Kết hợp song song thuật toán tối ưu đa vũ trụ và tối ưu bầy đàn với cơ chế trao đổi thông tin định kỳ và tìm kiếm cục bộ Hill Climbing

Nguyễn Thế Hữu[1, *], Lê Thị Bảo Trân[2]

[1]Trường Đại học Công Thương, Thành phố Hồ Chí Minh, Việt Nam
[2]Trường Đại học Ngoại ngữ – Tin học Thành phố Hồ Chí Minh, Việt Nam
[*]huunthe@huit.edu.vn

Tóm tắt

Nghiên cứu đề xuất phương pháp tối ưu hóa lai song song giữa thuật toán tối ưu đa vũ trụ và tối ưu bầy đàn, tích hợp cơ chế trao đổi thông tin định kỳ và tìm kiếm cục bộ Hill Climbing nhằm nâng cao hiệu quả tối ưu. Hai thuật toán được thực thi trên các quần thể độc lập, định kỳ trao đổi nghiệm tốt nhất để kết hợp khả năng khám phá toàn cục và khai thác cục bộ, trong khi Hill Climbing tinh chỉnh nghiệm nhằm cải thiện độ chính xác và tốc độ hội tụ. Phương pháp được đánh giá trên sáu hàm benchmark tiêu chuẩn và so sánh với hai thuật toán truyền thống. Kết quả cho thấy, phương pháp đề xuất đạt chất lượng nghiệm tốt hơn, đặc biệt đối với các bài toán có nhiều cực trị cục bộ, nhờ khả năng hội tụ nhanh và giảm nguy cơ mắc kẹt tại cực trị cục bộ. Phương pháp có tiềm năng ứng dụng trong các bài toán tối ưu hóa phức tạp đòi hỏi sự cân bằng giữa khám phá toàn cục, khai thác cục bộ và tinh chỉnh nghiệm.

Nhận 25/01/2026
Được duyệt 23/02/2026
Công bố 28/07/2026

Từ khóa

Tối ưu đa vũ trụ; tối ưu bầy đàn; tối ưu song song; tìm kiếm cục bộ; Hill Climbing.

1 Đặt vấn đề

Tối ưu hóa là một lĩnh vực quan trọng trong khoa học tính toán và trí tuệ nhân tạo, được ứng dụng rộng rãi trong nhiều lĩnh vực như tối ưu hóa tham số, điều khiển hệ thống, học máy và thiết kế kỹ thuật [1]. Trong đó, các thuật toán tối ưu hóa quần thể, lấy cảm hứng từ tự nhiên, đã chứng minh hiệu quả trong việc giải quyết các bài toán không tuyến tính và có nhiều cực trị cục bộ. Hai trong số các thuật toán phổ biến nhất là tối ưu bầy đàn (Particle Swarm Optimization – PSO) [2] và tối ưu đa vũ trụ (Multi-Verse Optimizer – MVO) [3], mỗi thuật toán có những ưu điểm và hạn chế riêng.

PSO, được Kennedy và Eberhart giới thiệu vào năm 1995, mô phỏng hành vi tìm kiếm thức ăn của bầy chim, trong đó mỗi phần tử (particle) di chuyển dựa trên vị trí tốt nhất của chính nó và toàn bộ quần thể. Thuật toán này có ưu điểm là hội tụ nhanh và khai thác cục bộ tốt, nhưng dễ bị kẹt vào cực trị cục bộ khi không gian tìm kiếm có nhiều cực trị. Trong khi đó, MVO, được đề xuất bởi Mirjalili vào năm 2016, dựa trên mô hình đa vũ trụ trong vật lý thiên văn, trong đó các vũ trụ có mật độ năng lượng thấp hơn sẽ thu hút các vũ trụ khác thông qua cơ chế lỗ sâu. MVO thể hiện khả năng khám phá toàn cục tốt, giúp tránh bị kẹt cục bộ, nhưng hiệu suất khai thác nghiệm cục bộ chưa cao.

Nhằm tận dụng thế mạnh của cả hai thuật toán, nghiên cứu này đề xuất một phương pháp kết hợp song song giữa MVO và PSO, trong đó cả hai thuật toán chạy đồng thời trên hai quần thể riêng biệt và trao đổi thông tin định kỳ sau mỗi K vòng lặp để tăng cường

khả năng tối ưu hóa [4]. Bên cạnh đó, Hill Climbing (HC) – một thuật toán tìm kiếm cục bộ đơn giản nhưng hiệu quả – được tích hợp để tinh chỉnh nghiệm tốt nhất nhằm nâng cao độ chính xác và tốc độ hội tụ [5]. Sự kết hợp này giúp MVO cải thiện khai thác cục bộ, PSO tận dụng thông tin từ MVO để mở rộng tìm kiếm, và HC đảm bảo thuật toán không bị kẹt tại các nghiệm chưa tối ưu.

Để đánh giá hiệu suất của phương pháp đề xuất, nghiên cứu này thực hiện thực nghiệm trên 6 hàm benchmark tiêu chuẩn gồm Sphere, Rastrigin, Rosenbrock, Ackley, Griewank và Zakharov. Kết quả so sánh với MVO, PSO và MVO-PSO không có HC cho thấy phương pháp đề xuất đạt giá trị hàm mục tiêu tốt hơn, hội tụ nhanh hơn và ít bị kẹt ở cực trị cục bộ hơn.

2 Tổng quan nghiên cứu

Xu hướng nghiên cứu hiện nay trong lĩnh vực tối ưu hóa là kết hợp (hybridization) các siêu thuật toán tìm kiếm nhằm tận dụng thế mạnh của từng phương pháp để nâng cao hiệu suất tìm kiếm và khả năng hội tụ. Trong đó, lai ghép PSO và MVO đã thu hút nhiều sự quan tâm nhờ khả năng bổ trợ lẫn nhau: PSO có hiệu quả trong khai thác cục bộ (exploitation), trong khi MVO có khả năng khám phá toàn cục (exploration) mạnh mẽ. Tuy nhiên, phần lớn các nghiên cứu trước đây mới chỉ tập trung vào việc kết hợp PSO-MVO theo cách truyền thống, chưa tối ưu hóa quá trình tìm kiếm cục bộ để tinh chỉnh nghiệm một cách hiệu quả.

2.1 Xu hướng nghiên cứu về kết hợp PSO – MVO

Một nghiên cứu tiêu biểu về thuật toán lai PSO-MVO đã đề xuất kết hợp hai cơ chế tìm kiếm của PSO và MVO nhằm nâng cao hiệu quả giải các bài toán tối ưu hóa số và bài toán phân phối công suất phản kháng tối ưu trong hệ thống điện [6]. Kết quả thực nghiệm cho thấy, phương pháp lai này đạt hiệu suất tốt hơn so với khi áp dụng riêng lẻ từng thuật toán. Tuy nhiên, nghiên cứu chưa xem xét cơ chế tinh chỉnh nghiệm sau khi thuật toán hội tụ.

Trong một hướng nghiên cứu khác, mô hình PSO-MVO được áp dụng cho bài toán ước lượng công sức phần mềm trong lĩnh vực chất lượng phần mềm [7]. Kết quả thực nghiệm cho thấy phương pháp này cải thiện khả năng tìm nghiệm tối ưu so với các thuật toán đơn lẻ, đặc biệt trong các bài toán có không gian tìm kiếm lớn. Tuy nhiên, phương pháp vẫn chưa tích hợp cơ chế tìm kiếm cục bộ nhằm cải thiện nghiệm sau giai đoạn hội tụ.

Nghiên cứu gần đây đã đề xuất mô hình kết hợp song song giữa MVO và PSO nhằm tận dụng khả năng khám phá toàn cục của MVO và khả năng khai thác cục bộ của PSO [8]. Kết quả thực nghiệm trên các hàm benchmark như Sphere, Rastrigin và Rosenbrock cho thấy phương pháp lai đạt hiệu suất tốt hơn so với khi sử dụng riêng lẻ từng thuật toán.

Nhìn chung, các nghiên cứu trên cho thấy việc kết hợp PSO và MVO có thể cải thiện đáng kể hiệu quả tìm kiếm nghiệm tối ưu. Tuy nhiên, phần lớn các phương pháp hiện có chưa chú trọng đến cơ chế tìm kiếm cục bộ sau hội tụ, điều này có thể khiến thuật toán dừng lại tại các nghiệm chưa thực sự tối ưu.

2.2 Xu hướng nghiên cứu về thuật toán lai ghép

Ngoài sự kết hợp PSO-MVO, nhiều nghiên cứu đã đề xuất lai ghép PSO hoặc MVO với các thuật toán khác nhằm cải thiện hiệu quả tối ưu hóa. Một hướng phổ biến là kết hợp PSO với thuật toán di truyền Genetic Algorithm (GA) để tận dụng khả năng duy trì đa dạng quần thể của GA trong giai đoạn đầu và khả năng khai thác cục bộ của PSO ở giai đoạn sau [9]. Phương pháp này đã được áp dụng trong tối ưu hóa tham số cho các mô hình học sâu và cho thấy hiệu quả trong việc cải thiện quá trình huấn luyện cũng như chất lượng nghiệm của mạng nơ-ron.

Bên cạnh đó, một số nghiên cứu đã lai ghép MVO với các thuật toán khác như Ant Colony Optimization (ACO), trong đó MVO đảm nhiệm vai trò khám phá không gian tìm kiếm toàn cục, còn ACO tập trung tối ưu hóa các lời giải cục bộ trong quá trình tìm kiếm [10]. Phương pháp này đã được áp dụng trong các bài toán tối ưu hóa tuyến đường giao thông và thiết kế mạng.

Gần đây, các nghiên cứu cũng đề xuất kết hợp MVO-PSO với các mô hình học sâu nhằm tối ưu hóa tham số của mô hình [11]. Kết quả cho thấy sự kết hợp giữa khả năng khám phá toàn cục của MVO và khả năng khai thác cục bộ của PSO có thể cải thiện đáng kể độ chính xác của mô hình.

Nhìn chung, các nghiên cứu trên cho thấy việc kết hợp các siêu thuật toán tìm kiếm có thể nâng cao hiệu

quả tìm kiếm nghiệm tối ưu. Tuy nhiên, phần lớn các phương pháp hiện nay vẫn chưa chú trọng đến cơ chế tinh chỉnh nghiệm sau khi hội tụ, dẫn đến nguy cơ thuật toán dừng lại ở các nghiệm chưa thực sự tối ưu.

2.3 Vấn đề đặt ra và hướng phát triển của nghiên cứu

Mặc dù các nghiên cứu trước đây đã chứng minh rằng kết hợp PSO và MVO giúp cải thiện hiệu suất tối ưu hóa, nhưng hầu hết các phương pháp vẫn chưa tối ưu hóa quá trình tinh chỉnh nghiệm sau khi thuật toán hội tụ. Các thuật toán trước đây chủ yếu tập trung vào việc cân bằng giữa khám phá và khai thác mà chưa xem xét đến cải thiện độ chính xác nghiệm sau khi tìm kiếm. Do đó, nghiên cứu này mở rộng mô hình lai ghép PSO-MVO bằng cách tích hợp thuật toán tìm kiếm cục bộ HC nhằm tăng cường khả năng hội tụ và cải thiện chất lượng nghiệm cuối cùng. HC giúp tinh chỉnh nghiệm tốt nhất sau mỗi K vòng lặp, cho phép thuật toán đạt độ chính xác cao hơn. Điểm khác biệt chính của phương pháp này so với các nghiên cứu trước đây gồm:

- Bổ sung giai đoạn tinh chỉnh nghiệm sau hội tụ, giúp thuật toán không chỉ tìm kiếm tốt mà còn đảm bảo nghiệm đạt chất lượng cao nhất;
- Cơ chế trao đổi thông tin linh hoạt giữa PSO và MVO, giúp tận dụng tốt hơn khả năng khai thác và khám phá;
- Áp dụng HC để tránh hội tụ sớm, giúp thuật toán thoát khỏi cực trị cục bộ hiệu quả hơn.

3 Thuật toán đề xuất

3.1 Giới thiệu phương pháp

Nghiên cứu đề xuất một thuật toán lai ghép song song giữa MVO và PSO được đề xuất, tích hợp cơ chế trao đổi thông tin định kỳ và tìm kiếm cục bộ HC để nâng cao hiệu suất tối ưu hóa. Thuật toán kết hợp ba thành phần chính:

Thứ nhất, MVO chịu trách nhiệm khám phá không gian tìm kiếm toàn cục, giúp mở rộng vùng tìm kiếm và tránh hội tụ sớm.

Thứ hai, PSO đảm nhiệm vai trò khai thác cục, giúp tăng cường độ chính xác khi tiến gần đến nghiệm tối ưu.

Thứ ba, HC được áp dụng lên nghiệm tốt nhất sau mỗi K vòng lặp để tinh chỉnh và tối ưu hóa chi tiết hơn.

Phương pháp này giúp tận dụng sức mạnh của cả ba thuật toán: MVO giúp thoát khỏi cực trị cục bộ, PSO giúp hội tụ nhanh hơn, và HC cải thiện độ chính xác nghiệm cuối cùng.

3.2 Thuật toán MVO

MVO là một thuật toán tối ưu hóa lấy cảm hứng từ lý thuyết đa vũ trụ trong vật lý thiên văn. Trong MVO, mỗi nghiệm được xem như một vũ trụ, và sự tương tác giữa các vũ trụ giúp tìm kiếm nghiệm tối ưu [3].

Trong MVO, các vũ trụ di chuyển dựa trên cơ chế lỗ sâu (Wormhole) và tỷ lệ hấp dẫn giữa các vũ trụ. Vị trí của vũ trụ i ở vòng lặp t được cập nhật như sau:

$$X_i^{(t+1)} = X_i^{(t)} + TDR \cdot (UB - LB) \cdot r_1$$

Hoặc

$$X_i^{(t+1)} = X_i^{(t)} + WEP \cdot (X_{best} - X_i^{(t)}) \cdot r_2$$

Trong đó:

- $X_i^{(t)}$ là vị trí của vũ trụ i tại vòng lặp t.
- X_{best} là vũ trụ tốt nhất hiện tại.
- UB, LB là giới hạn trên và dưới của không gian tìm kiếm.
- TDR (Travel Distance Rate) và WEP (Wormhole Existence Probability) là các tham số điều chỉnh quá trình tìm kiếm.
- r_1, r_2 là số ngẫu nhiên trong khoảng [0,1].

3.3 Thuật toán PSO

PSO là một thuật toán tối ưu hóa mô phỏng hành vi bầy đàn. Trong đó, mỗi cá thể (hạt) di chuyển dựa trên kinh nghiệm của chính nó và bầy đàn [2].

Vận tốc và vị trí của mỗi hạt được cập nhật như sau:

$$V_i^{(t+1)} = \omega V_i^{(t)} + c_1 r_1 \left(P_{best,i} - X_i^{(t)}\right) + c_2 r_2 \left(G_{best} - X_i^{(t)}\right)$$

$$X_i^{(t+1)} = X_i^{(t)} + V_i^{(t+1)}$$

Trong đó:

- $X_i^{(t)}$ là vị trí của vũ trụ i tại vòng lặp t.
- $V_i^{(t)}$ là vận tốc c ủa vũ trụ i tại vòng lặp t.
- $P_{best,\,i}$ là vị trí tốt nhất mà hạt i từng đạt được.
- G_{best} là vị trí tốt nhất của cả quần thể.
- ω là hệ số quán tính, giúp điều chỉnh mức độ ảnh hưởng của vận tốc trước đó.
- c_1, c_2 là hệ số học tập, xác định mức độ ảnh hưởng của nghiệm cá nhân và nghiệm toàn cục.
- r_1, r_2 là số ngẫu nhiên trong khoảng [0,1].

3.4 Cơ chế trao đổi thông tin giữa MVO và PSO

MVO và PSO được chạy song song trên hai quần thể riêng biệt. Sau mỗi K vòng lặp, hai thuật toán sẽ trao đổi thông tin về nghiệm tốt nhất, giúp cả hai cập nhật hướng tìm kiếm hiệu quả hơn.

Cơ chế trao đổi thông tin diễn ra như sau:

- Sau mỗi K vòng lặp, tìm nghiệm tốt nhất $X_{MVO\text{ - }best}$ từ MVO và $X_{PSO\text{ - }best}$ từ PSO.
- Cập nhật lại quần thể bằng cách thay thế một phần cá thể của PSO bằng nghiệm tốt nhất từ MVO, và ngược lại.
- Tiếp tục vòng lặp tối ưu hóa với quần thể mới.

Điều này giúp MVO tận dụng được khả năng khai thác của PSO, đồng thời PSO có thể mở rộng không gian tìm kiếm nhờ thông tin từ MVO.

3.5 Bổ sung tìm kiếm cục bộ bằng HC

Sau khi PSO và MVO cập nhật vị trí, thuật toán HC được áp dụng để tinh chỉnh nghiệm tốt nhất.

Lấy nghiệm tốt nhất hiện tại X_{best}.

Tạo lân cận mới bằng cách thêm nhiễu nhỏ:

$$X' = X_{best} + \epsilon,\ \varepsilon \sim U(-\delta,\delta)$$

với δ là bước nhảy nhỏ.

Cập nhật $X_{best} = X'$ nếu cá thể mới có fitness tốt hơn.

Lặp lại quá trình này trong một số vòng nhất định để tìm nghiệm tốt nhất có thể.

Việc bổ sung HC giúp thuật toán tinh chỉnh nghiệm chính xác hơn, giảm nguy cơ mắc kẹt tại cực trị cục bộ.

3.6 Thuật toán tổng quát

Bước 1: khởi tạo hai quần thể PSO và MVO.

Bước 2: chạy cập nhật MVO và PSO song song.

Bước 3: sau mỗi K thực hiện trao đổi thông tin giữa MVO và PSO.

Bước 4: áp dụng HC để tinh chỉnh nghiệm tốt nhất.

Bước 5: kiểm tra điều kiện dừng. Nếu chưa đạt, quay lại Bước 2.

Mã giả của thuật toán MVO-PSO-HC đề xuất

Input: Hàm mục tiêu f(x), số lượng cá thể N, số vòng lặp tối đa T, tham số PSO & MVO
Output: Nghiệm tối ưu X_best
1. Khởi tạo quần thể PSO: positions_pso[i] ∈ [LB, UB] với i = 1, ..., N
2. Khởi tạo quần thể MVO: positions_mvo[i] ∈ [LB, UB] với i = 1, ..., N
3. Đánh giá giá trị fitness của mỗi cá thể trong cả PSO và MVO
4. Xác định cá thể tốt nhất PSO_best và MVO_best
5. Lặp từ t = 1 đến T:
6. Cập nhật quần thể PSO:
    a. Tính toán vận tốc theo công thức PSO
    b. Cập nhật vị trí các hạt theo công thức PSO
    c. Cập nhật nghiệm tốt nhất PSO_best nếu có cá thể tốt hơn
7. Cập nhật quần thể MVO:
    a. Cập nhật vị trí của từng vũ trụ theo công thức MVO
    b. Cập nhật nghiệm tốt nhất MVO_best nếu có vũ trụ tốt hơn
8. Sau mỗi K vòng lặp, thực hiện trao đổi thông tin:
    a. Chọn một phần cá thể PSO để thay thế bằng MVO_best
    b. Chọn một phần cá thể MVO để thay thế bằng PSO_best
9. Tìm kiếm cục bộ bằng HC trên PSO_best và MVO_best:
    a. Tạo lân cận X_new bằng cách thêm nhiễu nhỏ vào X_best
    b. Nếu f(X_new) < f(X_best), cập nhật X_best = X_new
    c. Lặp lại bước này trong một số vòng để tinh chỉnh nghiệm
10. Trả về nghiệm tối ưu X_best

4 Kết quả và thảo luận

4.1 Thiết kế thí nghiệm

Môi trường thực nghiệm

Các thí nghiệm được thực hiện trên máy tính cá nhân chạy hệ điều hành Windows 11, với cấu hình phần cứng gồm bộ vi xử lý Intel Core i7, RAM 16GB. Thuật toán được triển khai bằng Python 3.12, sử dụng các thư viện khoa học như NumPy, SciPy và Matplotlib để hỗ trợ tính toán và trực quan hóa kết quả.

Các bài toán thử nghiệm

Để đánh giá hiệu suất của thuật toán đề xuất, các thử nghiệm được tiến hành trên 6 hàm benchmark phổ biến trong lĩnh vực tối ưu hóa, bao gồm Sphere, Rastrigin, Rosenbrock, Ackley, Griewank, và Zakharov. Các thuật toán được thực thi trong 1 000 vòng lặp, với kích thước quần thể 20 cá thể và không gian tìm kiếm có 10 chiều (dim = 10). Công thức và phạm vi biến của các hàm được trình bày chi tiết trong Bảng 1.

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

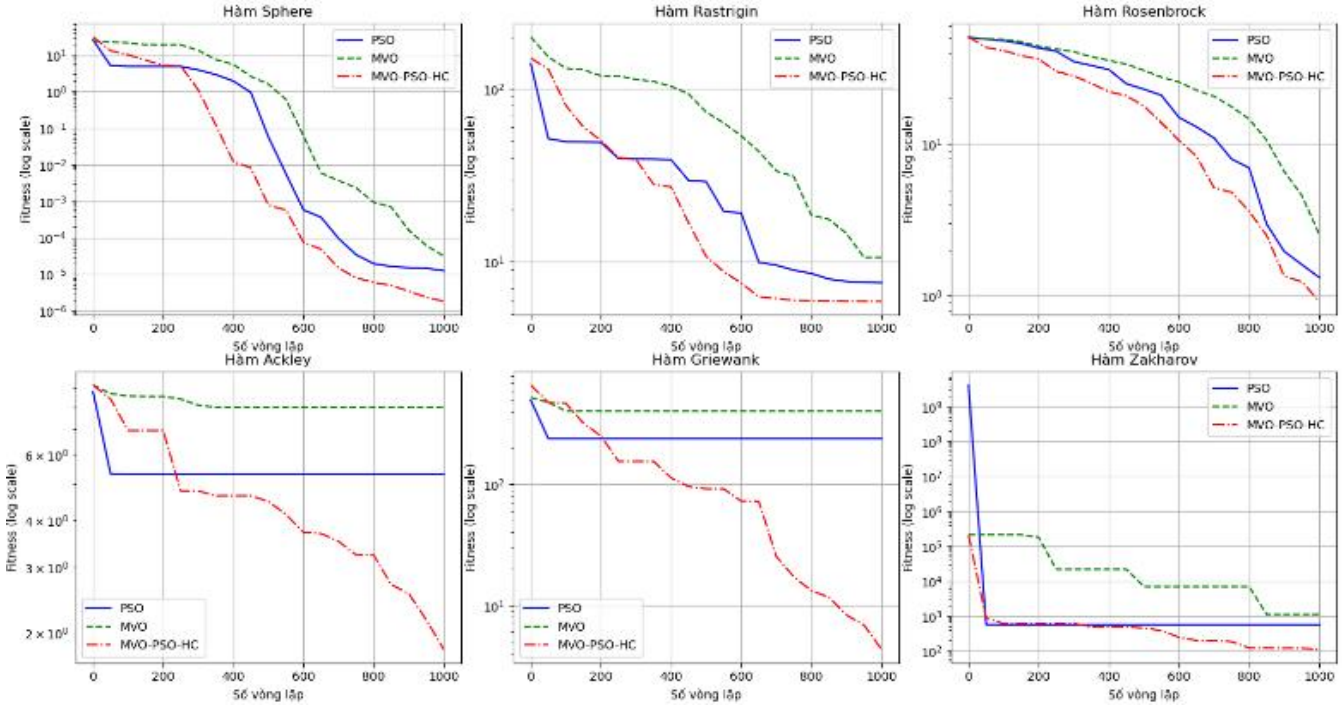
Hàm	Công thức	Phạm vi	Cực trị
Sphere	$\sum x_i^2$	[-2,2]	0
Rastrigin	$10n + \sum \left[x_i^2 - 10cos(2\pi x_i)\right]$	[-5,5]	0
Rosenbrock	$\sum_{i=1}^{n-1} \left[100\left(x_{i+1} - x_i^2\right)^2 + (1 - x_i)^2\right]$	[-5,12;5,12]	0
Ackley	$-20exp\left(-0.2\sqrt{\frac{1}{n}\sum x_i^2}\right) - exp\left(\frac{1}{n}\sum cos2\pi x_i\right) + 20$	[-32,32]	0
Griewank	$1 + \frac{1}{4000}\sum x_i^2 - \prod cos\left(\frac{x_i}{\sqrt{i}}\right)$	[-600,600]	0
Zakharov	$\sum x_i^2 + \left(\sum 0.5ix_i\right)^2 + \left(\sum 0.5ix_i\right)^4$	[-5,10]	0

4.2.1 Phân tích kết quả số liệu: dưới đây là bảng tổng hợp giá trị fitness tốt nhất đạt được sau 1 000 vòng lặp của ba thuật toán (PSO, MVO, MVO-PSO-HC) trên 6 hàm benchmark.

Bảng 2 Giá trị fitness tốt nhất của các thuật toán trên các hàm kiểm chứng

Hàm Benchmark		PSO	MVO	MVO-PSO-HC	Tốt nhất
Sphere		1,25E-05	3,12E-05	1,81E-06	MVO-PSO-HC
Rastrigin		7,576	10,6159	5,9174	MVO-PSO-HC
Rosenbrock		1,3176	2,5261	0,9174	MVO-PSO-HC
Ackley		5,3017	8,0189	1,8028	MVO-PSO-HC
Griewank		239,7289	405,755	4,3031	MVO-PSO-HC
Zakharov		565,8801	1119,312	109,2451	MVO-PSO-HC

Kết quả thực nghiệm cho thấy, thuật toán MVO-PSO-HC luôn đạt giá trị fitness thấp nhất trên cả 6 hàm benchmark, chứng minh hiệu quả của sự kết hợp giữa khám phá toàn cục của MVO, khai thác cục bộ của PSO và tinh chỉnh nghiệm HC. Trên hàm Sphere, MVO-PSO-HC hội tụ nhanh hơn và đạt giá trị chính xác hơn so với các thuật toán riêng lẻ, nhờ vào sự bổ trợ giữa các phương pháp. Đối với hàm Rastrigin, một bài toán có nhiều cực trị cục bộ, MVO tỏ ra yếu hơn PSO do dễ mắc kẹt tại cực trị, nhưng khi kết hợp cả ba phương pháp, thuật toán lại có khả năng vượt qua các bẫy cục bộ tốt hơn và đạt fitness tối ưu nhất.



Hình 1 Xu hướng fitness các hàm theo số vòng lặp

Trên hàm Rosenbrock, vốn có đường tối ưu hẹp, PSO thể hiện khả năng khai thác tốt hơn MVO, nhưng MVO-PSO-HC tiếp tục thể hiện ưu thế khi tận dụng được hướng tìm kiếm của PSO, đồng thời cải thiện hội tụ nhờ khả năng mở rộng không gian tìm kiếm của MVO và tinh chỉnh bằng HC. Đối với hàm Ackley, bài toán có nhiều cực trị và vùng phẳng rộng, MVO bị mắc kẹt trong vùng tìm kiếm phẳng, PSO có hiệu suất tốt hơn, nhưng thuật toán lai tiếp tục thể hiện sự vượt trội khi giảm đáng kể giá trị fitness nhờ vào sự kết hợp ba cơ chế tối ưu hóa.

Hàm Griewank, với cực trị cục bộ xen kẽ vùng phẳng, yêu cầu thuật toán có sự cân bằng giữa khai thác và khám phá. Ở bài toán này, PSO hội tụ tốt hơn MVO nhưng vẫn còn mắc kẹt ở giá trị cao, trong khi MVO-PSO-HC tận dụng tốt cả hai phương pháp để dẫn đến hiệu suất tối ưu cao nhất. Cuối cùng, trên hàm Zakharov, bài toán có độ dốc thay đổi phức tạp, MVO có hiệu suất kém nhất do khó tinh chỉnh nghiệm, PSO hoạt động tốt hơn nhờ khả năng khai thác cục bộ, nhưng MVO-PSO-HC tiếp tục đạt fitness tốt nhất, nhờ vào sự hỗ trợ giữa PSO và MVO trong điều hướng tìm kiếm và sự tinh chỉnh chính xác của HC.

Tóm lại, kết quả cho thấy MVO-PSO-HC không chỉ giúp hội tụ nhanh hơn mà còn cải thiện đáng kể độ chính xác của nghiệm tối ưu, giúp thuật toán tránh bẫy cục bộ và hoạt động hiệu quả hơn trên các bài toán có cấu trúc phức tạp.

4.2.2 Phân tích đồ thị hội tụ

Hình 1 trình bày sáu đồ thị mô tả sự thay đổi của giá trị fitness theo số vòng lặp cho từng hàm benchmark, giúp trực quan hóa hiệu suất của ba thuật toán. Quan sát đồ thị hội tụ của các thuật toán trên sáu hàm benchmark cho thấy thuật toán MVO-PSO-HC duy trì tốc độ giảm fitness ổn định hơn và đạt giá trị tối ưu thấp nhất trong hầu hết các trường hợp.

Trên hàm Sphere, MVO thể hiện tốc độ hội tụ chậm hơn PSO, trong khi thuật toán lai có độ dốc giảm fitness lớn hơn và đạt giá trị tối ưu tốt nhất. Với hàm Rastrigin, cả PSO và MVO đều chững lại ở mức fitness cao, trong khi MVO-PSO-HC tiếp tục giảm fitness rõ rệt sau mỗi giai đoạn trao đổi thông tin, nhờ vào cơ chế tinh chỉnh lân cận của HC.

Đối với hàm Rosenbrock, PSO có xu hướng hội tụ ổn định, thuật toán MVO-PSO-HC duy trì khả năng giảm fitness trong thời gian dài hơn, cho thấy sự điều chỉnh lân cận của HC giúp cải thiện chất lượng nghiệm. Ở hàm Ackley, MVO mắc kẹt tại giá trị fitness cao hơn, trong khi PSO hội tụ nhanh hơn; tuy nhiên, MVO-PSO-HC thể hiện ưu thế rõ rệt với tốc độ giảm fitness

đáng kể sau khoảng 500 vòng lặp, nhờ vào sự phối hợp giữa các cơ chế tối ưu hóa.

Với hàm Griewank, cả PSO và MVO đều gặp khó khăn trong việc giảm fitness do cấu trúc cực trị cục bộ xen kẽ vùng phẳng, trong khi thuật toán lai cho thấy sự suy giảm fitness đột ngột và ổn định hơn, chứng tỏ khả năng tránh mắc kẹt tại các cực trị cục bộ. Cuối cùng, trên hàm Zakharov, MVO ban đầu thể hiện kết quả tốt hơn PSO, nhưng về lâu dài, nhưng MVO-PSO-HC vượt trội hoàn toàn trong giai đoạn sau, nhờ khả năng duy trì xu hướng giảm fitness đến các vòng lặp cuối cùng. Nhìn chung, đồ thị hội tụ chứng minh rằng MVO-PSO-HC có khả năng tìm kiếm và khai thác tốt hơn so với từng thuật toán riêng lẻ, giúp tối ưu hóa nghiệm một cách hiệu quả hơn.

4.2.3 Đánh giá và so sánh với các nghiên cứu trước

So với một nghiên cứu trước đây về mô hình lai ghép PSO-MVO, phương pháp kết hợp chủ yếu tập trung vào việc tận dụng khả năng khám phá của MVO và khả năng khai thác của PSO nhằm cải thiện hiệu quả tìm kiếm toàn cục [8]. Tuy nhiên, các phương pháp này chưa tích hợp cơ chế tìm kiếm cục bộ để tinh chỉnh nghiệm sau khi thuật toán đã tiến gần đến vùng tối ưu.

Trong nghiên cứu này, thuật toán HC được bổ sung như một bước tìm kiếm cục bộ nhằm tinh chỉnh nghiệm tốt nhất trong quá trình lặp. Nhờ đó, thuật toán có thể khai thác hiệu quả vùng lân cận của nghiệm hiện tại, góp phần cải thiện độ chính xác của nghiệm cuối cùng và giảm khả năng mắc kẹt tại cực trị cục bộ. Kết quả thực nghiệm cho thấy, phương pháp MVO-PSO-HC đạt giá trị hàm mục tiêu tốt hơn so với các mô hình không sử dụng HC, cho thấy việc tích hợp HC là một hướng cải tiến hiệu quả so với các phương pháp lai ghép PSO-MVO trước đây.

5 Kết luận

Nghiên cứu này đề xuất một thuật toán lai ghép song song giữa MVO và PSO, kết hợp với HC để nâng cao hiệu suất tối ưu hóa. Phương pháp đề xuất tận dụng khả năng khám phá toàn cục của MVO, khả năng khai thác cục bộ của PSO, và tinh chỉnh nghiệm bằng HC. Đồng thời, cơ chế trao đổi thông tin định kỳ giữa PSO và MVO giúp thuật toán tận dụng tốt nhất thông tin từ cả hai phương pháp, cải thiện khả năng hội tụ và giảm nguy cơ mắc kẹt tại cực trị cục bộ.

Kết quả thực nghiệm trên 6 hàm benchmark phổ biến (Sphere, Rastrigin, Rosenbrock, Ackley, Griewank, Zakharov) cho thấy thuật toán đề xuất luôn đạt giá trị fitness tốt nhất, vượt trội so với PSO và MVO chạy riêng lẻ. MVO-PSO-HC hội tụ nhanh hơn trên các bài toán đơn giản như Sphere, đồng thời có khả năng tránh cực trị cục bộ tốt hơn trên các bài toán phức tạp như Rastrigin, Ackley, Griewank. Đặc biệt, HC đóng vai trò quan trọng trong việc tinh chỉnh nghiệm, giúp cải thiện độ chính xác so với các thuật toán truyền thống.

Kết quả đã chứng minh rằng kết hợp ba cấp độ tối ưu hóa (khám phá, khai thác, tinh chỉnh) giúp cải thiện đáng kể hiệu suất tối ưu hóa, mở ra hướng phát triển mới cho các thuật toán lai ghép trong tương lai.

Trong tương lai có thể mở rộng hướng nghiên cứu về thuật toán bằng cách tích hợp cơ chế điều chỉnh động, cho phép thuật toán tự động thay đổi tần suất trao đổi thông tin giữa MVO và PSO. Ngoài ra, có thể kết hợp thêm các phương pháp Deep Learning hoặc tối ưu hóa thích nghi để mở rộng khả năng ứng dụng trong các bài toán thực tế như tối ưu tham số mạng nơ-ron, lập lịch thông minh và tối ưu hóa hệ thống năng lượng.

Tài liệu tham khảo

1. S. Selvarajan. (2024). A comprehensive study on modern optimization techniques for engineering applications. *Artificial Intelligence Review* 57, 194. https://doi.org/10.1007/s10462-024-10829-9.

2. L. Abualigah. (2025). Particle Swarm Optimization: Advances, Applications, and Experimental Insights. *Computers, Materials and Continua* 82, 1539. https://doi.org/10.32604/cmc.2025.060765.

3. L. Abualigah. (2020). Multi-verse optimizer algorithm: a comprehensive survey of its results, variants, and applications. *Neural Computing and Applications* 32, 12381. https://doi.org/10.1007/s00521-020-04839-1.
4. A. Abdelhafez, G. Luque, E. Alba. (2020). Parallel execution combinatorics with metaheuristics: Comparative study. *Swarm and Evolutionary Computation* 55, 100692. https://doi.org/10.1016/j.swevo.2020.100692.
5. Y. Zhou, Z. Jiang and M. Zhou. (2024). Simulated Hill Climbing Search for the Solutions of k-Vertex Cut Problems. *2024 IEEE 20th International Conference on Automation Science and Engineering (CASE)*, 1729. https://doi.org/10.1109/CASE59546.2024.10711754.
6. P. Jangir, S.A. Parmar, I.N. Trivedi, R.H. Bhesdadiya. (2017). A novel hybrid Particle Swarm Optimizer with multi verse optimizer for global numerical optimization and Optimal Reactive Power Dispatch problem. *Engineering Science and Technology, an International Journal* 20, 570. https://doi.org/10.1016/j.jestch.2016.10.007.
7. E. Nehra. (2018). Investigating Scope of Hybrid Particle Swarm Optimization-Multi Verse Optimizer in Software Quality. *Journal of Software Engineering and Simulation*, Volume 4, Issue 8, pp: 32-38.
8. Nguyễn Thế Hữu, Đinh Nguyễn Trọng Nghĩa (2024). Kết hợp song song thuật toán tối ưu đa vũ trụ và tối ưu bầy đàn cho các bài toán tối ưu. *Tạp chí Công Thương*, Số 20, 380-383.
9. B. Wang, Y. Sun, B. Xue, M. Zhang. (2019). A Hybrid GA-PSO Method for Evolving Architecture and Short Connections of Deep Convolutional Neural Networks. *PRICAI 2019: Trends in Artificial Intelligence* 11672, 650.
10. BenMansour, I. (2023). An Effective Hybrid Ant Colony Optimization for the Knapsack Problem Using Multi-Directional Search. SN COMPUT. SCI. 4, 164. https://doi.org/10.1007/s42979-022-01564-5.
11. A. Gupta, D. Kaushik, M. Garg. (2020). Implementation of deep learning mechanism in big data using hybrid multi-verse optimizer with particle swarm optimization. *ICTACT Journal on Soft Computing* 10(4), 2031. https://doi.org/10.21917/ijsc.2020.0310.

Parallel Hybridization of Multi-Verse Optimizer and Particle Swarm Optimization with Periodic Information Exchange and Local Hill Climbing Search

Nguyen The Huu[1, *], Le Thi Bao Tran[2]

[1]Ho Chi Minh City University of Industry and Trade, Viet Nam
[2]Ho Chi Minh City University of Foreign Languages - Information Technology, Viet Nam
[*]huunthe@huit.edu.vn

Abstract This study proposes a hybrid optimization method that combined the Multi-Verse Optimizer (MVO) and Particle Swarm Optimization (PSO) in a parallel framework, incorporating periodic information exchange and the Hill Climbing local search strategy to enhance optimization performance. The two algorithms were executed on separate populations and periodically exchanged their best solutions, thereby integrating the global exploration capability of MVO with the local exploitation strength of PSO, while Hill Climbing further refined the best solution to improve accuracy and convergence speed. The proposed method was evaluated using six standard benchmark functions and compared with the conventional MVO and PSO algorithms. Experimental results demonstrated that the proposed approach achieved superior objective function values, particularly for multimodal optimization problems, owing to its faster convergence and reduced risk of being trapped in local optima. The proposed method showed strong potentials for solving complex optimization problems that require an effective balance between global exploration, local exploitation, and solution refinement.

Keywords Multi-Verse Optimizer; Particle Swarm Optimization; Parallel optimization; Local search; Hill Climbing.