lý, đặc biệt đối với các bài toán NP-hard. Những bài toán này không chỉ yêu cầu các giải pháp chính xác mà còn phải được tìm kiếm trong thời gian hợp lý, tạo ra thách thức lớn cho các phương pháp tối ưu hóa truyền thống (Mirjalili, 2014).

Thuật toán metaheuristic ra đời như một giải pháp hiệu quả, với khả năng tìm kiếm lời giải tối ưu gần đúng thông qua việc mô phỏng các hiện tượng tự nhiên và tận dụng tính ngẫu nhiên thay vì dựa vào 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 một cách linh hoạt (Tomar, 2023).

Metaheuristic có thể được chia thành nhiều loại, bao gồm các thuật toán dựa trên tiến hóa, hành vi quần thể, và vật lý, mỗi loại đều có ưu điểm riêng trong việc giải quyết các bài toán đa dạng. Một khảo sát về 110 thuật toán metaheuristic như Genetic Algorithm – GA (Holland, 1992), Particle Swarm Optimization – PSO (Kennedy, 1995), Firefly Algorithm – FFA (Yang, 2010) đã cho thấy sự phát triển mạnh mẽ và tiềm năng ứng dụng rộng rãi của chúng trong các bài toán thực tiễn. Tuy nhiên, việc lựa chọn thuật toán phù hợp và cải tiến hiệu suất vẫn là một vấn đề cần nghiên cứu sâu hơn (Alorf, 2023; Abdel-Basse, 2018).

Thuật toán Artic Puffin Optimization (APO) và Artificial Rabbits Optimization (ARO) là hai thuật toán meta-heuristic hiện đại lấy cảm hứng từ hành vi sinh học trong tự nhiên. APO mô phỏng hành vi sinh tồn của loài chim hải âu Bắc Cực, kết hợp giữa khả năng tìm kiếm cục bộ và bước nhảy Levy (chiến lược thoát săn mồi) để thoát khỏi cực trị địa phương (Wen-chuan Wang, 2024). Trong khi đó, ARO mô phỏng hành vi kiếm ăn và ẩn náu của loài thỏ, khai thác cơ chế chuyển động ngẫu nhiên có hướng để tối ưu hóa quá trình tìm kiếm (Liying Wang, 2022). Cả hai thuật toán đều cho thấy hiệu quả vượt trội trong việc giải các bài toán tối ưu phức tạp, nhờ khả năng cân bằng tốt giữa khai thác (exploitation) và khám phá (exploration). Bài báo này đề xuất và so sánh 5 chiến lược lai giữa APO và ARO nhằm khai thác ưu điểm của cả hai thuật toán, cải thiện hiệu suất trên các bài toán tối ưu hóa đa chiều.

2. Cơ sở lý thuyết

2.1. Thuật toán Chim hải âu Bắc Cực (APO)

Thuật toán Arctic Puffin Optimization (APO) là một thuật toán tối ưu hóa mô phỏng quá trình sinh tồn và tìm kiếm thức ăn của loài chim hải âu Bắc Cực. Thuật toán này khai thác hành vi tìm kiếm, hợp tác và săn mồi của chim hải âu để tìm kiếm giải pháp tối ưu (Wen-chuan Wang, 2024).

Hành vi chính của chim hải âu trong thuật toán này được chia thành ba giai đoạn: khởi tạo quần thể, thăm dò và hành vi khai thác. Các mô hình toán học cho mỗi giai đoạn được mô tả như sau (Wen-chuan Wang, 2024):

2.1.1. Khởi tạo quần thể

Mỗi chim hải âu Bắc Cực đại diện cho một giải pháp tiềm năng tham gia vào quá trình tối ưu hóa. Quá trình tạo ra quần thể khởi tạo được mô tả bằng phương trình sau:

$$\overrightarrow{X_i^t} = rand * (ub - lb) + lb, i = 1,2,3 \ldots N \quad (1)$$

Trong đó $\overrightarrow{X_i^t}$ đại diện cho vị trí của cá thể hải âu thứ i tại thời điểm t, rand tạo ra một số ngẫu nhiên từ 0 đến 1; ub và lb lần lượt đại diện cho giới hạn trên và dưới trong không gian

giải pháp, N là số cá thể trong quần thể.

2.1.2. Hành vi thăm dò

Chiến lược tìm kiếm trên không, phương trình cập nhật vị trí liên quan đến hành vi kiếm ăn trên không được thể hiện như sau:

$$\overrightarrow{Y_i^{t+1}} = \overrightarrow{X_i^t} + \left(\overrightarrow{X_i^t} - \overrightarrow{X_r^t}\right) * L(D) + R \quad (2)$$

Trong đó, $\overrightarrow{Y_i^{t+1}}$ là vị trí mới của cá thể hải âu i trong không gian tìm kiếm. Cá thể hải âu di chuyển theo một cá thể tham chiếu. Hệ số bổ sung R tính bằng:

$$R = round\big(0.5 * (0.05 + rand)\big) * \alpha \quad (3)$$

Chiến lược sà xuống săn mồi dùng để mô hình hóa hành vi lao xuống săn mồi, hệ số vận tốc S được sử dụng để điều chỉnh độ dịch chuyển của hải âu trong quá trình lao xuống. Dưới đây là phương trình cập nhật vị trí của hải âu:

$$\overrightarrow{Z_i^{t+1}} = \overrightarrow{Y_i^{t+1}} * S \quad (4)$$

$$S = tan((rand - 0.5) * \pi \quad (5)$$

Trong chiến lược bay này, chim hải âu điều chỉnh độ dịch chuyển của mình trong giai đoạn đầu tiên bằng cách đưa vào hệ số vận tốc S.

2.1.3. Hành vi khai thác dưới nước

Thu thập kiếm ăn là chiến lược mà những con chim ở trên mặt biển sẽ quan sát hành vi của các thành viên khác để xác định các điểm lặn hoặc nguồn thức ăn. Dưới đây là phương trình mô tả cách cập nhật vị trí của hải âu:

$$\overrightarrow{W_i^{t+1}} = \{\overrightarrow{X_{r1}^t} + F * L(D) * \left(\overrightarrow{X_{r2}^t} - \overrightarrow{X_{r3}^t}\right) rand \geq 0.5 \quad (6)$$

$$\overrightarrow{X_{r1}^t} + F * \left(\overrightarrow{X_{r2}^t} - \overrightarrow{X_{r3}^t}\right) rand < 0.5 \quad (7)$$

Trong các phương trình này, khi $rand < 0.5$, chim hải âu tham gia kiếm ăn hợp tác với các hải âu khác $\overrightarrow{X_{r1}^t}$, $\overrightarrow{X_{r2}^t}$, $\overrightarrow{X_{r3}^t}$ với các biến r_1, r_2, r_3 là các số nguyên ngẫu từ 1 đến $N - 1$ (không bao gồm i), sử dụng hệ số hợp tác F = 0.5 để khám phá môi trường xung quanh.

Tăng cường tìm kiếm chim hải âu phải thay đổi vị trí dưới nước của mình để tìm kiếm thêm cá hoặc các nguồn thức ăn dưới nước khác. Phương trình cập nhật vị trí được thể hiện như sau:

$$\overrightarrow{Y_i^{t+1}} = \overrightarrow{W_i^{t+1}} * (1 + f) \quad (8)$$

$$f = 0.1 * (rand - 1) * \frac{T - t}{T} \quad (9)$$

Trong đó T biểu thị tổng số lần lặp lại và t biểu thị số lần lặp hiện tại. Tham số f là một yếu tố thích ứng được sử dụng để điều chỉnh vị trí của hải âu trong nước.

2.1.4. Tránh kẻ săn mồi

Phương trình cập nhật vị trí cho chiến lược này được thể hiện như sau:

$$\overrightarrow{Z_i^{t+1}} = \{\overrightarrow{X_i^t} + F * L(D) * \left(\overrightarrow{X_{r1}^t} - \overrightarrow{X_{r2}^t}\right) rand \geq 0.5 \quad (10)$$

$$\overrightarrow{X_i^t} + \beta * \left(\overrightarrow{X_{r1}^t} - \overrightarrow{X_{r2}^t}\right) rand < 0.5 \quad (11)$$

Trong đó β là số ngẫu nhiên phân phối đều giữa (0,1). Khi $rand \geq 0{,}5$ chỉ ra rằng có kẻ săn mồi ở gần, chim hải âu chọn né tránh ngay lập tức. Ngược lại, hành vi của chim hải âu có xu hướng thận trọng hơn bằng cách thay đổi vị trí ngẫu nhiên.

2.2. Thuật toán Thỏ nhân tạo (ARO)

Artificial Rabbits Optimization (ARO) là một thuật toán tối ưu hóa lấy cảm hứng từ hành vi tìm kiếm thức ăn và trốn tránh kẻ thù của loài thỏ trong tự nhiên, mô phỏng cách thỏ cân bằng giữa việc khám phá môi trường xung quanh và tập trung khai thác các khu vực có nguồn thức ăn dồi dào để tìm ra lời giải tối ưu (Liying Wang, 2022).

Hành vi chính của thỏ trong thuật toán này được chia thành 2 chiến lược: kiếm ăn vòng quanh và ẩn náu ngẫu nhiên, cùng với cơ chế thu nhỏ năng lượng (energy shrink) để điều chỉnh chiến lược tìm kiếm (Liying Wang, 2022).

2.2.1. Khởi tạo quần thể

Khởi tạo ngẫu nhiên một quần thể các thỏ nhân tạo, mỗi thỏ đại diện cho một giải pháp tiềm năng tham gia vào quá trình tối ưu hóa. Quá trình tạo ra quần thể khởi tạo được mô tả bằng phương trình sau:

$$\overrightarrow{X_i^t} = rand * (ub - lb) + lb, i = 1,2,3.. \quad (12)$$

Trong đó $\overrightarrow{X_i^t}$ đại diện cho vị trí của cá thể thỏ i tại thời điểm t, $rand$ tạo ra một số ngẫu nhiên từ 0 đến 1, ub và lb lần lượt đại diện cho giới hạn trên và dưới của bài toán (không gian giải pháp), N là số cá thể trong quần thể.

2.2.2. Cơ chế thu nhỏ năng lượng

Hệ số năng lượng $A(t)$ quyết định thuật toán đang ở giai đoạn khám phá hay khai thác, hệ số năng lượng được tính bằng công thức sau:

$$A(t) = 4 \times \left(1 - \frac{t}{T}\right) \times ln\frac{1}{r_1} \quad (13)$$

Trong đó, t là vòng lặp hiện tại, T là số vòng lặp tối đa, r_1 là số ngẫu nhiên trong khoảng [0, 1].

2.2.3. Tính độ dài chạy L

Độ dài chạy L mô phỏng mức độ di chuyển của thỏ, phản ảnh sự thu hẹp phạm vi di chuyển khi thuật toán tiến gần đến giải pháp tối ưu, được tính bằng công thức:

$$L = (e - e^{-\left(\frac{t}{T}\right)^2}) \times \sin(2\pi r_2) \quad (14)$$

Trong đó, e là cơ số logarit tự nhiên (≈ 2.718), r_2 là số ngẫu nhiên trong [0, 1].

2.2.4. Tính toán tử chạy R

Toán tử chạy R xác định hướng và độ lớn của bước di chuyển trong không gian tìm kiếm. R được tính từ độ dài chạy L và một vector ngẫu nhiên c trong đó:

$$c(k) = \begin{cases} 1 & if\ k = g(l) \\ 0 & else \end{cases} \quad (15)$$

Trong đó, $k = 1, 2, \ldots d$ chỉ số chiều của bài toán, $g = randperm(d)$ là hoán vị ngẫu nhiên các số nguyên từ 1 đến d, $l = \lceil r_3 \cdot d \rceil$ là số chiều được chọn để biến đổi, $r_3 \in (0,1)$ là

số ngẫu nhiên, còn $\lceil . \rceil$ là hàm làm tròn lên. Toán tử R được tính:

$$R = L \times c \quad (16)$$

2.2.5. Hành vi kiếm ăn

Điều kiện và mục đích: nếu $A(t) > 1$. Thăm dò các vùng mới trong không gian tìm kiếm để tìm ra vùng giải pháp tiềm năng, được cập nhật bằng công thức:

$$\overrightarrow{v_i}(t+1) = \overrightarrow{X_j}(t) + R \times (\overrightarrow{X_i}(t) - \overrightarrow{X_j}(t)) + round(0.5 \times (0.05 + r_4)) \times n_1 \quad (17)$$

Trong đó $X_j(t)$ là vị trí thỏ j được chọn ngẫu nhiên với $(j \neq i)$, r_4: Số ngẫu nhiên trong khoảng [0,1], n_1: Số ngẫu nhiên theo phân phối Gauss chuẩn $N(0,1)$, $round\big(0.5 \times (0.05 + r_4)\big) \times n_1$: thêm nhiễu ngẫu nhiên để tăng khả năng khám phá.

Sau đó kiểm tra và cập nhật: Sau khi cập nhật vị trí $\overrightarrow{X_i}(t+1)$, kiểm tra nếu $fit_i < fobj(x_{best})$, cập nhật: $\overrightarrow{X_{best}} = \overrightarrow{X_i}(t+1)$ và ngược lại, giữ nguyên $\overrightarrow{X_i}(t)$.

2.2.6. Hành vi ẩn náu

Điều kiện và mục đích: nếu $A(t) \leq 1$. Mô phỏng hành vi thỏ tạo nhiều hang quanh tổ và chọn ngẫu nhiên một hang để ẩn náu, nhằm tối ưu hóa cục bộ xung quanh vị trí hiện tại. Đầu tiên tính tham số ẩn náu H theo công thức:

$$H = \frac{T-t+1}{T} \times r_5 \quad (18)$$

Sau đó sẽ tạo các hang quanh vị trí hiện tại, với mỗi chiều $k = 1, 2, \ldots, d$, mỗi thỏ i tạo d hang (ứng viên) quanh vị trí hiện tại $\overrightarrow{X_i}(t)$:

$$\overrightarrow{b_{i,k}}(t) = X_i(t) + H \times \left(g_k \times \overrightarrow{X_i}(t)\right) \quad (19)$$

Trong đó, g_k là vector đơn vị chọn ngẫu nhiên để biến đổi theo chiều k (chỉ phần tử thứ k bằng 1, các phần tử khác bằng 0). Sau đó, một chỉ số r từ 1 đến d, lấy hang $\overrightarrow{b_{i,r}}(t)$, với r_6 là số ngẫu nhiên trong đoạn [0, 1], cập nhật ứng viên vào hang được chọn theo công thức:

$$\overrightarrow{v_i}(t+1) = \overrightarrow{X_i}(t) + R \times (r_6 \times \overrightarrow{b_{i,r}}(t) - \overrightarrow{X_j}(t)) \quad (20)$$

Kiểm tra và cập nhật giải pháp như hành vi kiếm ăn.

3. Lý do kết hợp và các phương án đề xuất

APO và ARO là hai thuật toán tối ưu hóa metaheuristic mãnh mẽ, tuy nhiên mỗi thuật toán vẫn 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. APO nổi bật với khả năng khai thác không gian tìm kiếm mạnh mẽ của các cá thể chim hải âu Bắc Cực, giúp duy trì sự cân bằng giữa việc khám phá và khai thác không gian tìm kiếm. Còn ARO mô phỏng hành vi loài thỏ trong tự nhiên, nổi bật với khả năng khai thác mạnh mẽ thông qua các hành vi kiếm ăn và ẩn náu, giúp duy trì quần thể và tăng khả năng thoát khỏi các cực trị cục bộ.

Bên cạnh những ưu điểm, cả ARO và APO đều có những hạn chế. ARO có thể gặp khó khăn với hội tụ sớm trong các bài toán nhiều chiều, trong khi cơ chế thích nghi của APO có thể tốn kém về mặt tính toán đối với một số hàm. Để khắc phục những hạn chế này, nghiên cứu này đề xuất năm chiến lược kết hợp cơ học nhằm kết hợp một cách hiệu quả giữa ARO và APO. Việc lựa chọn kết hợp 5 phương án thay vì chỉ sử dụng một phương án duy nhất xuất phát từ nhu cầu đánh giá toàn diện hiệu quả của sự kết hợp giữa hai thuật toán APO và ARO trong các trường hợp khác nhau. Mỗi phương án đại diện cho một cách tiếp cận riêng biệt:

tuần tự giữa APO-ARO và ngược lại, luân phiên APO-ARO và ngược lại, cuối cùng là phương án chạy song song chia sẻ cá thể tốt nhất. Các phương pháp này giúp khai thác các khía cạnh đa dạng của hai thuật toán, từ khả năng thăm dò, khai thác đến tính tương tác giữa chúng. Sau đó so sánh hiệu suất trên nhiều khía cạnh, như tốc độ hội tụ, khả năng tránh cực trị cục bộ và tính ổn định trên các hàm benchmark.

3.1. Phương án kết hợp tuần tự

3.1.1. Tuần tự APO-ARO

Thuật toán lai tuần tự APO–ARO nhằm giải quyết bài toán tối ưu toàn cục bằng cách chia quá trình tìm kiếm thành hai giai đoạn: thăm dò và khai thác. Giai đoạn đầu sử dụng APO với khả năng thăm dò mạnh nhờ cơ chế chiến lược thoát săn mồi, chạy trong nửa đầu số vòng lặp để khám phá không gian tìm kiếm và xác định vùng tiềm năng. Giai đoạn sau chuyển sang ARO, dùng kết quả từ APO làm đầu vào, tiếp tục khai thác cục bộ nhằm tinh chỉnh nghiệm và cải thiện giá trị fitness. Phương pháp không tạo ra phương trình lai trực tiếp, mà kết hợp tuần tự hai thuật toán: $X_{ARO} \leftarrow X_{APO}$, với quá trình cập nhật vị trí và giá trị fitness diễn ra liên tục trong cả hai pha.

Pseudocode

```
Khởi tạo N cá thể hải âu, X¯ᵢᵗ (i=1,2,...,N), sử dụng Eq.(1) // Khởi tạo
quần thể ban đầu
Đánh giá độ phù hợp mỗi cá thể X¯ᵢᵗ, tìm một cá thể có độ phù hợp tốt nhất
X¯ᵗ
t ← 1
while (t < T)
     // APO (T/2 vòng lặp đầu tiên)
     if t ≤ T/2
        update to APO
     else // ARO (T/2 vòng lặp còn lại)
        update to ARO
     end if
end while
return X¯ᵗ
```

3.1.2. Tuần tự ARO-APO

Phương án này đảo ngược thứ tự so với APO–ARO: ARO được thực thi trước để cân bằng giữa thăm dò và khai thác, sau đó APO tiếp tục thăm dò sâu và tinh chỉnh nghiệm. Trong nửa đầu vòng lặp, ARO khám phá và khai thác không gian tìm kiếm để tạo ra một quần thể ban đầu chất lượng cao. Nửa sau, APO được áp dụng nhằm mở rộng không gian tìm kiếm, tận dụng chiến lược thoát săn mồi để thoát khỏi tối ưu cục bộ và nâng cao độ đa dạng. Cách kết hợp này phù hợp với bài toán yêu cầu giai đoạn đầu ổn định và giai đoạn sau tăng cường thăm dò. Tương tự phương án trước, quá trình lai không dùng phương trình kết hợp trực tiếp mà thực hiện tuần tự: $X_{APO} \leftarrow X_{ARO}$, với fitness toàn cục được cập nhật xuyên suốt hai giai đoạn.

Pseudocode

```
Khởi tạo N cá thể hải âu, X¯ᵢᵗ (i=1,2,...,N), sử dụng Eq.(1) // Khởi tạo
quần thể ban đầu
```

```
Đánh giá độ phù hợp mỗi cá thể X¯ᵢᵗ, tìm một cá thể có độ phù hợp tốt nhất X¯ᵗ
t ← 1
while (t < T)
     // ARO (T/2 vòng lặp đầu tiên)
     if t ≤ T/2
        update to ARO
      else // APO (T/2 vòng lặp còn lại)
        update to APO
      end if
end while
return X¯ᵗ
```

3.2. Phương án kết hợp luân phiên

3.2.1. Luân phiên APO-ARO

Phương án này kết hợp luân phiên APO và ARO nhằm duy trì sự cân bằng giữa thăm dò và khai thác trong suốt quá trình tối ưu. Cụ thể, ARO được áp dụng ở các vòng lặp chẵn để khai thác và khám phá không gian tìm kiếm nhờ cơ chế cân bằng động; APO được áp dụng ở các vòng lẻ để thăm dò sâu và tránh tối ưu cục bộ nhờ chiến lược thoát săn mồi. Việc luân phiên hai thuật toán giúp duy trì độ đa dạng quần thể và tận dụng điểm mạnh của từng phương pháp. Cách tiếp cận này phù hợp với các bài toán phức tạp đòi hỏi khả năng thích ứng liên tục giữa khám phá và khai thác.

Pseudocode

```
Khởi tạo N cá thể hải âu, X¯ᵢᵗ (i=1,2,...,N), sử dụng Eq.(1) // Khởi tạo quần thể ban đầu
Đánh giá độ phù hợp mỗi cá thể X¯ᵢᵗ, tìm một cá thể có độ phù hợp tốt nhất X¯ᵗ
t ← 1
while (t < T)
        if t % 2 != 0
            update to APO
        else
            update to ARO
        end if
end while
return X¯ᵗ
```

3.2.2. Luân phiên ARO-APO

Khác với phương án luân phiên APO–ARO, chiến lược này đảo ngược thứ tự: ARO thực thi ở vòng lẻ và APO ở vòng chẵn. ARO khai thác và khám phá không gian tìm kiếm qua cơ chế cân bằng động, đảm bảo quần thể đa dạng và chất lượng. APO, với chiến lược thoát săn mồi, tiếp tục thăm dò sâu và tinh chỉnh nghiệm, giúp tránh kẹt ở tối ưu cục bộ. Cách kết hợp này duy trì cân bằng giữa thăm dò và khai thác trong suốt quá trình, đồng thời cho phép đánh giá ảnh hưởng của thứ tự luân phiên đến hiệu quả tối ưu hóa, đặc biệt với các bài toán phức tạp.

Pseudocode

```
Khởi tạo N cá thể hải âu, X¯ᵢᵗ (i=1,2,...,N), sử dụng Eq.(1) // Khởi tạo quần thể ban đầu
Đánh giá độ phù hợp mỗi cá thể X¯ᵢᵗ, tìm một cá thể có độ phù hợp tốt nhất X¯ᵗ
t ← 1
while (t < T)
       if t % 2 != 0
           update to ARO
       else
           update to APO
       end if
end while
return X¯ᵗ
```

3.3. Phương án kết hợp song song theo phương pháp chia sẻ cá thể

Phương án này chạy ARO và APO song song trên hai quần thể riêng biệt, định kỳ (mỗi k vòng lặp) trao đổi cá thể tốt nhất nhằm duy trì đa dạng và thúc đẩy hội tụ. ARO khai thác không gian tìm kiếm qua cơ chế cân bằng động, trong khi APO sử dụng chiến lược thoát săn mồi để thăm dò và tinh chỉnh nghiệm. Sau mỗi k vòng, cá thể tốt nhất của ARO được chèn vào quần thể APO và ngược lại: Trao đổi cá thể: $X_{APO} \leftarrow Best_{ARO}$ và $X_{ARO} \leftarrow Best_{APO.}$

Cơ chế chia sẻ này kết hợp hiệu quả ưu điểm của cả hai thuật toán, giúp cải thiện chất lượng nghiệm và tốc độ hội tụ, đặc biệt với các bài toán tối ưu phức tạp. Phương pháp không xây dựng phương trình lai trực tiếp mà kết hợp song song, hỗ trợ trao đổi thông tin giải pháp giữa hai bên.

Pseudocode

```
Khởi tạo N cá thể cho APO và ARO
Đánh giá X¯ᵢᵗ_APO và X¯ᵢᵗ_ARO, tìm cá thể có giá trị fitness tốt nhất trong X¯ᵗ
current_iteration ← 0
while (current_iteration < T)
     for It ← 1:k
        if current_iteration + It > T
               break
        cập nhật theo APO
     end for
     // Giai đoạn khám phá và khai thác:
     for It ← 1:k
        if current_iteration + It > T
               break
        cập nhật theo ARO
     end for
     // Trao đổi cá thể tốt nhất giữa hai quần thể
```

```
    best_APO ← Tìm cá thể tốt nhất trong X⁻ᵗ_APO
    best_ARO ← Tìm cá thể tốt nhất trong X⁻ᵗ_ARO
    rand_index_APO ← Chỉ số ngẫu nhiên trong X⁻ᵗ_APO
    rand_index_ARO ← Chỉ số ngẫu nhiên trong X⁻ᵗ_ARO
    X⁻ᵗ_APO[rand_index_APO] ← best_APO
    X⁻ᵗ_ARO[rand_index_ARO] ← best_ARO
    // Cập nhật giá trị fitness tốt nhất toàn cục
    best_fitness_APO ← Giá trị fitness tốt nhất trong X⁻ᵗ_APO
    best_fitness_ARO ← Giá trị fitness tốt nhất trong X⁻ᵗ_ARO
    X⁻ᵗ ← Cá thể có giá trị min(best_fitness_APO, best_fitness_ARO)
    current_iteration ← current_iteration + k
end while
return X⁻ᵗ
```

4. Mô hình thực nghiệm

4.1. Thiết lập thông số và các hàm mục tiêu

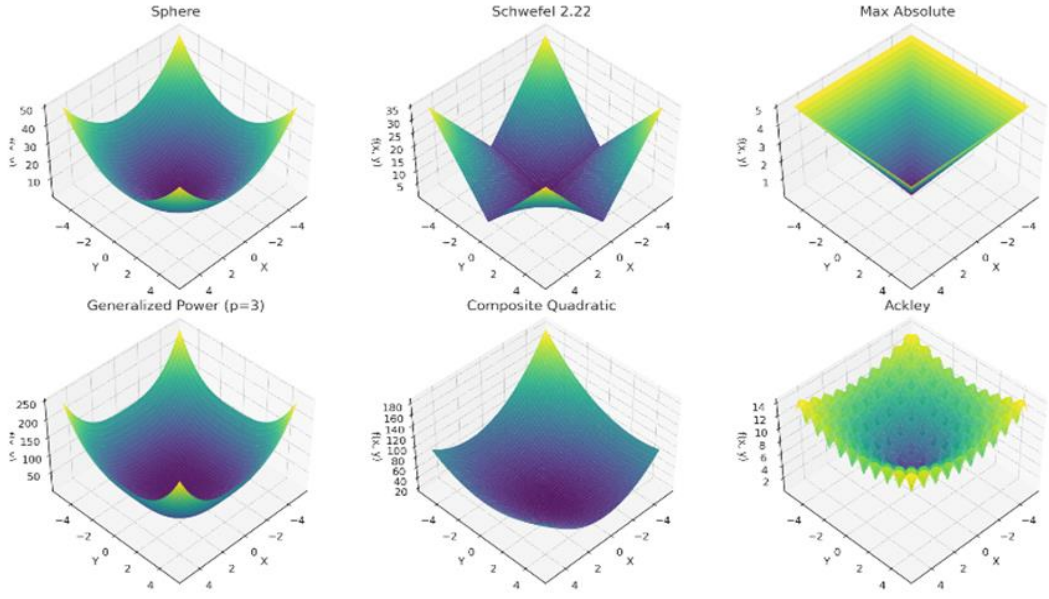
Để đánh giá hiệu suất năm chiến lược lai đề xuất, nhóm thực hiện thực nghiệm trên sáu hàm chuẩn F1–F6 từ Jamil & Yang (2013), bao gồm cả hàm đơn và đa đỉnh, nhằm kiểm tra khả năng thăm dò và khai thác trong không gian tìm kiếm phức tạp (Layeb, 2022; Adam P. Piotrowski, 2023). Thực nghiệm được triển khai bằng Python trên máy tính cấu hình CPU Intel Core i5-10700 (2.9 GHz), RAM 16 GB, Windows 11; sử dụng các thư viện NumPy và Pandas cho tính toán và phân tích.

Các tham số chính: kích thước quần thể $N = 50$, số vòng lặp tối đa $T = 1000$, và số chiều (dim) là 20 và 70. Tất cả thuật toán (bao gồm ARO và APO độc lập) sử dụng cùng quần thể khởi tạo nhằm đảm bảo tính công bằng. Các hàm mục tiêu được thiết lập với giới hạn tìm kiếm $[l_b, u_b]$ tương ứng.

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 **(Sphere)**	$f(x) = \sum_{i=1}^{n} x_i^2$	[-5, 5]	20, 70	0
F2 **(Schwefel's Problem 2.22)**	$f(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	[-10,10]	20, 70	0
F3 **(Max Absolute)**	$f(x) = max_{1 \le i \le n} \lvert x_i \rvert$	[-100, 100]	20, 70	0
F4 **(Generalized Power)**	$f(x) = \sum_{i=1}^{n} \lvert x_i \rvert^i$	[-2, 2]	20, 70	0
F5 **(Composite Quadratic)**	$f(x) = \sum_{i=1}^{n} (x_i^2 + \frac{D}{2} \times x_i^2 + \frac{D}{2} \times x_i^4$	[-100, 100]	20, 70	0

F6 **(Ackley)**	$f(x) = -20exp - 0.2\sqrt{\frac{1}{n}\sum x_i^2} - exp\frac{1}{n}\sum cos2\pi x_i + 20 + exp1$	[-32, 32]	20, 70	0



Hình 1. Biểu diễn 3D của các hàm benchmark

4.2. Quy trình thực nghiệm

Khởi tạo: Tạo quần thể ban đầu cho mỗi hàm chuẩn bằng cách sử dụng phân phối ngẫu nhiên đồng nhất trong khoảng [lb, ub].

Kết quả: Giá trị hàm mục tiêu tốt nhất được ghi lại sau mỗi vòng lặp và lưu vào các tệp CSV tương ứng.

Thực thi các thuật toán:

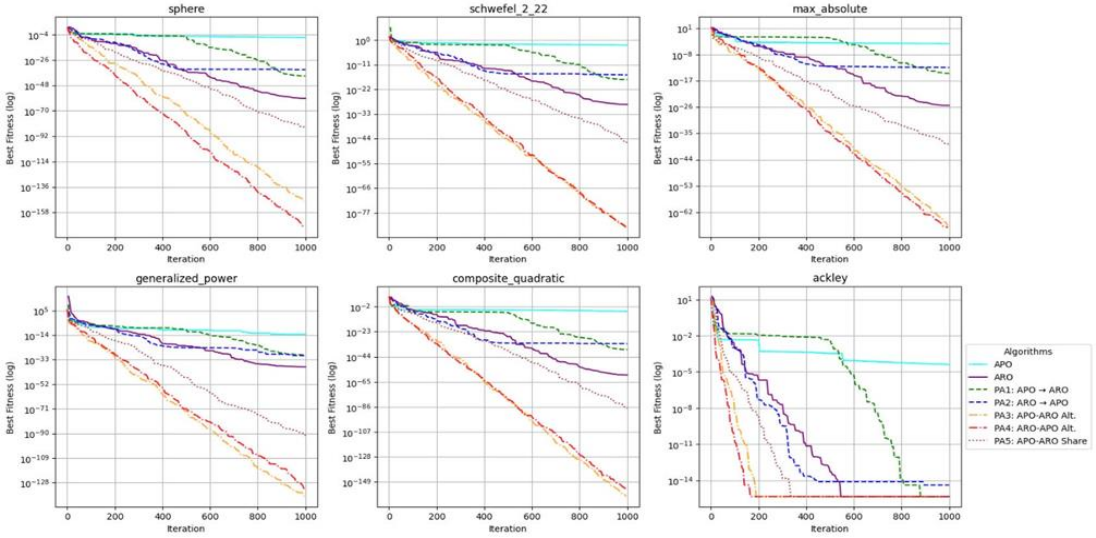
ARO và APO độc lập: Chạy ARO và APO trên mỗi hàm mục tiêu trong 1000 vòng lặp.

Kết hợp tuần tự APO-ARO và ARO-APO: Chạy APO trong 500 vòng lặp, sau đó sử dụng kết quả làm đầu vào cho ARO trong 500 vòng lặp tiếp theo và ngược lại. (sau đây sẽ lần lượt ký hiệu là PA1 và PA2).

Kết hợp luân phiên APO-ARO và ARO-APO: Luân phiên giữa APO và ARO tại mỗi vòng lặp (APO cho vòng lặp lẻ, ARO cho vòng lặp chẵn) trong 1000 vòng lặp và ngược lại. (sau đây sẽ lần lượt ký hiệu là PA3 và PA4).

Kết hợp song song APO-ARO với chia sẻ cá thể: Chạy APO và ARO song song, sau đó trao đổi cá thể tốt nhất giữa hai quần thể sau mỗi k vòng lặp (k = 10), lặp lại cho đến khi đạt 1000 vòng lặp. (sau đây sẽ ký hiệu là PA5).

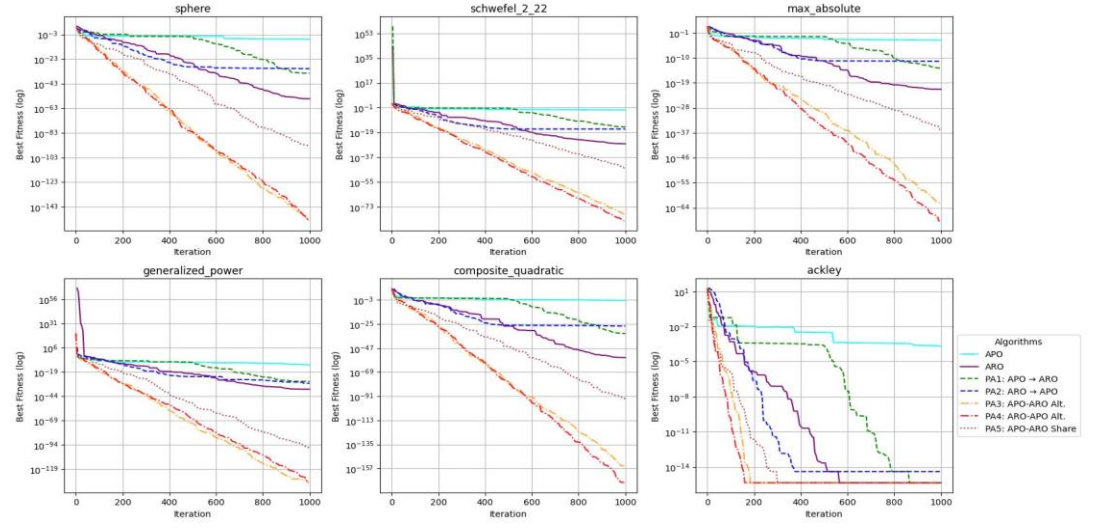
4.3. Kết quả thực nghiệm



Hình 2. Biểu đồ kết quả thực nghiệm với số chiều là 20

Bảng 2. Kết quả giá trị hàm mục tiêu tốt nhất với số chiều là 20

	F1	F2	F3	F4	F5	F6
APO	7.74E-09	6.00E-04	2.49E-05	1.26E-11	1.23E-05	8.80E-04
ARO	4.28E-58	7.98E-29	1.50E-21	1.61E-37	1.02E-60	4.44E-16
PA1	1.50E-39	2.01E-19	6.50E-15	9.82E-30	1.21E-33	4.00E-15
PA2	2.67E-31	4.37E-19	3.64E-12	4.59E-32	1.12E-30	4.00E-15
PA3	3.81E-16	3.99E-89	1.12E-70	2.20E-112	7.54E-155	4.44E-16
PA4	2.08E-16	4.16E-84	3.45E-64	5.91E-123	2.75E-163	4.44E-16
PA5	3.79E-88	7.10E-51	1.76E-39	1.17E-90	3.69E-90	4.44E-16



Hình 3. Biểu đồ kết quả thực nghiệm với số chiều là 70

Bảng 3. Kết quả giá trị hàm mục tiêu tốt nhất với số chiều là 70

	F1	F2	F3	F4	F5	F6
APO	7.18E-05	7.00E-03	1.00E-03	5.00E-00	4.65E-06	2.00E-04
ARO	2.13E-61	2.28E-33	6.00E-25	6.77E-42	9.92E-54	4.44E-16
PA1	1.12E-31	2.54E-18	5.45E-14	7.37E-34	1.17E-34	4.44E-16
PA2	2.10E-27	3.80E-15	1.17E-11	1.14E-29	4.02E-28	2.88E-14
PA3	3.18E-16	4.90E-86	3.02E-65	2.01E-134	6.47E-162	4.44E-16
PA4	5.06E-16	5.29E-82	4.17E-70	6.32E-143	1.26E-156	4.44E-16
PA5	6.15E-92	6.20E-50	3.11E-38	3.35E-95	2.12E-86	4.44E-16

4.4. Đánh giá

Nghiên cứu này đánh giá hiệu quả của các phương án kết hợp APO-ARO thông qua các hàm kiểm thử tiêu chuẩn F1 đến F6, với kích thước bài toán lần lượt là 20 và 70 chiều, và thuật toán được chạy 1000 vòng lặp trên mỗi trường hợp để đảm bảo độ tin cậy. Các phương án kết hợp (PA1 đến PA5) tích hợp các chiến lược tuần tự, luân phiên và song song, nhằm cải thiện khả năng hội tụ và tối ưu hóa toàn cục. Kết quả thực nghiệm cho thấy:

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

Các phương án kết hợp, đặc biệt PA3 và PA4 thể hiện tốc độ hội tụ nhanh và nhất quán, đặc biệt trên các hàm F1, F4 và F5. Theo Bảng 2, ở 20 chiều, PA3 đạt giá trị tối ưu 3.81E-167 trên F1 và 2.20E-112 trên F4, trong khi PA4 đạt 5.91E-123 trên F4. Ở 70 chiều (Bảng 3), PA3 và PA4 tiếp tục vượt trội với F1 (PA3: 3.18E-160, PA4: 5.06E-162). Các phương án này đạt giá trị gần tối ưu trên tất cả kích thước chiều, nhờ sự kết hợp giữa khả năng thăm dò toàn cục của APO và khai thác cục bộ của ARO.

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

So với APO và ARO riêng lẻ, các phương án kết hợp cho thấy độ chính xác và tốc độ hội tụ vượt trội, đặc biệt trên F1, F4 và F5. Ở 20 chiều, APO chỉ đạt 7.74E-09 trên F1, trong khi PA3 đạt 3.81E-167 (Bảng 2). Tương tự, ở 70 chiều, ARO đạt 9.92E-54 trên F5, nhưng PA3 đạt 6.47E-162 (Bảng 3). Hình 2 và Hình 3 cũng cho thấy PA3 và PA4 hội tụ nhanh hơn, đạt giá trị gần tối ưu sau 400-600 vòng lặp, nhờ khả năng tương tác động giữa APO và ARO trong các chiến lược luân phiên và song song.

4.4.3. So sánh với các thuật toán APO và ARO đơn lẻ

So với APO và ARO riêng lẻ, các phương án kết hợp cho thấy độ chính xác và tốc độ hội tụ vượt trội, đặc biệt trên F1, F4 và F5. Ở 20 chiều, APO chỉ đạt 7.74E-09 trên F1, trong khi PA3 đạt 3.81E-167 (Bảng 2). Tương tự, ở 70 chiều, ARO đạt 9.92E-54 trên F5, nhưng PA3 đạt 6.47E-162 (Bảng 3). Hình 2 và Hình 3 cũng cho thấy PA3 và PA4 hội tụ nhanh hơn, đạt giá trị gần tối ưu sau 400-600 vòng lặp, nhờ khả năng tương tác giữa APO và ARO trong các chiến lược luân phiên và song song.

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

Các phương án kết hợp, đặc biệt PA5, duy trì tốt sự đa dạng trong quần thể, giúp thuật toán tránh hội tụ sớm và không mắc kẹt tại các cực trị cục bộ. Ở 20 chiều, PA5 đạt giá trị ổn định 1.76E-39 trên F3 và 3.69E-90 trên F5 (Bảng 2). Cơ chế chia sẻ cá thể trong PA5 và sự luân phiên động trong PA3, PA4 giúp tăng cường tính đa dạng, đặc biệt quan trọng với các bài toán đa đỉnh như F5 và F6.

Nhìn chung, các phương án kết hợp APO-ARO, đặc biệt là PA3, PA4 và PA5, là những phương pháp tối ưu hóa mạnh mẽ, có khả năng cân bằng giữa khai thác và khám phá. Kết quả thực nghiệm cho thấy các phương án này hiệu quả và đáng tin cậy cho các bài toán tối ưu hóa đa chiều phức tạp, vượt trội hơn APO và ARO đơn lẻ về tốc độ hội tụ, độ chính xác và khả

năng xử lý cực trị cục bộ. Điều này cho thấy tiềm năng áp dụng hiệu quả vào các bài toán tối ưu hóa đa chiều phức tạp.

5. Kết luận và phương hướng phát triển

Nghiên cứu đã đề xuất năm chiến lược lai giữa APO và ARO, gồm các mô hình tuần tự, luân phiên và song song có chia sẻ cá thể, nhằm kết hợp hiệu quả khả năng thăm dò của APO và khai thác của ARO. Thực nghiệm trên sáu hàm benchmark cho thấy các mô hình lai, đặc biệt PA3–PA5, cải thiện rõ rệt về tốc độ hội tụ, độ chính xác và khả năng tránh cực trị cục bộ so với các thuật toán đơn.

Sự phối hợp giữa APO và ARO giúp cân bằng giữa thăm dò và khai thác, từ đó nâng cao hiệu suất trên các bài toán tối ưu hóa phức tạp và đa đỉnh. Tuy nhiên, các mô hình chưa tích hợp cơ chế tự điều chỉnh tham số, chỉ mới thử nghiệm trên hàm chuẩn và một số mô hình như PA5 còn tiêu tốn tài nguyên tính toán.

Hướng phát triển tiếp theo gồm: (1) xây dựng cơ chế tự thích nghi theo bài toán; (2) tích hợp với học sâu để tăng khả năng dự đoán; và (3) ứng dụng vào các bài toán thực tế như lập lịch, điều phối năng lượng, hoặc quản lý chuỗi cung ứng trong hệ thống thông minh.

Tài liệu tham khảo

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

Alorf, A. (2023). A survey of recently developed metaheuristics and their comparative analysis. *Engineering Applications of Artificial Intelligence, 117*(A), 105622. https://doi.org/10.1016/j.engappai.2022.105622

Holland, J. H. (1992). Genetic Algorithms. Scientific American. *Scientific American Magazine*, *267*(1), 66–72. https://doi.org/10.1038/scientificamerican0792-66

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

Kennedy, J., & Eberhart, R. (1995). Particle Swarm Optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, 4, 1942–1948. https://doi.org/10.1109/ICNN.1995.488968

Layeb, A. (2022). New hard benchmark functions for global optimization. *arXiv - Neural and Evolutionary Computing, 22*(2). https://doi.org/10.48550/arXiv.2202.04606

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

Tomar, V., Bansal, M., & Singh, P. (2023). Metaheuristic algorithms for optimization: A brief review. *International Conference on Recent Advances in Science and Engineering, 59*(1), 238. https://doi.org/10.3390/engproc2023059238

Wang, L., Cao, Q., Zhang, Z., Mirjalili, S., & Zhao, W. (2022). Artificial rabbits optimization: A new bio-inspired meta-heuristic algorithm for solving engineering optimization problems. *Engineering Applications of Artificial Intelligence, 114*, 105082. https://doi.org/10.1016/j.engappai.2022.105082

Wang, W.-C., Tian, W.-C., Xu, D.-M., & Zang, H.-F. (2024). Arctic puffin optimization: A bio-inspired metaheuristic algorithm for solving engineering design optimization. *Advances in Engineering Software, 195*, 103694. https://doi.org/10.1016/j.advengsoft.2024.103694

Yang, X.-S. (2010). Firefly algorithm, stochastic test functions and design optimisation. *International Journal of Bio-Inspired Computation, 2*(2), 78–84. https://doi.org/10.1504/IJBIC.2010.032124