**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ GIAO THÔNG VẬN TẢI**

**KHOA CÔNG NGHỆ THÔNG TIN**

A white rectangular sign with orange text

Description automatically generated

|  |
| --- |
| **ĐỀ TÀI** |
| **Tìm hiểu thuật toán moth flame optimization(MFO)**  **và bat algorithm (BA)** |

|  |  |
| --- | --- |
| **Giảng viên hướng dẫn:** | **Đỗ Bảo Sơn** |
| **Nhóm sinh viên thực hiện:** | **Nguyễn Vũ Long**  **Lê Thành Công**  **Trần Việt Anh**  **Nguyễn Văn Sâm**  **Nông Thanh Bình** |
| **Lớp:** | **73DCTT23 – Nhóm 8** |

**HÀ NỘI – 2024**

**BẢNG PHÂN CÔNG CÔNG VIỆC**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Họ và tên** | **Chức vụ** | **Mã sinh viên** | **Công việc** | **Khối lượng công việc** |
| Lê Thành Công | Nhóm trưởng | 73DCTT23286 |  |  |
| Nguyễn Vũ Long | Thành viên | 73DCTT22358 |  |  |
| Trần Việt Anh | Thành viên | 73DCTT22442 |  |  |
| Nguyễn Văn Sâm | Thành viên | 73DCTT22166 |  |  |
| Nông Thanh Bình | Thành viên | 73DCTT21001 |  |  |

**NHẬN XÉT CỦA GIẢNG VIÊN**

………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………

[LỜI NÓI ĐẦU 5](#_Toc184758661)

[CHƯƠNG I: MOTH FLAME OPTIMIZATION (MFO) 6](#_Toc184758662)

[1. Giới thiệu và trình bày động lực đề xuất thuật toán 6](#_Toc184758663)

[1.1. Giới thiệu thuật toán tối ưu hóa ngọn lửa bướm đêm (MFO) 6](#_Toc184758664)

[1.2. Động lực đề xuất thuật toán 7](#_Toc184758665)

[1.3. Nguồn càm hứng từ bướm đêm 9](#_Toc184758666)

[2. Nội dung và các bước của thuật toán 12](#_Toc184758667)

[3. Thực nghiệm, chạy thử, đánh giá kết quả của thuật toán 21](#_Toc184758668)

[3.1 Thực nghiệm thuật toán MFO 21](#_Toc184758669)

[3.2 chạy thử thuật toán MFO 22](#_Toc184758670)

[3.3 Đánh giá thuật toán MFO 23](#_Toc184758671)

[4. Ưu nhược điểm của thuật toán Moth flame optimization(MFO) 24](#_Toc184758672)

[4.1. Ưu điểm của MFO 24](#_Toc184758673)

[4.2. Nhược điểm của MFO 25](#_Toc184758674)

[5. Kết Luận 26](#_Toc184758675)

[CHƯƠNG II: BAT ALGORITHM (BA) 28](#_Toc184758676)

[1. Giới thiệu và trình bày động lực đề xuất thuật toán 28](#_Toc184758677)

[1.1. Giới thiệu thuật toán con dơi (BA) 28](#_Toc184758678)

[1.2. Động lực đề xuất thuật toán 29](#_Toc184758679)

[1.3. Nguồn càm hứng từ loài dơi 30](#_Toc184758680)

[2. Nội dung và các bước của thuật toán 34](#_Toc184758681)

[2.1 Thuật toán chuyển động dơi 34](#_Toc184758682)

[2.2 Các biến thể về âm lượng và tốc độ xung 35](#_Toc184758683)

[3. Thực nghiệm, chạy thử, đánh giá kết quả của thuật toán 35](#_Toc184758684)

[3.1 Thực nghiệm thuật toán BA 35](#_Toc184758685)

[3.2 chạy thử thuật toán MFO 36](#_Toc184758686)

[3.3 Đánh giá thuật toán BA 38](#_Toc184758687)

[4. Ưu nhược điểm của thuật toán Bat algorithm (BA) 39](#_Toc184758688)

[4.1. Ưu điểm của BA 39](#_Toc184758689)

[4.2. Nhược điểm của BA 40](#_Toc184758690)

[5. Kết Luận 42](#_Toc184758691)

# LỜI NÓI ĐẦU

Trong kỷ nguyên công nghệ hiện đại, các thuật toán tối ưu hóa đóng vai trò quan trọng trong việc giải quyết các vấn đề phức tạp thuộc nhiều lĩnh vực như trí tuệ nhân tạo, khoa học dữ liệu, kỹ thuật và kinh tế. Hai trong số những thuật toán nổi bật trong mảng này là **Thuật toán tối ưu hóa bướm đêm và ngọn lửa (Moth Flame Optimization - MFO)** và **Thuật toán dơi (Bat Algorithm - BA).**

Thuật toán MFO được lấy cảm hứng từ hành vi điều hướng của bướm đêm dựa vào ánh sáng. Nó mô phỏng cách các bướm đêm duy trì khoảng cách không đổi với nguồn sáng, dẫn đến hiệu quả tìm kiếm lời giải tối ưu trong không gian phức tạp. Tính ổn định, khả năng khai thác và khám phá của thuật toán đã chứng minh tiềm năng trong nhiều bài toán tối ưu hóa phi tuyến.

Trong khi đó, thuật toán BA được phát triển dựa trên khả năng định vị bằng tiếng vang (echolocation) của loài dơi. Với cơ chế khai thác và khám phá kết hợp nhịp nhàng, BA mang đến hiệu quả vượt trội trong việc tìm kiếm lời giải tốt, đặc biệt trong các bài toán có không gian tìm kiếm rộng lớn và không trơn tru.

Nghiên cứu này tập trung phân tích và so sánh hai thuật toán MFO và BA, từ cơ sở lý thuyết, cấu trúc thuật toán cho đến các ứng dụng thực tế. Qua đó, nhóm tác giả hy vọng mang lại cái nhìn toàn diện, góp phần mở rộng ứng dụng của hai thuật toán này trong các bài toán tối ưu hóa đa ngành.

*Nhóm em xin chân thành cảm ơn*

|  |  |
| --- | --- |
|  | **Sinh viên thực hiện**  Nguyễn Vũ Long  Lê Thành Công  Trần Việt Anh  Nguyễn Văn Sâm  Nông Thanh Bình |

# CHƯƠNG I: MOTH FLAME OPTIMIZATION (MFO)

## 1. Giới thiệu và trình bày động lực đề xuất thuật toán

### 1.1. Giới thiệu thuật toán tối ưu hóa ngọn lửa bướm đêm (MFO)

Tối ưu hóa là quá trình tìm kiếm giải pháp tốt nhất cho một vấn đề cụ thể. Khi độ phức tạp của các vấn đề ngày càng tăng, nhu cầu về các phương pháp tối ưu hóa mới cũng trở nên cấp thiết. Các phương pháp tối ưu hóa toán học truyền thống chủ yếu mang tính xác định (deterministic), nhưng chúng thường gặp phải những hạn chế như bẫy cực trị cục bộ (local optima entrapment) và yêu cầu thông tin về đạo hàm hoặc gradient của không gian tìm kiếm. Những hạn chế này làm cho các phương pháp truyền thống trở nên kém hiệu quả khi áp dụng vào các bài toán thực tế có không gian tìm kiếm phức tạp hoặc chưa biết trước.

Để khắc phục, các thuật toán tối ưu hóa lấy cảm hứng từ tự nhiên đã được nghiên cứu và phát triển mạnh mẽ trong vài thập kỷ qua. Những thuật toán này, như Thuật toán Di truyền (GA), Tối ưu hóa Bầy đàn hạt (PSO), và Tối ưu hóa Bầy Kiến (ACO), đã chứng minh hiệu quả vượt trội trong việc tránh bẫy cực trị cục bộ nhờ các thành phần ngẫu nhiên, khả năng tìm kiếm toàn cục và không yêu cầu thông tin gradient. Tuy nhiên, theo định lý "No-Free-Lunch (NFL)", không có một thuật toán nào phù hợp để giải quyết tất cả các bài toán tối ưu hóa. Điều này mở ra cơ hội nghiên cứu và phát triển các thuật toán tối ưu hóa mới nhằm đáp ứng các thách thức đa dạng trong thực tế.

Trong bối cảnh đó, thuật toán tối ưu hóa bướm đêm và ngọn lửa (Moth Flame Optimization - MFO) ra đời, lấy cảm hứng từ hành vi định hướng của bướm đêm trong tự nhiên. Cụ thể, bướm đêm sử dụng cơ chế định hướng ngang (transverse orientation), duy trì một góc cố định với nguồn sáng để điều hướng. Hành vi này dẫn đến quỹ đạo bay xoắn ốc khi bướm đêm tiếp cận các nguồn sáng nhân tạo. Thuật toán MFO mô phỏng hành vi này thông qua một mô hình toán học, trong đó:

* Các bướm đêm đại diện cho các lời giải khả thi trong không gian tìm kiếm.
* Ngọn lửa tượng trưng cho các lời giải tối ưu tiềm năng.

Bằng cách mô phỏng quỹ đạo xoắn ốc, MFO duy trì sự cân bằng giữa khai thác (exploitation) các khu vực tiềm năng và khám phá (exploration) các khu vực mới trong không gian tìm kiếm. Điều này giúp thuật toán tránh được bẫy cực trị cục bộ và cải thiện hiệu quả tối ưu hóa.

Nghiên cứu này sẽ trình bày cấu trúc thuật toán MFO, cơ chế hoạt động và các ứng dụng của nó trong việc giải quyết các bài toán tối ưu hóa thực tế. Thuật toán đã chứng minh hiệu quả vượt trội trong các bài toán tiêu chuẩn và các bài toán thiết kế kỹ thuật phức tạp như thiết kế kết cấu, chế tạo cơ khí, và tối ưu hóa hệ thống.

### 1.2. Động lực đề xuất thuật toán

Động lực để đề xuất thuật toán tối ưu hóa bướm đêm (MFO) xuất phát từ nhu cầu ngày càng cao trong việc giải quyết các bài toán tối ưu hóa phức tạp trong nhiều lĩnh vực, từ kỹ thuật, khoa học tự nhiên đến quản lý và kinh doanh. Tối ưu hóa là một phần không thể thiếu trong việc tìm kiếm giải pháp tốt nhất, nhưng các phương pháp truyền thống thường gặp phải những hạn chế lớn khi đối mặt với các bài toán có không gian tìm kiếm phức tạp hoặc phi tuyến.

Các phương pháp tối ưu hóa toán học, vốn được coi là nền tảng ban đầu, hoạt động chủ yếu dựa trên các nguyên tắc xác định. Tuy nhiên, những phương pháp này thường gặp phải vấn đề lớn là bẫy cực trị cục bộ, nghĩa là chúng dễ dàng bị mắc kẹt trong các giải pháp không phải là tối ưu toàn cục. Hơn nữa, các thuật toán toán học như những phương pháp dựa trên gradient yêu cầu thông tin đạo hàm của không gian tìm kiếm, điều này không phải lúc nào cũng khả thi đối với các bài toán thực tế có cấu trúc phức tạp hoặc thiếu thông tin về mô hình. Những hạn chế này không chỉ làm giảm hiệu quả của các phương pháp truyền thống mà còn làm tăng đáng kể chi phí tính toán và thời gian xử lý trong các bài toán quy mô lớn hoặc phi tuyến.

Sự ra đời của các thuật toán lấy cảm hứng từ tự nhiên đã đánh dấu một bước ngoặt quan trọng trong lĩnh vực tối ưu hóa. Những thuật toán như Thuật toán Di truyền (Genetic Algorithm - GA), Tối ưu hóa Bầy đàn hạt (Particle Swarm Optimization - PSO), và Tối ưu hóa Bầy Kiến (Ant Colony Optimization - ACO) đã cho thấy hiệu quả vượt trội nhờ khả năng tích hợp các yếu tố ngẫu nhiên và cách tiếp cận dựa trên quần thể. Những thuật toán này có thể khám phá không gian tìm kiếm một cách linh hoạt hơn, tránh được bẫy cực trị cục bộ và không yêu cầu thông tin gradient. Tuy nhiên, các thuật toán này không phải là giải pháp hoàn hảo cho tất cả các bài toán. Theo định lý "No-Free-Lunch" (NFL), không có thuật toán nào có thể giải quyết hiệu quả tất cả các bài toán tối ưu hóa. Điều này có nghĩa là, mỗi bài toán cụ thể có thể yêu cầu một thuật toán được thiết kế riêng để tận dụng các đặc điểm của không gian tìm kiếm hoặc bản chất của bài toán đó.

Trong bối cảnh này, việc nghiên cứu và phát triển các thuật toán mới để đáp ứng các thách thức chưa được giải quyết trở thành một yêu cầu cấp thiết. Một trong những nguồn cảm hứng đầy hứa hẹn là hành vi tự nhiên của động vật, vốn đã tiến hóa qua hàng triệu năm để tối ưu hóa khả năng sinh tồn và tìm kiếm tài nguyên. Đặc biệt, hành vi định hướng của bướm đêm trong tự nhiên đã trở thành động lực chính cho việc phát triển thuật toán tối ưu hóa bướm đêm và ngọn lửa (MFO).

Bướm đêm sử dụng một cơ chế điều hướng đặc biệt gọi là định hướng ngang (transverse orientation), trong đó chúng duy trì một góc cố định với nguồn sáng. Khi áp dụng vào các nguồn sáng nhân tạo, hành vi này dẫn đến quỹ đạo bay xoắn ốc độc đáo. Quá trình này có thể được xem như một dạng tối ưu hóa tự nhiên, nơi các bướm đêm không ngừng tiếp cận các nguồn sáng, tương tự như cách các giải pháp trong không gian tìm kiếm tiến gần đến điểm tối ưu toàn cục.

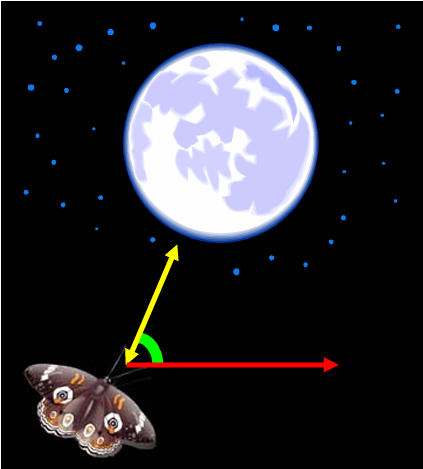
Thuật toán MFO được phát triển dựa trên mô hình toán học mô phỏng hành vi này, trong đó các bướm đêm đại diện cho các giải pháp khả thi và ngọn lửa tượng trưng cho các giải pháp tối ưu tiềm năng. Hành vi bay xoắn ốc không chỉ tạo ra sự cân bằng giữa khai thác các vùng tiềm năng và khám phá các vùng chưa được kiểm tra trong không gian tìm kiếm, mà còn giúp thuật toán tránh được bẫy cực trị cục bộ. Điều này đặc biệt quan trọng đối với các bài toán phi tuyến, phức tạp và có nhiều ràng buộc trong thực tế.

Động lực chính để đề xuất thuật toán MFO là mong muốn bổ sung một công cụ tối ưu hóa mạnh mẽ và linh hoạt vào kho tàng các phương pháp hiện có, với khả năng giải quyết hiệu quả hơn các bài toán chưa được giải quyết hoặc các bài toán mà các thuật toán hiện tại chưa tối ưu. Với cách tiếp cận mới dựa trên hành vi tự nhiên độc đáo, MFO không chỉ có tiềm năng cải thiện hiệu suất tối ưu hóa mà còn mở ra các hướng nghiên cứu mới trong lĩnh vực tối ưu hóa dựa trên sinh học và hành vi động vật.

### 1.3. Nguồn càm hứng từ bướm đêm

Nguồn cảm hứng từ bướm đêm cho thuật toán tối ưu hóa bướm đêm (MFO) đến từ một hiện tượng tự nhiên thú vị liên quan đến cơ chế định hướng của loài bướm đêm. Trong tự nhiên, bướm đêm sử dụng một phương pháp điều hướng gọi là định hướng ngang (transverse orientation). Đây là một chiến lược mà chúng duy trì một góc cố định với nguồn sáng, chẳng hạn như ánh sáng từ mặt trăng, để định vị và điều hướng trong không gian.

Phương pháp định hướng này hoạt động hiệu quả trong môi trường tự nhiên, nơi nguồn sáng, như ánh sáng mặt trăng, thường cách xa bướm đêm và có thể được coi là gần như song song. Tuy nhiên, khi bướm đêm tiếp xúc với các nguồn sáng nhân tạo, chẳng hạn như đèn điện, hành vi này dẫn đến một kết quả rất khác biệt. Do nguồn sáng nhân tạo gần hơn nhiều, các tia sáng không còn song song mà tỏa ra theo mọi hướng. Khi bướm đêm cố gắng duy trì góc cố định với nguồn sáng, quỹ đạo của chúng chuyển thành các vòng xoắn ốc, liên tục quay quanh và ngày càng tiếp cận nguồn sáng.

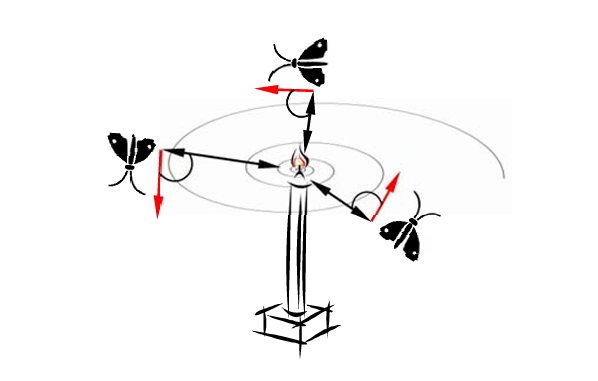


*Hình 1 : Bướm đêm di chuyển theo hướng ngang*

Hành vi này, mặc dù không hiệu quả trong việc điều hướng ở môi trường nhân tạo, lại là một ví dụ nổi bật về cách các sinh vật trong tự nhiên điều chỉnh hành vi để thích nghi với môi trường. Đặc điểm quỹ đạo xoắn ốc của bướm đêm đã gợi ý một cách tiếp cận độc đáo để mô phỏng trong lĩnh vực tối ưu hóa. Trong bài toán tối ưu hóa, các giải pháp khả thi có thể được xem như các bướm đêm, trong khi các lời giải tối ưu tiềm năng tương tự như nguồn sáng (ngọn lửa).

Nguồn cảm hứng chính của thuật toán MFO đến từ việc mô phỏng quỹ đạo bay này trong không gian tìm kiếm. Khi áp dụng vào bài toán tối ưu hóa, các bướm đêm đại diện cho các giải pháp trong không gian tìm kiếm, và hành vi xoắn ốc của chúng được sử dụng như một cơ chế tìm kiếm để khám phá các vùng tiềm năng. Ngọn lửa, tượng trưng cho các lời giải tốt nhất, sẽ liên tục hướng dẫn các bướm đêm tiếp cận gần hơn với tối ưu toàn cục.

Cách thức bay xoắn ốc của bướm đêm không chỉ mang lại sự đa dạng trong hành vi tìm kiếm mà còn tạo ra sự cân bằng giữa khai thác (exploitation) và khám phá (exploration) không gian tìm kiếm. Trong khi khai thác giúp tập trung vào việc cải thiện các giải pháp gần tối ưu, khám phá giúp mở rộng không gian tìm kiếm và tránh bị mắc kẹt trong các cực trị cục bộ.



*Hình 2 : Đường bay xoắn ốc xung quanh các nguồn sáng gần*

Ngoài ra, hành vi của bướm đêm trong tự nhiên còn cung cấp một cái nhìn sâu sắc về cách các chiến lược đơn giản, khi được áp dụng đúng cách, có thể dẫn đến các giải pháp hiệu quả trong việc xử lý các bài toán phức tạp. Cảm hứng từ bướm đêm nhấn mạnh vai trò của tự nhiên như một nguồn tài nguyên phong phú để phát triển các thuật toán tối ưu hóa mới, dựa trên việc học hỏi từ những chiến lược đã được tự nhiên tối ưu hóa qua hàng triệu năm tiến hóa.

Nhờ sự kết hợp giữa cơ chế định hướng tự nhiên và mô hình toán học, thuật toán MFO không chỉ mô phỏng thành công hành vi của bướm đêm mà còn mang lại một công cụ mạnh mẽ để giải quyết các bài toán tối ưu hóa. Nguồn cảm hứng từ bướm đêm không chỉ dừng lại ở việc tái hiện hành vi mà còn nhấn mạnh cách các hiện tượng tự nhiên có thể được khai thác để tạo ra những phương pháp mới trong khoa học tính toán.

## 2. Nội dung và các bước của thuật toán

Trong thuật toán MFO được đề xuất, chúng tôi giả định rằng các giải pháp ứng cử viên là bướm đêm và các biến của vấn đề là vị trí của bướm đêm trong không gian. Do đó, bướm đêm có thể bay trong không gian 1-D, 2-D, 3-D hoặc siêu chiều với việc thay đổi vectơ vị trí của chúng. Vì thuật toán MFO là một thuật toán dựa trên dân số, chúng tôi biểu diễn tập hợp các con bướm đêm trong một ma trận như sau:

Ảnh có chứa văn bản, biên lai, Phông chữ, ảnh chụp màn hình

Mô tả được tạo tự động

* Trong đó n là số bướm đêm và d là số biến (chiều).

Đối với tất cả các loài bướm đêm, chúng tôi cũng giả định rằng có một mảng để lưu trữ các giá trị thể dục tương ứng như sau:

Ảnh có chứa văn bản, Phông chữ, thuật in máy, thiết kế

Mô tả được tạo tự động

* Trong đó n là số bướm đêm.

Lưu ý rằng giá trị phù hợp là giá trị trả về của hàm phù hợp (mục tiêu) cho mỗi bướm đêm. Vectơ vị trí (ví dụ: hàng đầu tiên trong ma trận M) của mỗi bướm đêm được chuyển đến hàm phù hợp và đầu ra của hàm phù hợp được gán cho bướm đêm tương ứng làm hàm phù hợp của nó (ví dụ như OM1 trong ma trận OM).

Một thành phần quan trọng khác trong thuật toán được đề xuất là ngọn lửa. Xem xét một ma trận tương tự như ma trận bướm đêm như sau:

Ảnh có chứa văn bản, biên lai, biểu đồ, Phông chữ

Mô tả được tạo tự động

* Trong đó n là số bướm đêm và d là số biến (chiều).

Có thể thấy trong Phương trình rằng kích thước của mảng M và F bằng nhau. Đối với ngọn lửa, chúng ta cũng giả định rằng có một mảng để lưu trữ các giá trị phù hợp tương ứng như sau:

Ảnh có chứa văn bản, Phông chữ, màu trắng, thiết kế

Mô tả được tạo tự động

* Trong đó n là số bướm đêm.

Cần lưu ý rằng cả bướm đêm và ngọn lửa đều là các lời giải trong thuật toán. Sự khác biệt giữa chúng nằm ở cách chúng được xử lý và cập nhật trong mỗi vòng lặp. Bướm đêm là các tác nhân tìm kiếm thực tế, di chuyển xung quanh không gian tìm kiếm, trong khi ngọn lửa đại diện cho vị trí tốt nhất của các bướm đêm mà chúng đạt được cho đến thời điểm đó. Nói cách khác, ngọn lửa có thể được coi là các cột mốc hoặc điểm mà các bướm đêm đánh dấu khi chúng tìm kiếm trong không gian tìm kiếm. Vì vậy, mỗi bướm đêm tìm kiếm xung quanh một ngọn lửa (cột mốc) và sẽ cập nhật lại ngọn lửa nếu tìm được một giải pháp tốt hơn. Với cơ chế này, mỗi bướm đêm không bao giờ mất đi lời giải tốt nhất của nó.

Thuật toán MFO là ba bộ gần đúng tối ưu toàn cầu của các bài toán tối ưu hóa và được định nghĩa như sau:

*MFO = (I, P, T)*

Ilà một hàm tạo ra một quần thể bướm đêm ngẫu nhiên và các giá trị phù hợp tương ứng. Mô hình phương pháp của hàm này như sau*:*

*I : {M, OM}*

Hàm P, là hàm chính, di chuyển bướm đêm xung quanh không gian tìm kiếm. Hàm này nhận được ma trận M và cuối cùng trả về ma trận cập nhật của nó.

*P : M M*

Hàm T trả về true nếu tiêu chí chấm dứt được thỏa mãn và false nếu tiêu chí chấm dứt không được thỏa mãn:

*T : M {TRUE , FALSE}*

Với I, P và T, khung chung của thuật toán MFO được định nghĩa như sau:

M=I();

***while*** T(M) is equal to false

M=P(M);

***End***

Hàm I phải tạo ra các giải pháp ban đầu và tính toán các giá trị hàm mục tiêu. Bất kỳ phân phối ngẫu nhiên nào cũng có thể được sử dụng trong hàm này. Những gì chúng tôi thực hiện như sau:

**for** i = 1 : n

**for** j= 1 : d 8

M(i,j)=(ub(i)-lb(i))\* rand()+lb(i);

**end**

**end**

OM=FitnessFunction(M);

Như có thể thấy, có hai mảng khác được gọi là ub và lb. Các ma trận này xác định giới hạn trên và dưới của các biến như sau:

**

Trong đó cho biết giới hạn trên của biến thứ i.



Trong đó cho biết giới hạn trên của biến thứ i.

Sau khi khởi tạo, hàm P được chạy lặp đi lặp lại cho đến khi hàm T trả về true. Hàm P là hàm chính di chuyển bướm đêm xung quanh không gian tìm kiếm. Như đã đề cập ở trên, nguồn cảm hứng của thuật toán này là hướng ngang. Để mô hình hóa hành vi này một cách toán học, chúng tôi cập nhật vị trí của mỗi con bướm đêm đối với ngọn lửa bằng cách sử dụng phương trình sau:

 (2.1)

Trong đó Mi biểu thị bướm đêm thứ i, Fj biểu thị ngọn lửa thứ j và S là hàm xoắn ốc.

Chọn xoắn ốc logarit làm cơ chế cập nhật chính của bướm đêm trong bài báo này. Tuy nhiên, bất kỳ loại xoắn ốc nào cũng có thể được sử dụng ở đây tùy thuộc vào các điều kiện sau:

* Điểm ban đầu của Spiral nên bắt đầu từ bướm đêm
* Điểm cuối cùng của xoắn ốc phải là vị trí của ngọn lửa
* Sự dao động của phạm vi xoắn ốc không được vượt quá không gian tìm kiếm

Xem xét những điểm này, xác định xoắn ốc logarit cho thuật toán MFO như sau:

*S( Mj, Fj) = Di . ebt . cos(2t) + Fj (2.2)*

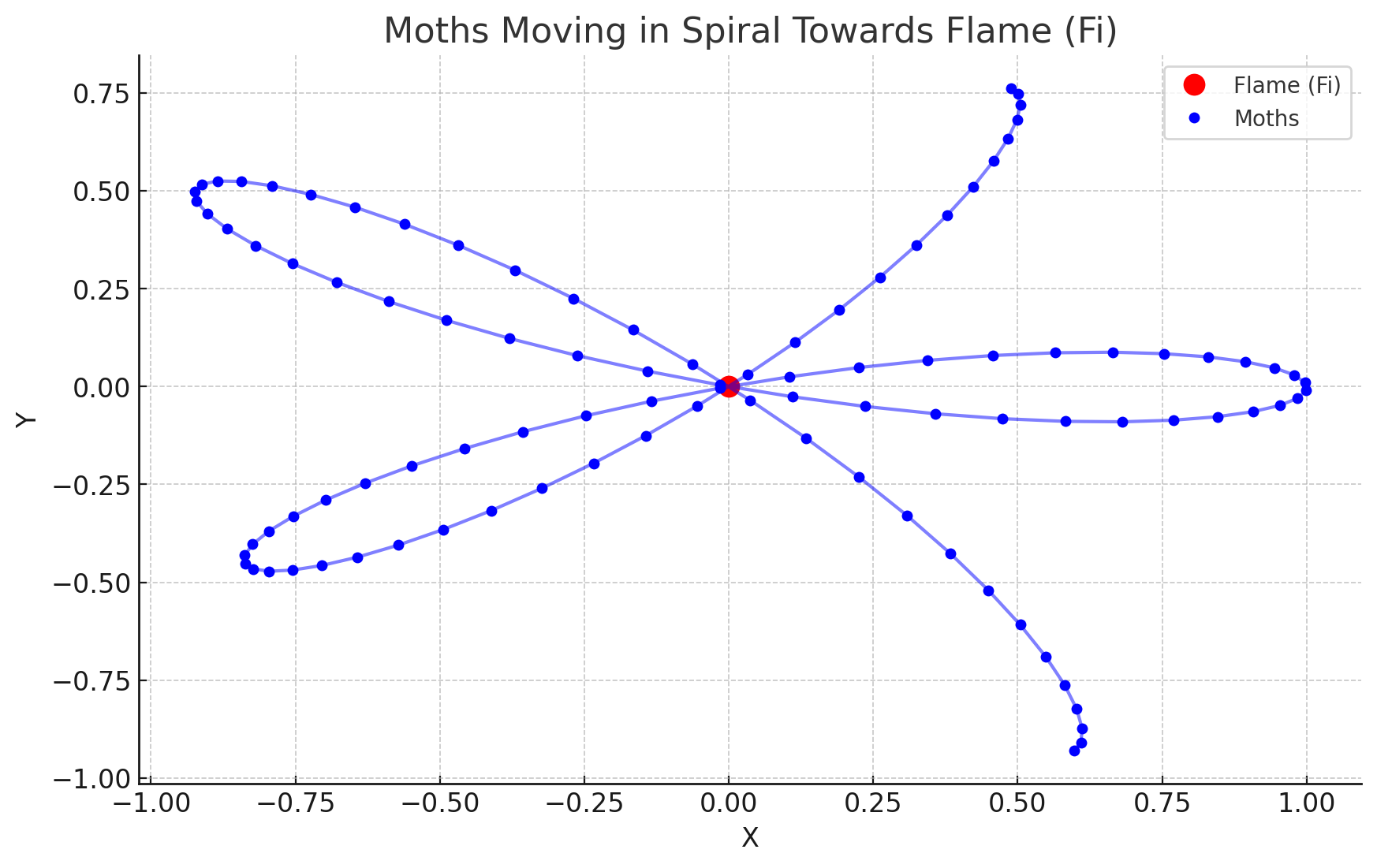
Di cho biết khoảng cách của bướm đêm thứ i đối với ngọn lửa thứ j, b là hằng số để xác định hình dạng của xoắn ốc logarit và t là một số ngẫu nhiên trong [-1,1].

D được tính như sau:

*Di = Fj – Mi | (2.3)*

Trong đó, Mi ​ chỉ ra con bướm thứ i, Fj ​ chỉ ra ngọn lửa thứ j, và Di chỉ ra khoảng cách của con bướm thứ i đối với ngọn lửa thứ j.

Phương trình (2.2) là nơi mô phỏng đường bay xoắn ốc của bướm đêm. Như có thể thấy trong phương trình này, vị trí tiếp theo của bướm đêm được xác định đối với ngọn lửa. Tham số t trong phương trình xoắn ốc xác định vị trí tiếp theo của bướm đêm nên gần ngọn lửa bao nhiêu (t = -1 là vị trí gần ngọn lửa nhất, trong khi t = 1 cho thấy vị trí xa nhất). Do đó, một siêu hình elip có thể được giả định xung quanh ngọn lửa theo mọi hướng và vị trí tiếp theo của bướm đêm sẽ nằm trong không gian này. Chuyển động xoắn ốc là thành phần chính của phương pháp được đề xuất vì nó quyết định cách bướm đêm cập nhật vị trí của chúng xung quanh ngọn lửa. Phương trình xoắn ốc cho phép một con bướm đêm bay "xung quanh" ngọn lửa và không nhất thiết phải ở trong khoảng trống giữa chúng. Do đó, việc thăm dò và khai thác không gian tìm kiếm có thể được đảm bảo. Xoắn ốc logarit, không gian xung quanh ngọn lửa và vị trí xem xét t khác nhau trên đường cong được minh họa trong Hình 3.



*Hình 3 : biểu đồ mô phỏng các con bướm di chuyển quanh ngọn lửa theo hình xoắn ốc*

Ảnh có chứa biểu đồ, hàng, Sơ đồ, ảnh chụp màn hình

Mô tả được tạo tự động

*Hình 4 : Một số vị trí có thể được bướm đêm có thể đạt được đối với ngọn lửa bằng cách sử dụng xoắn ốc logarit*

Hình 4 cho thấy một mô hình khái niệm về cập nhật vị trí của một con bướm đêm xung quanh ngọn lửa. Lưu ý rằng trục dọc chỉ hiển thị một chiều (1 biến/tham số của một bài toán nhất định), nhưng phương pháp được đề xuất có thể được sử dụng để thay đổi tất cả các biến của bài toán. Các vị trí có thể có (đường đứt nét màu đen) có thể được chọn làm vị trí tiếp theo của bướm đêm (đường ngang màu xanh lam) xung quanh ngọn lửa (đường ngang màu xanh lá cây) trong Hình 4 cho thấy rõ ràng rằng một con bướm đêm có thể khám phá và khai thác không gian tìm kiếm xung quanh ngọn lửa trong một chiều. Khám phá xảy ra khi vị trí tiếp theo nằm ngoài không gian giữa bướm đêm và lửa như có thể thấy trong các mũi tên được dán nhãn 1, 3 và 4. Khai thác xảy ra khi vị trí tiếp theo nằm bên trong khoảng trống giữa bướm đêm và ngọn lửa như có thể quan sát thấy trong mũi tên được dán nhãn bằng 2. Có một số quan sát thú vị cho mô hình này như sau:

* Một con bướm đêm có thể hội tụ đến bất kỳ điểm nào trong vùng lân cận của ngọn lửa bằng cách thay đổi t
* T càng thấp, khoảng cách càng gần ngọn lửa.
* Tần suất cập nhật vị trí ở cả hai bên của ngọn lửa được tăng lên khi bướm đêm đến gần ngọn lửa hơn

Một câu hỏi có thể nảy sinh ở đây là việc cập nhật vị trí trong Phương trình (2.2) chỉ yêu cầu bướm đêm di chuyển về phía ngọn lửa, nhưng nó khiến thuật toán MFO bị mắc kẹt trong tối ưu cục bộ một cách nhanh chóng. Để ngăn chặn điều này, mỗi con bướm đêm bắt buộc phải cập nhật vị trí của nó chỉ bằng cách sử dụng một trong các ngọn lửa trong Phương trình (2.2). Mỗi lần lặp lại và sau khi cập nhật danh sách ngọn lửa, ngọn lửa được sắp xếp dựa trên giá trị phù hợp của chúng. Sau đó, bướm đêm cập nhật vị trí của chúng đối với ngọn lửa tương ứng của chúng. Con bướm đêm đầu tiên luôn cập nhật vị trí của nó đối với ngọn lửa tốt nhất, trong khi con bướm đêm cuối cùng cập nhật vị trí của nó đối với ngọn lửa tồi tệ nhất trong danh sách. Hình 5 cho thấy cách mỗi con bướm đêm được gán cho một ngọn lửa trong danh sách ngọn lửa.

Cần lưu ý rằng giả định này được thực hiện để thiết kế thuật toán MFO, trong khi có thể nó không phải là hành vi thực tế của bướm đêm trong tự nhiên. Tuy nhiên, định hướng ngang vẫn được thực hiện bởi bướm đêm nhân tạo. Lý do tại sao một ngọn lửa cụ thể được chỉ định cho mỗi con bướm đêm là để ngăn chặn sự trì trệ tối ưu cục bộ. Nếu tất cả bướm đêm bị thu hút bởi một ngọn lửa duy nhất, tất cả chúng sẽ hội tụ đến một điểm trong không gian tìm kiếm vì chúng chỉ có thể bay về phía ngọn lửa chứ không phải ra ngoài. Tuy nhiên, yêu cầu chúng di chuyển xung quanh các ngọn lửa khác nhau gây ra việc khám phá không gian tìm kiếm cao hơn và xác suất trì trệ tối ưu cục bộ thấp hơn.

Ảnh có chứa văn bản, ảnh chụp màn hình, biểu đồ, hàng

Mô tả được tạo tự động

*Hình 5 : Mỗi con bướm đêm được gán cho một ngọn lửa*

Do đó, việc khám phá không gian tìm kiếm xung quanh các vị trí tốt nhất thu được cho đến nay được đảm bảo bằng phương pháp này vì những lý do sau:

* Bướm đêm cập nhật vị trí của chúng trong siêu cầu xung quanh các giải pháp tốt nhất thu được cho đến nay.
* Trình tự ngọn lửa được thay đổi dựa trên các giải pháp tốt nhất trong mỗi lần lặp lại và bướm đêm được yêu cầu cập nhật vị trí của chúng đối với ngọn lửa được cập nhật. Do đó, việc cập nhật vị trí của bướm đêm có thể xảy ra xung quanh các ngọn lửa khác nhau, một cơ chế gây ra sự di chuyển đột ngột của bướm đêm trong không gian tìm kiếm và thúc đẩy việc khám phá.

Một mối quan tâm khác ở đây là việc cập nhật vị trí của bướm đêm đối với n vị trí khác nhau trong không gian tìm kiếm có thể làm giảm khả năng khai thác các giải pháp hứa hẹn tốt nhất. Để giải quyết mối quan tâm này, chúng tôi đề xuất một cơ chế thích ứng với số lượng ngọn lửa. Hình 6 cho thấy số lượng ngọn lửa giảm một cách thích ứng như thế nào trong quá trình lặp lại. Chúng tôi sử dụng công thức sau về vấn đề này:

 (2.4)

Trong đó L là số lần lặp hiện tại, N là số lần lặp tối đa và T biểu thị số lần lặp tối đa.

Ảnh có chứa biểu đồ, hàng, văn bản, Hình chữ nhật

Mô tả được tạo tự động

*Hình 6. Số lượng ngọn lửa được giảm thích ứng trong quá trình lặp lại*

Hình 6 cho thấy có N số ngọn lửa trong các bước đầu tiên của các lần lặp. Tuy nhiên, bướm đêm chỉ cập nhật vị trí của chúng đối với ngọn lửa tốt nhất trong các bước cuối cùng của các lần lặp lại. Sự giảm dần về số lượng ngọn lửa cân bằng giữa việc khám phá và khai thác không gian tìm kiếm. Rốt cuộc, các bước chung của hàm P như sau.

Update flame no using Equation (2.4)

OM=FitnessFunction(M);

**if** iteration==1

F=sort(M);

OF=sort(OM);

**Else**

F=sort(Mt-1, Mt);

OF=sort(Mt-1, Mt);

**end**

**for** i = 1 : n

**for** j= 1 : d

Update r and t

Calculate D using Equation (2.3) with respect to the corresponding moth

Update M(i,j) using Eqs. (2.3) and (2.2) with respect to the corresponding moth

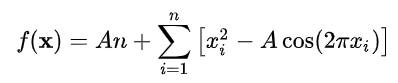
**end**

**end**

Như đã nói ở trên, hàm P được thực thi cho đến khi hàm T trả về true. Sau khi kết thúc hàm P, bướm đêm tốt nhất được trả về dưới dạng xấp xỉ thu được tốt nhất của tối ưu.

Ảnh có chứa biểu đồ, hàng, nghệ thuật gấp giấy origami, thiết kế

Mô tả được tạo tự độngSử dụng hàm Rastrigin trong không gian 2D hoặc 3D được định nghĩa như sau :



Trong đó:

* AAA: Hằng số dương, thường được chọn là A=10A = 10A=10.
* ddd: Số chiều của không gian. (Ở đây d=3d = 3d=3, tương ứng với các biến x,y,zx, y, zx,y,z).
* xix\_ixi​: Biến tại chiều thứ iii.
* Thành phần xi2x\_i^2xi2​: Làm hàm tăng nhanh với giá trị lớn của xix\_ixi​.
* Thành phần −A⋅cos⁡(2πxi)-A \cdot \cos(2 \pi x\_i)−A⋅cos(2πxi​): Tạo ra các dao động cục bộ (cực trị cục bộ).

## 3. Thực nghiệm, chạy thử, đánh giá kết quả của thuật toán

### 3.1 Thực nghiệm thuật toán MFO

**Bài toán tối ưu hóa:**

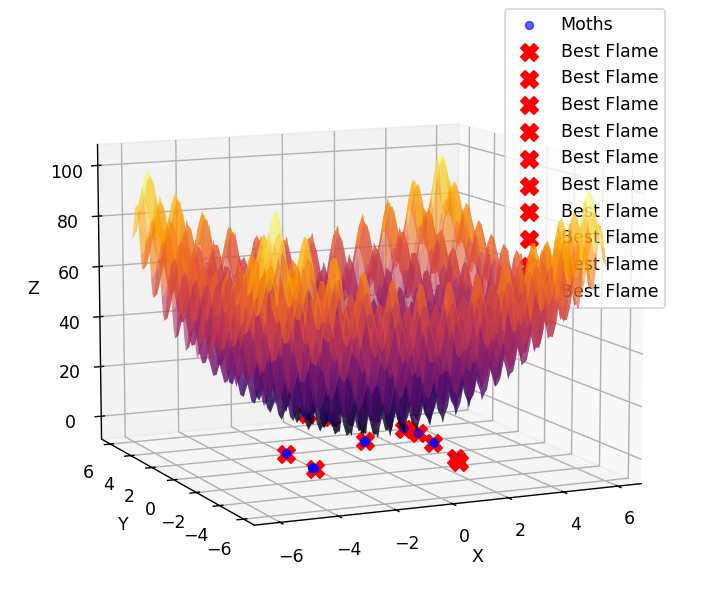
**Hàm mục tiêu:** Các bài toán tối ưu hóa sử dụng hàm mục tiêu Rastrigin.

* Hàm Rastrigin là hàm kiểm tra phổ biến trong tối ưu hóa vì nó có nhiều cực trị cục bộ, giúp kiểm tra khả năng thoát khỏi các bẫy cực trị của thuật toán.

**Thuật toán MFO:**

* **Khởi tạo:** Các con ngài (moths) được khởi tạo ngẫu nhiên trong không gian tìm kiếm.
* **Cập nhật:** Moths di chuyển về phía các flame (ngọn lửa) tốt nhất, quỹ đạo di chuyển có thể là xoắn ốc.
* **Số lượng bướm (moths) và ngọn lửa (flames):** Điều chỉnh số lượng bướm và ngọn lửa để kiểm tra tính ổn định và khả năng hội tụ.

### 3.2 chạy thử thuật toán MFO



Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm

Mô tả được tạo tự động

### 3.3 Đánh giá thuật toán MFO

MFO có thể xử lý các bài toán tối ưu hóa phức tạp và đạt được các kết quả tối ưu trong một thời gian hợp lý. Những yếu tố sau ảnh hưởng đến tính hiệu quả của thuật toán:

* **Khả năng thoát khỏi cực trị cục bộ**: MFO có khả năng tránh bị mắc kẹt trong cực trị cục bộ nhờ vào cơ chế bay xoắn ốc của bướm đêm (moths). Các bướm di chuyển theo quỹ đạo xoắn ốc hướng về các ngọn lửa (flames) để tìm kiếm giải pháp tối ưu. Điều này giúp tăng cường khả năng khám phá không gian tìm kiếm.
* **Khả năng khai thác**: Sau khi khám phá không gian, các bướm dần dần di chuyển về ngọn lửa tốt nhất, giúp thuật toán khai thác các giải pháp tốt hơn trong quá trình tìm kiếm.
* **Tốc độ hội tụ**: MFO có thể hội tụ nhanh trong các bài toán đơn giản, nhưng trong các bài toán phức tạp hoặc với không gian tìm kiếm lớn, tốc độ hội tụ có thể chậm. Điều này phụ thuộc vào số lượng bướm (moths) và số lượng ngọn lửa (flames).

MFO có thể được áp dụng trong nhiều lĩnh vực khác nhau nhờ vào tính linh hoạt của nó:

* **Tối ưu hóa đa chiều**: Thuật toán MFO có thể được áp dụng cho các bài toán tối ưu hóa đa chiều (nhiều biến), điều này làm nó phù hợp với các bài toán thực tế trong kỹ thuật, thiết kế, tài chính, v.v.
* **Tối ưu hóa trong các bài toán phi tuyến**: MFO là một thuật toán không yêu cầu thông tin đạo hàm và có thể áp dụng trong các bài toán tối ưu phi tuyến, không khả vi hoặc có cấu trúc phức tạp.
* **Ứng dụng trong học máy và AI**: MFO có thể được sử dụng để tối ưu hóa các siêu tham số trong các mô hình học máy, giúp cải thiện độ chính xác của các mô hình học sâu hoặc các thuật toán phân tích dữ liệu.

## 4. Ưu nhược điểm của thuật toán Moth flame optimization(MFO)

### 4.1. Ưu điểm của MFO

1. **Khả năng tránh cực trị cục bộ**: Một trong những điểm mạnh chính của thuật toán MFO là khả năng tránh được các cực trị cục bộ, một vấn đề phổ biến trong các bài toán tối ưu hóa. Nhờ vào cơ chế bay xoắn ốc của bướm đêm, thuật toán duy trì một sự cân bằng giữa khai thác (exploitation) và khám phá (exploration). Điều này giúp MFO có thể tiếp cận các vùng tìm kiếm chưa được khám phá, từ đó giảm thiểu khả năng bị mắc kẹt tại các điểm tối ưu cục bộ, một vấn đề mà nhiều thuật toán tối ưu hóa truyền thống như gradient-based algorithms hay các thuật toán toán học thường gặp phải.
2. **Dễ dàng áp dụng**: MFO có thể được áp dụng vào các bài toán tối ưu hóa mà không yêu cầu thông tin gradient hay đạo hàm của các hàm mục tiêu, làm cho thuật toán trở nên cực kỳ hữu ích trong những bài toán có không gian tìm kiếm phức tạp hoặc không thể tính toán được gradient. Điều này cũng giúp thuật toán có thể ứng dụng rộng rãi vào các bài toán thực tế trong nhiều lĩnh vực, như kỹ thuật, khoa học, kinh tế, và thiết kế hệ thống.
3. **Khả năng xử lý các bài toán đa dạng**: MFO có khả năng giải quyết các bài toán tối ưu hóa đa dạng, bao gồm các bài toán phi tuyến tính, các bài toán với không gian tìm kiếm cao chiều, hay các bài toán với nhiều ràng buộc. Thuật toán có thể xử lý cả các bài toán có tính chất hỗn loạn hoặc không thể mô hình hóa dễ dàng bằng các phương pháp tối ưu hóa truyền thống. Điều này khiến MFO trở thành một công cụ mạnh mẽ và linh hoạt cho nhiều bài toán thực tế.
4. **Tính đơn giản và dễ triển khai**: Một trong những ưu điểm nổi bật của MFO là cấu trúc đơn giản và dễ triển khai. Thuật toán không yêu cầu các phép toán phức tạp, giúp tiết kiệm tài nguyên tính toán và thời gian phát triển phần mềm. Hơn nữa, thuật toán có thể được triển khai trong môi trường đa dạng mà không cần quá nhiều sự điều chỉnh hay tối ưu hóa các tham số ban đầu.

### 4.2. Nhược điểm của MFO

1. **Hiệu suất chưa ổn định**: Mặc dù MFO có thể hoạt động hiệu quả trong nhiều trường hợp, nhưng hiệu suất của nó không phải lúc nào cũng ổn định. Trong một số tình huống, khi áp dụng vào các bài toán tối ưu hóa có không gian tìm kiếm cực kỳ phức tạp hoặc có nhiều yếu tố không xác định, thuật toán có thể gặp khó khăn trong việc duy trì sự ổn định của quá trình hội tụ. Điều này có thể dẫn đến kết quả tối ưu không được cải thiện đáng kể sau nhiều vòng lặp, hoặc mất nhiều thời gian để hội tụ đến giá trị tối ưu.
2. **Cần điều chỉnh các tham số**: MFO yêu cầu một số tham số nhất định, chẳng hạn như kích thước quần thể (số lượng bướm đêm), tốc độ hội tụ, các yếu tố điều chỉnh quỹ đạo bay và các yếu tố khác. Việc lựa chọn và điều chỉnh các tham số này có thể là một quá trình tốn thời gian và phức tạp, vì các tham số này có ảnh hưởng lớn đến hiệu quả của thuật toán. Việc tối ưu hóa các tham số này thường yêu cầu thử nghiệm và điều chỉnh nhiều lần, điều này có thể làm tăng chi phí tính toán trong quá trình phát triển.
3. **Khả năng hội tụ chậm**: Một nhược điểm khác của MFO là khả năng hội tụ chậm trong một số trường hợp, đặc biệt là đối với các bài toán có không gian tìm kiếm rộng và phức tạp. Mặc dù thuật toán có thể tránh được cực trị cục bộ, nhưng quá trình khám phá và khai thác không gian tìm kiếm có thể mất nhiều thời gian, dẫn đến hiệu quả không cao nếu so với các thuật toán tối ưu hóa khác, đặc biệt là khi không gian tìm kiếm có nhiều chiều hoặc rất phức tạp.
4. **Tính toán chi phí cao trong không gian siêu chiều**: Khi áp dụng MFO vào các bài toán có không gian tìm kiếm siêu chiều (high-dimensional), thuật toán có thể gặp phải vấn đề về chi phí tính toán cao. Điều này là do thuật toán phải xử lý nhiều thông tin và trạng thái của quần thể bướm đêm trong không gian đa chiều, làm tăng độ phức tạp tính toán. Trong những bài toán có rất nhiều chiều, việc tìm kiếm và hội tụ có thể trở nên tốn kém về mặt thời gian và tài nguyên tính toán.

* Mặc dù có những nhược điểm này, MFO vẫn là một thuật toán mạnh mẽ với nhiều ưu điểm trong việc giải quyết các bài toán tối ưu hóa phức tạp. Với khả năng tránh cực trị cục bộ, không yêu cầu thông tin gradient, và có thể xử lý nhiều loại bài toán khác nhau, MFO hứa hẹn sẽ tiếp tục là một công cụ quan trọng trong các ứng dụng tối ưu hóa trong tương lai.

## 5. Kết Luận

Thuật toán Moth Flame Optimization (MFO) là một thuật toán tối ưu hóa metaheuristic được phát triển dựa trên hành vi tự nhiên của loài bướm khi tìm kiếm ngọn lửa. Cảm hứng từ quá trình di chuyển của bướm về phía ngọn lửa để thu hút sự chú ý của nó được sử dụng để mô phỏng quá trình tối ưu hóa, với mục tiêu tìm kiếm giá trị tối ưu trong không gian tìm kiếm đa chiều. Một điểm mạnh nổi bật của MFO là khả năng tránh được các cực trị cục bộ nhờ vào sự đa dạng trong cách di chuyển của các cá thể bướm, giúp thuật toán tìm ra giải pháp tối ưu toàn cục, thay vì dừng lại ở các cực trị cục bộ như một số thuật toán khác.

Quá trình di chuyển của bướm trong thuật toán MFO được mô phỏng thông qua các quỹ đạo xoắn ốc, nơi mỗi con bướm (moth) di chuyển về phía ngọn lửa (flame) gần nhất, theo một tốc độ và hướng được điều chỉnh dựa trên khoảng cách giữa nó và ngọn lửa. Điều này giúp các con bướm dần tiến gần hơn đến các giá trị tối ưu trong không gian tìm kiếm. Các tham số của thuật toán như số lượng bướm, số vòng lặp, và giới hạn không gian tìm kiếm có thể được điều chỉnh để tối ưu hóa quá trình tìm kiếm.

Thuật toán MFO có thể được áp dụng rộng rãi trong nhiều lĩnh vực, từ tối ưu hóa các bài toán phức tạp trong công nghiệp, tối ưu hóa tham số trong học máy, đến các bài toán tìm kiếm giải pháp trong các hệ thống phức tạp. MFO có thể dễ dàng kết hợp với các thuật toán khác để tăng cường hiệu quả tìm kiếm và cải thiện tốc độ hội tụ, đồng thời có khả năng làm việc tốt trong các không gian tìm kiếm lớn và không xác định. Bởi vì MFO là một thuật toán không có sự giả định về hình thức của hàm mục tiêu, nó có thể hoạt động hiệu quả trên các hàm không tuyến tính, không liên tục, hoặc có nhiều cực trị.

Tuy nhiên, MFO cũng có những hạn chế nhất định. Một trong những vấn đề phổ biến của thuật toán là tốc độ hội tụ chậm, đặc biệt khi không có sự điều chỉnh phù hợp về tham số hoặc trong các bài toán có không gian tìm kiếm quá rộng. Thêm vào đó, việc điều chỉnh tham số của thuật toán, chẳng hạn như số lượng bướm, số vòng lặp và tốc độ di chuyển, có thể gặp khó khăn khi không có thông tin đầy đủ về bài toán cụ thể. Điều này có thể ảnh hưởng đến hiệu quả của thuật toán, khiến nó không đạt được kết quả tối ưu trong một khoảng thời gian hợp lý.

Tuy vậy, MFO vẫn được đánh giá là một công cụ mạnh mẽ và linh hoạt trong việc giải quyết các bài toán tối ưu hóa phức tạp. Với sự cải tiến và kết hợp với các kỹ thuật khác như học máy, thuật toán di truyền, hoặc các phương pháp tối ưu hóa bầy đàn, MFO có thể đạt được hiệu quả cao hơn và vượt qua những nhược điểm của nó, giúp giải quyết một loạt các bài toán khó mà các thuật toán khác không thể giải quyết được. MFO có thể dễ dàng áp dụng cho các bài toán tối ưu hóa trong các lĩnh vực khoa học, kỹ thuật, tài chính, và nhiều ứng dụng thực tế khác.

# CHƯƠNG II: BAT ALGORITHM (BA)

## 1. Giới thiệu và trình bày động lực đề xuất thuật toán

### 1.1. Giới thiệu thuật toán con dơi (BA)

Các thuật toán tối ưu hóa hiện đại thường lấy cảm hứng từ thiên nhiên, đặc biệt là từ trí thông minh bầy đàn. Những phương pháp này rất đa dạng, dẫn đến sự xuất hiện của nhiều loại thuật toán khác nhau. Tuy nhiên, hầu hết các thuật toán này đều sử dụng một số đặc điểm chung để xây dựng công thức cập nhật. Ví dụ, thuật toán di truyền được lấy cảm hứng từ lý thuyết tiến hóa của Darwin về các hệ sinh thái sinh học, và sử dụng các toán tử như giao nhau, đột biến và lựa chọn người phù hợp nhất. Các giải pháp trong thuật toán di truyền thường được biểu diễn dưới dạng nhiễm sắc thể hoặc chuỗi nhị phân/thực.

Trong khi đó, thuật toán tối ưu hóa bầy hạt (PSO) lại dựa trên hành vi bầy đàn của các loài chim và cá. Hệ thống đa tác nhân này thể hiện các đặc điểm nổi bật của trí thông minh nhóm và có thể tìm ra các giải pháp tối ưu hiệu quả (Kennedy và Eberhart, 1995). Các biến thể của PSO cũng như các cải tiến của nó đã được nghiên cứu và phát triển (Cui, 2009; Yang, 2010; Yang và Deb, 2010; Yang và cộng sự, 2011; Yang và cộng sự, 2013). Mặc dù thuật toán di truyền và PSO rất hữu ích, chúng vẫn gặp phải một số hạn chế khi xử lý các vấn đề tối ưu hóa đa phương thức.

Một cải tiến đáng chú ý là thuật toán đom đóm (FA), được lấy cảm hứng từ hành vi nhấp nháy của đom đóm nhiệt đới (Yang, 2008). Các đặc điểm như hành vi hấp dẫn, mã hóa cường độ ánh sáng và sự phụ thuộc vào khoảng cách giúp thuật toán đom đóm có khả năng xử lý hiệu quả các bài toán tối ưu hóa đa phương thức và phi tuyến. Bên cạnh đó, thuật toán tìm kiếm chim cu gáy (CS), dựa trên hành vi ấp trứng của các loài chim cu gáy (Yang và Deb, 2009; Gandomi và cộng sự, 2013), kết hợp với các chuyến bay L'evy, mang lại hiệu quả cao nhờ hành vi hội tụ tốt, có thể được lý giải qua lý thuyết xác suất Markov. Những phương pháp khác như chiến lược đại bàng cũng tỏ ra rất hiệu quả (Yang và Deb, 2010; Gandomi và cộng sự, 2012).

Một thuật toán mới và thú vị là thuật toán dơi (BA), dựa trên đặc điểm định vị tiếng vang của loài dơi (Yang, 2010). BA sử dụng kỹ thuật điều chỉnh tần số để tăng sự đa dạng của các giải pháp trong quần thể. Đồng thời, thuật toán này sử dụng cơ chế thu phóng tự động để cân bằng quá trình thăm dò và khai thác trong suốt quá trình tìm kiếm, bắt chước sự thay đổi trong tốc độ phát xạ xung và độ ồn của dơi khi săn mồi. Kết quả là thuật toán này có khả năng khởi động nhanh và hiệu quả. Tuy nhiên, vẫn còn nhiều không gian để cải thiện, và bài báo này dự định sẽ xem xét những phát triển mới nhất của thuật toán dơi.

### 1.2. Động lực đề xuất thuật toán

Thuật toán bay của dơi (Bat Algorithm - BA) được đề xuất dựa trên cơ chế sinh học của loài dơi trong việc tìm kiếm thức ăn, đặc biệt là cách mà chúng sử dụng tiếng vang để điều hướng trong không gian. Dơi di chuyển trong không gian ba chiều và sử dụng tín hiệu âm thanh (hay sóng siêu âm) để phát hiện vật thể xung quanh. Chúng phát ra sóng siêu âm, sau đó lắng nghe phản hồi từ các vật thể để xác định khoảng cách và vị trí của chúng. Cơ chế này được gọi là “echolocation” (định vị bằng tiếng vang). Dựa trên nguyên lý này, thuật toán BA được thiết kế để mô phỏng cách dơi di chuyển và tìm kiếm nguồn thức ăn trong không gian tìm kiếm.

Trong thế giới thực, dơi có thể thay đổi tốc độ bay và tần số sóng siêu âm để phù hợp với môi trường và tìm kiếm một cách hiệu quả. Chúng điều chỉnh tần số sóng siêu âm của mình, từ cao đến thấp, để tối ưu hóa khả năng phát hiện mục tiêu và tránh các chướng ngại vật. Cũng giống như vậy, thuật toán BA mô phỏng sự thay đổi của vị trí dơi trong không gian tìm kiếm, điều chỉnh tốc độ bay và phương hướng của dơi sao cho tối ưu hóa được kết quả tìm kiếm. Vị trí của dơi trong không gian được coi là các giải pháp khả thi của bài toán tối ưu hóa.

Một điểm đặc biệt của thuật toán BA là khả năng điều chỉnh sự thay đổi tốc độ bay và tần số sóng siêu âm, phản ánh quá trình dõi theo và tìm kiếm mục tiêu với các mức độ khác nhau trong suốt quá trình tối ưu hóa. Ban đầu, dơi có thể bay nhanh và không xác định rõ mục tiêu, nhưng theo thời gian, chúng dần chậm lại và hướng đến mục tiêu cụ thể khi đã nhận diện được khoảng cách và hướng đi. Tương tự, thuật toán BA bắt đầu với một tập hợp các giải pháp ngẫu nhiên và dần dần tìm kiếm điểm tối ưu thông qua các bước di chuyển của dơi, từ nhanh (khám phá) sang chậm và chính xác (khai thác).

Động lực đề xuất thuật toán BA xuất phát từ khả năng kết hợp giữa quá trình khai thác (exploitation) và khám phá (exploration) trong không gian tìm kiếm. Trong khi các thuật toán tối ưu hóa truyền thống có thể gặp phải vấn đề với việc rơi vào tối ưu cục bộ, thuật toán BA với cơ chế điều chỉnh tốc độ và tần số cho phép cân bằng giữa việc khám phá không gian rộng lớn và khai thác các vùng đã tìm thấy giải pháp tốt. Điều này giúp thuật toán có khả năng thoát khỏi các cực trị địa phương, tìm kiếm giải pháp tối ưu toàn cục.

Thuật toán BA đặc biệt thích hợp cho các bài toán tối ưu hóa phức tạp, nơi không có thông tin rõ ràng về hàm mục tiêu và không gian tìm kiếm rộng lớn. Bằng cách mô phỏng hành vi tự nhiên của dơi trong việc xác định vị trí mục tiêu, thuật toán có thể tìm ra các giải pháp tối ưu hoặc gần tối ưu trong các bài toán đa biến, không tuyến tính.

### 1.3. Nguồn càm hứng từ loài dơi

Thuật toán con dơi (Bat Algorithm - BA) được lấy cảm hứng từ hành vi định vị bằng sóng siêu âm của loài dơi trong tự nhiên. Đây là một trong những cơ chế tìm kiếm và điều hướng độc đáo, giúp dơi săn mồi và tránh vật cản trong môi trường phức tạp. Nguồn cảm hứng này có thể được hiểu qua các đặc điểm sinh học và hành vi đặc trưng của dơi:

\* **Định vị bằng sóng siêu âm (Echolocation)**

Dơi sử dụng sóng siêu âm để xác định vị trí của con mồi và các vật thể xung quanh trong bóng tối. Cơ chế này hoạt động như sau:

* **Phát sóng siêu âm:**
* Dơi phát ra các tín hiệu âm thanh có tần số cao (khoảng từ 20 kHz đến 100 kHz), vượt quá ngưỡng nghe của con người.
* **Thu sóng phản xạ:**

Khi sóng âm chạm vào các vật thể trong môi trường, chúng phản xạ lại và được tai hoặc các cơ quan cảm biến của dơi thu nhận.

* **Phân tích tín hiệu:**

Dựa trên thời gian hồi âm (time delay), cường độ tín hiệu, và sự thay đổi tần số, dơi xác định được khoảng cách, vị trí, và kích thước của vật thể.

* + **Khoảng cách gần:** Sóng phản xạ quay lại nhanh.
  + **Kích thước lớn:** Sóng phản xạ mạnh hơn.
* **Ứng dụng trong săn mồi:**

Dơi sử dụng echolocation để phát hiện con mồi nhỏ như côn trùng đang bay và bắt chúng một cách chính xác ngay cả trong bóng tối hoàn toàn.

**Cảm hứng trong thuật toán:**

* Cơ chế phát và thu tín hiệu trong echolocation được mô phỏng trong thuật toán để tìm kiếm và đánh giá các giải pháp trong không gian tìm kiếm.
* Mỗi con dơi đại diện cho một cá thể tìm kiếm, sử dụng tần số và âm lượng để điều hướng và cập nhật vị trí.

\* **Khả năng điều chỉnh tần số và âm lượng**

Trong tự nhiên, dơi điều chỉnh tần số và âm lượng của sóng siêu âm theo khoảng cách và mục tiêu:

* **Tần số:**
  + Dơi tăng tần số để phát hiện các vật thể nhỏ hơn hoặc ở khoảng cách gần.
  + Tần số thấp hơn được sử dụng để khám phá khu vực rộng lớn.
* **Âm lượng:**
  + Khi dơi tiếp cận mục tiêu, âm lượng sóng siêu âm giảm dần, giúp tập trung vào khu vực tiềm năng mà không gây nhiễu từ môi trường xung quanh.

**Cảm hứng trong thuật toán:**

* Tần số của dơi trong thuật toán được liên kết với tốc độ di chuyển và khả năng khám phá không gian tìm kiếm.
* Âm lượng đại diện cho mức độ khai thác các vùng tiềm năng, giảm dần khi thuật toán tiến gần đến lời giải tối ưu.

**\*Cân bằng giữa khám phá và khai thác**

Hành vi săn mồi của dơi là một sự cân bằng hoàn hảo giữa:

* **Khám phá (Exploration):**
  + Khi dơi bắt đầu săn mồi, chúng khám phá một khu vực rộng lớn để tìm kiếm dấu hiệu của con mồi.
* **Khai thác (Exploitation):**
  + Khi phát hiện con mồi, chúng tập trung vào khu vực nhỏ, giảm biên độ sóng để tăng độ chính xác trong việc bắt mồi.

**Cảm hứng trong thuật toán:**

* Thuật toán con dơi được thiết kế để cân bằng giữa việc khám phá không gian rộng (tìm kiếm toàn cục) và khai thác chi tiết trong các vùng tiềm năng cao (tìm kiếm cục bộ).
* Điều này giúp thuật toán tránh rơi vào cực trị cục bộ và tăng cơ hội tìm được lời giải tối ưu toàn cục.

**\* Khả năng thích nghi với môi trường phức tạp**

Dơi sống trong môi trường tối và phức tạp, nơi chúng phải:

* Tránh va chạm với các vật cản như cây cối, hang động.
* Đối phó với các yếu tố gây nhiễu (tiếng vang, gió, hoặc nhiều vật thể di chuyển).

**Cảm hứng trong thuật toán:**

* Mỗi cá thể dơi trong thuật toán tự điều chỉnh tần số, tốc độ, và hướng di chuyển để thích nghi với không gian tìm kiếm.
* Tính ngẫu nhiên trong thuật toán được sử dụng để mô phỏng khả năng thích nghi và tạo ra sự đa dạng trong hành vi tìm kiếm.

**\* Hành vi bầy đàn**

Mặc dù dơi là loài săn mồi đơn lẻ, chúng thường di chuyển theo nhóm lớn. Hành vi bầy đàn này mang lại lợi ích:

* **Chia sẻ thông tin:**

Các dơi trong đàn có thể tận dụng thông tin từ những cá thể khác để định hướng và săn mồi hiệu quả hơn.

* **Cạnh tranh và hợp tác:**

Các cá thể dơi trong đàn cạnh tranh để bắt mồi nhưng cũng phối hợp để giảm nguy cơ va chạm hoặc bị lạc.

**Cảm hứng trong thuật toán:**

* Trong thuật toán con dơi, các cá thể không chỉ di chuyển độc lập mà còn bị ảnh hưởng bởi cá thể tốt nhất, giúp tăng hiệu quả tìm kiếm và hội tụ nhanh hơn.

**\* Tính ngẫu nhiên trong hành vi**

Dơi sử dụng các chiến lược bay ngẫu nhiên để khám phá môi trường và tăng khả năng tìm thấy con mồi. Sự ngẫu nhiên này giúp chúng thích nghi tốt hơn với môi trường không quen thuộc hoặc phức tạp.

**Cảm hứng trong thuật toán:**

* Thành phần ngẫu nhiên được tích hợp để cải thiện khả năng khám phá không gian tìm kiếm và tránh rơi vào cực trị cục bộ.

## 2. Nội dung và các bước của thuật toán

Dựa trên mô tả và đặc điểm về định vị tiếng vang của dơi, Xin-She Yang (2010) đã phát triển thuật toán dơi với ba quy tắc lý tưởng hóa sau: 1. Tất cả dơi đều sử dụng định vị tiếng vang để cảm nhận khoảng cách, và chúng cũng 'biết' sự khác biệt giữa thức ăn / con mồi và rào cản nền theo một cách kỳ diệu nào đó; 2. Dơi bay ngẫu nhiên với vận tốc vi ở vị trí xi với tần số, thay đổi bước sóng λ và độ lớn A0 để tìm kiếm con mồi. Chúng có thể tự động điều chỉnh bước sóng (hoặc tần số) của các xung phát ra và điều chỉnh tốc độ phát xạ xung r ∈ [0, 1], tùy thuộc vào khoảng cách của mục tiêu của chúng; 3. Mặc dù âm lượng có thể thay đổi theo nhiều cách, nhưng chúng tôi giả định rằng âm lượng thay đổi từ A0 lớn (dương) đến giá trị hằng số tối thiểu .

Để đơn giản, chúng tôi không sử dụng dò tia trong thuật toán này, mặc dù nó có thể tạo thành một tính năng thú vị để mở rộng thêm. Nói chung, dò tia có thể mang tính toán mở rộng, nhưng nó có thể là một tính năng rất hữu ích cho hình học tính toán và các ứng dụng khác. Hơn nữa, một tần số nhất định về bản chất được liên kết với một bước sóng. Ví dụ, dải tần [20kHz, 500kHz] tương ứng với dải bước sóng từ 0,7mm đến 17mm trong không khí. Do đó, chúng ta có thể mô tả sự thay đổi về tần số f hoặc bước sóng λ để phù hợp với các ứng dụng khác nhau, tùy thuộc vào mức độ dễ thực hiện và các yếu tố khác.

### 2.1 Thuật toán chuyển động dơi

Mỗi con dơi được liên kết với một vận tốc và một địa điểm , Tại lần lặp t, trong không gian tìm kiếm hoặc giải pháp d-dimension. Trong số tất cả các con dơi, tồn tại một giải pháp tốt nhất hiện tại x\*. Do đó, ba quy tắc trên có thể được dịch thành các phương trình cập nhật cho và vận tốc.  
 =min + (max -- min), (1)

= + ( it –1 - \*)i, (2)

=  –1 + , (3)

Đâu là một vectơ ngẫu nhiên được vẽ từ một phân phối đồng nhất.

Như đã đề cập trước đó, chúng ta có thể sử dụng bước sóng hoặc tần số để thực hiện, chúng ta sẽ sử dụng min = 0 và max  = O(1) tùy thuộc vào kích thước tên miền của vấn đề quan tâm. Ban đầu, mỗi con dơi được chỉ định ngẫu nhiên một tần số được rút ra đồng nhất từ . Vì lý do này, thuật toán dơi có thể được coi là một thuật toán điều chỉnh tần số để cung cấp sự kết hợp cân bằng giữa khám phá và khai thác. Âm lượng và tốc độ phát xạ xung về cơ bản cung cấp một cơ chế để điều khiển tự động và tự động phóng to vào khu vực với các giải pháp đầy hứa hẹn.

### 2.2 Các biến thể về âm lượng và tốc độ xung

Để cung cấp cơ chế hiệu quả để kiểm soát việc thăm dò, khai thác và chuyển sang giai đoạn khai thác khi cần thiết, chúng ta phải thay đổi âm lượng và tỷ lệ phát xạ xung trong quá trình lặp lại. Vì âm lượng thường giảm khi một con dơi đã tìm thấy con mồi, trong khi tốc độ phát ra xung tăng lên, độ lớn có thể được chọn như bất kỳ giá trị thuận tiện nào, giữa và Giả có nghĩa là một con dơi vừa tìm thấy con mồi và tạm thời ngừng phát ra bất kỳ âm thanh nào. Với những giả định này, chúng ta có

+1 = , +1 = (4)

Đâu là hằng số. Về bản chất, ở đây tương tự như hệ số làm mát của lịch làm mát trong ủ mô phỏng. Đối với bất kỳ , chúng tôi có

, , (5

Trong trường hợp đơn giản nhất, chúng ta có thể sử dụng , và chúng tôi đã sử dụng trong mô phỏng của chúng tôi.

## 3. Thực nghiệm, chạy thử, đánh giá kết quả của thuật toán

### 3.1 Thực nghiệm thuật toán BA

**Bài toán tối ưu hóa:**

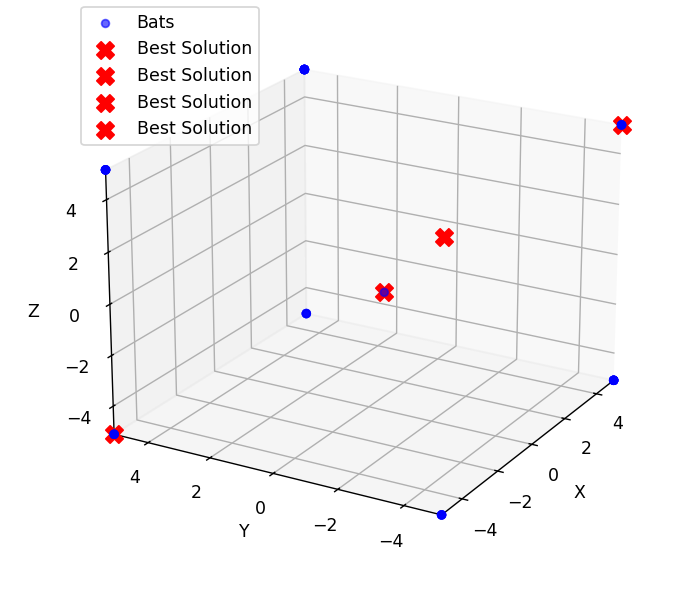
**Hàm mục tiêu:** Các bài toán tối ưu hóa sử dụng hàm mục tiêu Rastrigin.

* Hàm Rastrigin là hàm kiểm tra phổ biến trong tối ưu hóa vì nó có nhiều cực trị cục bộ, giúp kiểm tra khả năng thoát khỏi các bẫy cực trị của thuật toán.

**Thuật toán BA:**

* **Khởi tạo:** Các con dơi (bats) được khởi tạo ngẫu nhiên trong không gian tìm kiếm. Mỗi dơi sẽ có một **vị trí ngẫu nhiên** và **vận tốc ngẫu nhiên** trong không gian đó. Các tham số khác như **tần số phát sóng, tần suất phát sóng**, và **độ lớn sóng âm** cũng được khởi tạo ngẫu nhiên.
* **Cập nhật:** Trong mỗi vòng lặp, các dơi sẽ được cập nhật vận tốc và vị trí của mình dựa trên các yếu tố như tần số sóng âm, tần suất phát sóng, và độ lớn sóng âm. Đồng thời, dơi sẽ khám phá không gian và cập nhật lại vị trí của mình nếu phát hiện ra một vị trí tốt hơn (giá trị fitness tốt hơn).
* **Số lượng dơi (bats)**: Số lượng dơi ảnh hưởng đến khả năng tìm kiếm không gian giải pháp của thuật toán. Nếu số lượng dơi quá ít, thuật toán có thể hội tụ nhanh nhưng có thể bỏ qua các giá trị tối ưu khác trong không gian tìm kiếm. Ngược lại, số lượng dơi quá nhiều có thể làm tăng chi phí tính toán nhưng giúp thuật toán khám phá không gian tìm kiếm rộng hơn.

### 3.2 chạy thử thuật toán MFO





Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm

Mô tả được tạo tự động

### 3.3 Đánh giá thuật toán BA

**Khả năng hội tụ nhanh**:

* + **Bat Algorithm** có thể hội tụ khá nhanh đối với các bài toán có không gian tìm kiếm không quá lớn. Trong những bài toán đơn giản, BA có thể nhanh chóng di chuyển đến cực trị tối ưu hoặc gần tối ưu.
  + Tuy nhiên, trong các bài toán tối ưu hóa với không gian tìm kiếm lớn và phức tạp, tốc độ hội tụ có thể giảm đi, đặc biệt là khi không gian có nhiều cực trị cục bộ. Điều này xảy ra vì BA phải di chuyển qua nhiều điểm trong không gian tìm kiếm, có thể làm giảm tốc độ hội tụ.

**Khả năng thoát khỏi cực trị cục bộ**:

* + **BA** sử dụng cơ chế tiếng vang và di chuyển ngẫu nhiên để giúp dơi tránh bị mắc kẹt trong các cực trị cục bộ. Điều này đặc biệt quan trọng đối với các bài toán có nhiều cực trị cục bộ như hàm **Rastrigin** hay **Sphere Function.**
  + Tuy nhiên, khả năng thoát khỏi cực trị cục bộ cũng phụ thuộc vào việc điều chỉnh các tham số như tần số, độ lớn âm lượng và tỷ lệ xung. Nếu tham số không được điều chỉnh hợp lý, thuật toán có thể bị mắc kẹt ở cực trị cục bộ.

**Ứng dụng rộng rãi**:

* + BA có thể áp dụng vào nhiều bài toán tối ưu hóa khác nhau, đặc biệt là những bài toán có không gian tìm kiếm lớn và có nhiều cực trị cục bộ.
  + Nó đã được sử dụng thành công trong các lĩnh vực như **học máy, tối ưu hóa tham số** trong mô hình, **quản lý tài nguyên**, và **hệ thống kiểm soát.**

**Tối ưu hóa đa mục tiêu**:

* + **BA** có thể được mở rộng để giải quyết các bài toán tối ưu hóa đa mục tiêu. Tuy nhiên, khi áp dụng vào các bài toán có nhiều mục tiêu, BA có thể gặp khó khăn trong việc cân bằng giữa các mục tiêu khác nhau.

**Không gian tìm kiếm phi tuyến và không khả vi**:

* + BA có thể xử lý các bài toán tối ưu hóa mà không yêu cầu thông tin đạo hàm của hàm mục tiêu, vì vậy rất thích hợp với các bài toán phi tuyến, không khả vi hoặc không liên tục.

## 4. Ưu nhược điểm của thuật toán Bat algorithm (BA)

4.1. **Ưu điểm của BA**

* **Cân bằng giữa khám phá và khai thác**: Một trong những ưu điểm lớn nhất của thuật toán BA là khả năng cân bằng giữa việc khám phá không gian tìm kiếm (exploration) và khai thác các vùng đã tìm thấy giải pháp tốt (exploitation). Trong giai đoạn đầu, dơi có thể bay với tốc độ cao và khám phá không gian rộng lớn để phát hiện các vùng tiềm năng. Sau đó, dơi sẽ giảm tốc độ và tập trung vào việc khai thác các vùng đã tìm thấy giải pháp tốt, giúp tránh được việc rơi vào cực trị cục bộ và đạt được kết quả tối ưu toàn cục.
* **Đơn giản và dễ triển khai**: Thuật toán BA có cấu trúc đơn giản và dễ hiểu, không yêu cầu phải tính toán đạo hàm hay gradient như một số thuật toán tối ưu hóa truyền thống. Điều này làm cho thuật toán BA phù hợp với các bài toán tối ưu hóa phức tạp, đặc biệt là những bài toán mà hàm mục tiêu không có dạng rõ ràng hoặc không khả vi. Chính vì vậy, BA có thể được triển khai trong nhiều ứng dụng khác nhau mà không gặp phải vấn đề phức tạp trong việc tính toán các đạo hàm.
* **Hội tụ nhanh**: So với nhiều thuật toán tối ưu hóa sinh học khác, BA thường hội tụ nhanh hơn đến giải pháp tối ưu. Nhờ vào khả năng điều chỉnh linh hoạt giữa khám phá và khai thác, thuật toán có thể tìm kiếm và đạt được giải pháp tốt trong một khoảng thời gian ngắn hơn, đặc biệt là khi không gian tìm kiếm không quá lớn và phức tạp.
* **Tính ngẫu nhiên linh hoạt**: Thuật toán BA sử dụng tính ngẫu nhiên trong việc điều chỉnh tốc độ và phương hướng bay của các cá thể, điều này giúp tăng cường khả năng khám phá không gian tìm kiếm một cách rộng rãi và đa dạng. Nhờ vào tính ngẫu nhiên này, thuật toán có thể tránh được những vùng đã cạn kiệt thông tin và tiếp tục tìm kiếm các giải pháp mới trong các vùng chưa được khám phá, từ đó tạo ra một không gian tìm kiếm phong phú và đa dạng.
* **Linh hoạt và đa năng**: Thuật toán BA có thể được áp dụng cho nhiều loại bài toán khác nhau, bao gồm bài toán tối ưu hóa liên tục, rời rạc, và thậm chí là bài toán đa mục tiêu. Điều này giúp thuật toán BA có thể ứng dụng rộng rãi trong nhiều lĩnh vực như tối ưu hóa chuỗi cung ứng, tối ưu hóa thiết kế kỹ thuật, hoặc các bài toán tối ưu trong học máy.
* **Không yêu cầu thông tin không gian**: BA không yêu cầu thông tin về gradient hay cấu trúc không gian của hàm mục tiêu, điều này làm cho thuật toán thích hợp với các bài toán tối ưu hóa phi tuyến, không khả vi hoặc không liên tục. Vì vậy, thuật toán có thể xử lý tốt các bài toán phức tạp mà các phương pháp tối ưu hóa truyền thống không thể giải quyết hiệu quả.

4.2. **Nhược điểm của BA**

* **Hội tụ sớm**: Một trong những hạn chế của thuật toán BA là khả năng hội tụ sớm, đặc biệt khi không gian tìm kiếm có quá nhiều cực trị cục bộ. Nếu không có cơ chế điều chỉnh thích hợp, dơi có thể nhanh chóng rơi vào một cực trị không phải là tối ưu toàn cục, dẫn đến việc không thể tìm ra giải pháp tối ưu thật sự. Điều này có thể làm giảm hiệu quả của thuật toán trong các bài toán có không gian tìm kiếm phức tạp.
* **Phụ thuộc vào tham số**: Thuật toán BA rất nhạy cảm với việc lựa chọn các tham số khởi tạo, chẳng hạn như tốc độ bay, tần số sóng siêu âm, và các tham số điều chỉnh khác. Việc lựa chọn sai tham số có thể khiến thuật toán hoạt động kém hiệu quả, làm cho quá trình tối ưu hóa trở nên chậm chạp hoặc dễ rơi vào cực trị cục bộ. Do đó, việc điều chỉnh tham số cho phù hợp với từng bài toán là rất quan trọng.
* **Hiệu quả kém với không gian tìm kiếm lớn**: Mặc dù thuật toán BA có thể đạt được hiệu quả tốt trong các không gian tìm kiếm nhỏ hoặc vừa, nhưng khi không gian tìm kiếm trở nên quá lớn (ví dụ như trong các bài toán đa chiều), thuật toán có thể gặp khó khăn trong việc tìm kiếm giải pháp tối ưu. Sự tăng trưởng theo cấp số nhân của không gian tìm kiếm có thể dẫn đến việc dơi phải bay qua rất nhiều vùng không cần thiết, làm giảm tốc độ hội tụ và gia tăng thời gian tính toán.
* **Tính ngẫu nhiên bất ổn**: Do tính ngẫu nhiên của thuật toán, kết quả của BA có thể không ổn định giữa các lần chạy. Dơi có thể có các quỹ đạo bay khác nhau trong các lần chạy khác nhau, điều này có thể dẫn đến sự thay đổi trong kết quả cuối cùng, ngay cả khi các tham số khởi tạo là giống nhau. Việc này có thể làm cho thuật toán khó đoán được và không thể đảm bảo kết quả ổn định.
* **Chi phí tính toán cao**: Khi số lượng cá thể trong quần thể dơi tăng lên, chi phí tính toán của thuật toán cũng sẽ tăng theo. Sự gia tăng số lượng cá thể dẫn đến việc phải tính toán nhiều phép toán hơn, đặc biệt khi không gian tìm kiếm có kích thước lớn. Điều này có thể khiến thuật toán trở nên chậm và không khả thi khi phải xử lý một số lượng lớn các cá thể trong một không gian tìm kiếm rộng.
* **Hạn chế trong bài toán rời rạc**: Mặc dù thuật toán BA rất mạnh trong các bài toán tối ưu hóa liên tục, nhưng nó gặp phải một số khó khăn khi áp dụng vào các bài toán rời rạc. Vì dơi trong thuật toán di chuyển trong không gian liên tục, nên khi áp dụng cho các bài toán rời rạc, thuật toán cần phải được điều chỉnh để xử lý các giá trị không liên tục hoặc các ràng buộc trong bài toán. Việc này có thể yêu cầu thêm các bước xử lý phức tạp và làm giảm hiệu quả của thuật toán.

## 5. Kết Luận

Thuật toán BA (Bat Algorithm) là một phương pháp tối ưu hóa tự nhiên lấy cảm hứng từ hành vi của loài dơi, đặc biệt là cách dơi bay và tìm kiếm con mồi. Thuật toán này đã chứng tỏ được hiệu quả trong việc giải quyết nhiều bài toán tối ưu hóa phức tạp, với khả năng cân bằng giữa việc khám phá không gian tìm kiếm và khai thác các vùng đã tìm thấy giải pháp tốt. Sự linh hoạt trong cách điều chỉnh các tham số và khả năng áp dụng cho nhiều bài toán khác nhau, từ liên tục đến rời rạc và đa mục tiêu, khiến thuật toán BA trở thành một công cụ mạnh mẽ trong lĩnh vực tối ưu hóa.

Một trong những điểm mạnh của BA là khả năng hội tụ nhanh, giúp thuật toán đạt được giải pháp tối ưu trong thời gian ngắn. Bên cạnh đó, không giống như các thuật toán truyền thống, BA không yêu cầu thông tin gradient, làm cho nó trở thành lựa chọn lý tưởng cho các bài toán không khả vi hoặc phi tuyến. Tuy nhiên, BA cũng có một số nhược điểm như hội tụ sớm, phụ thuộc vào việc chọn tham số khởi tạo, và hiệu quả kém trong các không gian tìm kiếm lớn hoặc các bài toán rời rạc.

Dù vậy, với các cải tiến về tham số và cơ chế điều chỉnh trong quá trình tối ưu hóa, thuật toán BA có thể được điều chỉnh để giải quyết các vấn đề trong các bài toán tối ưu hóa đa dạng. Việc ứng dụng thuật toán BA trong thực tế đòi hỏi người sử dụng phải hiểu rõ tính chất của bài toán và điều chỉnh thuật toán sao cho phù hợp với yêu cầu cụ thể. Tổng thể, BA là một thuật toán tối ưu hóa có tính linh hoạt cao, dễ triển khai, và có thể giải quyết nhiều bài toán khó mà các thuật toán tối ưu hóa khác gặp khó khăn.