TRƯỜNG ĐẠI HỌC CÔNG NGHỆ GTVT
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO BÀI TẬP LỚN
TRÍ TUỆ NHÂN TẠO

Tên đề tài:
FIREFLY ALGORITHM & CUCKOO SEARCH ALGORITHM

GIẢNG VIÊN HƯỚNG DẪN: **ĐỖ BẢO SƠN**

NHÓM: **10**

SINH VIÊN THỰC HIỆN: **1. QUANG HỒNG ÁNH SỨ**
2. NGUYỄN TIẾN HƯNG
3. NGÔ THỊ THANH VÂN

LỚP: **73DCTT23**

HÀ NỘI 13-12-2024

Nhận Xét Đánh Giá Của Giảng Viên

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

Hà Nội, Ngày 13, Tháng 12, Năm 2024

Giảng viên hướng dẫn

Đỗ Bảo Sơn

Bảng Phân Công Công Việc

STT	Họ và Tên	Mã Sinh Viên	Công Việc	Mức Độ Đóng Góp	Ghi Chú
1	Quang Hồng Ánh Sứ	73DCTT22466	Chương I	**33.3%**	TN
2	Ngô Thị Thanh Vân	73DCTT22248	Chương II	**33.3%**	
3	Nguyễn Tiến Hưng	73DCTT23310	Chương II	**33.3%**	

LỜI NÓI ĐẦU

Trong những năm gần đây, với sự phát triển mạnh mẽ của các lĩnh vực khoa học máy tính, trí tuệ nhân tạo và tối ưu hóa, các thuật toán tối ưu hóa toàn cục đã trở thành công cụ quan trọng trong việc giải quyết những bài toán phức tạp mà các phương pháp truyền thống không thể xử lý hiệu quả. Các thuật toán tối ưu hóa toàn cục dựa trên các chiến lược tìm kiếm thông minh, giúp tìm ra các nghiệm tối ưu trong không gian tìm kiếm rộng lớn và có thể đối mặt với các vấn đề không xác định rõ ràng hay không có một giải pháp tối ưu đơn giản. Một trong những hướng đi thú vị và đầy tiềm năng trong nghiên cứu thuật toán tối ưu hóa là việc áp dụng các mô hình tự nhiên, đặc biệt là các hành vi của động vật và các hiện tượng tự nhiên để phát triển các thuật toán tối ưu hóa hiệu quả.

Trong đó, **Firefly Algorithm** (Thuật toán Nhấp Nháy của Đom Đóm) và **Cuckoo Search Algorithm** (Thuật toán Tìm Kiếm Chim Cúc Cu) là hai trong số những thuật toán tối ưu hóa tự nhiên nổi bật. Firefly Algorithm được đề xuất bởi Yang vào năm 2008, lấy cảm hứng từ hành vi của loài đom đóm, trong đó việc phát sáng của chúng được sử dụng để giao tiếp và thu hút những con đom đóm khác, giúp hình thành một chiến lược tìm kiếm thông minh trong không gian giải pháp. Các đom đóm với cường độ ánh sáng mạnh sẽ thu hút các đom đóm khác, mô phỏng quá trình tìm kiếm và tối ưu hóa bằng cách di chuyển về phía các giải pháp tốt hơn trong không gian tìm kiếm. Với khả năng tìm kiếm hiệu quả và phạm vi ứng dụng rộng rãi, thuật toán này đã được áp dụng thành công trong nhiều bài toán tối ưu hóa khác nhau, từ tối ưu hóa chức năng đến các bài toán trong các lĩnh vực khoa học và kỹ thuật.

Tương tự, Cuckoo Search Algorithm được đề xuất bởi Yang và Deb vào năm 2009, mô phỏng hành vi sinh sản của loài chim cúc cu. Chim cúc cu nổi tiếng với chiến lược đẻ trứng trong tổ của loài chim khác, và thuật toán này áp dụng phương pháp này để phát triển các giải pháp tối ưu. Cuckoo Search Algorithm dựa trên việc khai thác tính ngẫu nhiên và tính hiệu quả trong việc chọn lọc các tổ chim, giúp tối ưu hóa các bài toán có không gian tìm kiếm phức tạp. Với ưu điểm là khả năng tránh bị mắc kẹt trong các cực trị địa phương và tìm kiếm được các nghiệm tối ưu toàn cục, thuật toán này đã được áp dụng thành công trong nhiều lĩnh vực như tối ưu hóa mạng, tối ưu hóa thiết kế và học máy.

Bài báo cáo này nhằm mục đích trình bày một cách tổng quan về hai thuật toán Firefly và Cuckoo Search, bao gồm nguyên lý hoạt động cơ bản, các phương pháp cải tiến và ứng dụng thực tế của chúng. Cụ thể, báo cáo sẽ giới thiệu về quá trình hình thành và phát triển của hai thuật toán này, cách chúng mô phỏng các hành vi tự nhiên của đom đóm và chim cúc cu để giải quyết các bài toán tối ưu hóa. Đồng thời, chúng tôi cũng sẽ đánh giá hiệu quả của hai thuật toán này qua các ví dụ minh họa và so sánh chúng với một số thuật toán tối ưu hóa khác trong các tình huống cụ thể. Mục đích cuối cùng của báo cáo là giúp người đọc hiểu rõ hơn về các đặc điểm của Firefly Algorithm và Cuckoo Search Algorithm, từ đó có thể áp dụng chúng một cách linh hoạt và hiệu quả trong các bài toán tối ưu hóa thực tế.

Hy vọng bài báo cáo này sẽ cung cấp những thông tin hữu ích và góp phần làm rõ hơn về tiềm năng của các thuật toán tối ưu hóa tự nhiên, từ đó mở ra các hướng nghiên cứu và ứng dụng mới trong tương lai.

MỤC LỤC

LỜI NÓI ĐẦU 4
CHƯƠNG 1: FIREFLY ALGORITHM (THUẬT TOÁN ĐOM ĐÓM) 6
I. GIỚI THIỆU 6
II. ĐỘNG LỰC ĐỀ XUẤT THUẬT TOÁN FIREFLY ALGORITHM 7
1. Hành vi phát sáng của đom đóm trong tự nhiên 7
2. Khả năng tối ưu hóa 7
3. Lý thuyết bầy đàn và tính tự tổ chức 7
4. Khả năng áp dụng và linh hoạt 7
II. MỤC ĐÍCH 8
III. NỘI DUNG VÀ CÁC BƯỚC CỦA THUẬT TOÁN FIREFLY ALGORITHM (FA) 9
1. Nội Dung 9
2. Các bước thực hiện 10
IV. THUẬT TOÁN 11
1. Ví dụ về bài toán tối ưu: 11
2. Thực nghiệm thuật toán đom đóm: 12
3. Đánh giá kết quả: 12
V. KẾT LUẬN 13
CHƯƠNG 2: CUCKOO SEARCH ALGORITHM 14
I. GIỚI THIỆU 14
II. ĐỘNG LỰC ĐỀ XUẤT THUẬT TOÁN CUCKOO SEARCH 15
1. Hành vi sinh sản độc đáo của loài chim cúc cu 15
2. Khả năng tìm kiếm hiệu quả thông qua bước Levy Flight 15
3. Khắc phục hạn chế của các thuật toán tối ưu hóa truyền thống 15
4. Tính đơn giản và linh hoạt 16
5. Hiệu quả trong nhiều lĩnh vực 16
III. MỤC ĐÍCH 16
1. Tìm kiếm giải pháp tối ưu (Optimization) 16
2. Khắc phục cực trị địa phương (Avoiding Local Optima) 17
3. Cân bằng giữa khai thác và thăm dò (Exploitation vs. Exploration) 17
4. Giải quyết bài toán phi tuyến tính và không liên tục 18
5. Đơn giản hóa và khả năng mở rộng (Simplicity and Scalability) 18
III. CÁC BƯỚC CỦA THUẬT TOÁN CUCKOO SEARCH (CS) 18
IV. THUẬT TOÁN CUCKOO SEARCH 19
1. Phân tích bài toán 19
2. Thuật toán 20
3. Các bước thực hiện thuật toán 20
V. KẾT LUẬN 22

CHƯƠNG 1: FIREFLY ALGORITHM (THUẬT TOÁN ĐOM ĐÓM)

I. GIỚI THIỆU

Thuật toán Firefly (FA) là một thuật toán tối ưu hóa bầy đàn (swarm intelligence) được Xin-She Yang đề xuất vào năm 2008. Đây là một thuật toán lấy cảm hứng từ hành vi phát sáng của đom đóm trong tự nhiên. Đặc điểm nổi bật của thuật toán Firefly là khả năng tìm kiếm và tối ưu hóa trong các không gian phức tạp, không cần thông tin về đạo hàm và có thể xử lý hiệu quả các bài toán tối ưu hóa đa mục tiêu, bài toán NP-hard và các vấn đề tối ưu hóa trong các hệ thống không gian tìm kiếm rộng lớn.

Thuật toán này dựa trên cơ chế phát sáng của đom đóm: mỗi đom đóm trong không gian tìm kiếm đại diện cho một giải pháp, và ánh sáng của nó thể hiện chất lượng của giải pháp đó. Những đom đóm sáng hơn (tức là những giải pháp tốt hơn) sẽ thu hút các đom đóm kém sáng hơn, tạo ra động lực di chuyển về hướng các giải pháp tối ưu.

Để tìm ra giải pháp cho bất kỳ vấn đề tối ưu hóa nào, chúng ta có thể sử dụng các thuật toán tối ưu hóa thông thường như phương pháp leo đồi và phương pháp simplex, hoặc các phương pháp heuristic như thuật toán di truyền hoặc các kết hợp thích hợp của chúng. Các thuật toán meta-heuristic hiện đại đang trở nên mạnh mẽ trong việc giải quyết các vấn đề tối ưu hóa toàn, đặc biệt là đối với các vấn đề NP-khó như vấn đề người bán hàng du lịch. Ví dụ, tối ưu hóa bầy đàn hạt (PSO) được Kennedy và Eberhart phát triển vào năm 1995, dựa trên hành vi bầy đàn như cá và chim tụ tập trong tự nhiên. Hiện nay, nó đã được áp dụng để tìm giải pháp cho nhiều ứng dụng tối ưu hóa. Một ví dụ khác là Thuật toán Firefly do tác giả đầu tiên phát triển đã chứng minh được tính ưu việt đầy hứa hẹn so với nhiều thuật toán khác. Các chiến lược tìm kiếm trong các thuật toán đa tác nhân này là ngẫu nhiên hóa có kiểm soát và khai thác các giải pháp tốt nhất. Tuy nhiên, ngẫu nhiên hóa như vậy thường sử dụng phân phối đồng đều hoặc phân phối Gauss. Trên thực tế, kể từ khi PSO được phát triển, khá nhiều thuật toán đã được phát triển và chúng có thể vượt trội hơn PSO theo nhiều cách khác nhau.

Mặt khác, luôn có một số bất định và nhiễu liên quan đến tất cả các vấn đề tối ưu hóa trong thế giới thực. Do đó, các hàm mục tiêu có thể có nhiễu và các ràng buộc cũng có thể có nhiễu ngẫu nhiên. Trong trường hợp này, một vấn đề tối ưu hóa tiêu chuẩn trở thành một vấn đề tối ưu hóa ngẫu nhiên. Các phương pháp hoạt động tốt cho các vấn đề tối ưu hóa tiêu chuẩn không thể được áp dụng trực tiếp cho tối ưu hóa ngẫu nhiên; nếu không, các kết quả thu được là không chính xác hoặc thậm chí vô nghĩa. Các vấn đề tối ưu hóa phải được xây dựng lại đúng cách hoặc các thuật toán tối ưu hóa phải được sửa đổi cho phù hợp, mặc dù trong hầu hết các trường hợp, chúng ta phải thực hiện cả hai.

II. ĐỘNG LỰC ĐỀ XUẤT THUẬT TOÁN FIREFLY ALGORITHM

Thuật toán Firefly được phát triển dựa trên những động lực sau:

1. Hành vi phát sáng của đom đóm trong tự nhiên

Một trong những yếu tố quan trọng nhất dẫn đến sự phát triển của thuật toán Firefly là hành vi phát sáng đặc trưng của đom đóm trong tự nhiên. Đom đóm sử dụng ánh sáng của mình để thu hút các đom đóm khác, chủ yếu phục vụ cho mục đích giao phối. Mỗi đom đóm phát sáng với cường độ khác nhau, tùy thuộc vào các yếu tố như loài, môi trường và nhu cầu giao phối. Cường độ ánh sáng có thể coi là một chỉ báo về chất lượng hoặc "fitness" của đom đóm trong không gian tìm kiếm. Vì vậy, các đom đóm sáng hơn sẽ thu hút các đom đóm ít sáng hơn, làm chúng di chuyển về phía các giải pháp tối ưu hơn.

2. Khả năng tối ưu hóa

Một yếu tố quan trọng khác là nhu cầu tìm kiếm các phương pháp tối ưu hóa mới trong các bài toán khó và phức tạp mà các thuật toán truyền thống gặp phải khó khăn. Các phương pháp như Gradient Descent (tìm kiếm theo gradient) thường không hiệu quả trong những không gian tìm kiếm lớn và không liên tục, hoặc khi bài toán không có cấu trúc lồi (non-convex). Hơn nữa, nhiều bài toán tối ưu hóa hiện nay (như tối ưu hóa đa mục tiêu, hoặc bài toán NP-hard) đòi hỏi một phương pháp tổng quát, không dựa vào đạo hàm để giải quyết. Firefly Algorithm, với khả năng di chuyển của các đom đóm trong không gian tìm kiếm dựa trên độ sáng của chúng, có thể giúp giải quyết những vấn đề này.

3. Lý thuyết bầy đàn và tính tự tổ chức

Firefly Algorithm là một ví dụ điển hình của các thuật toán tối ưu hóa bầy đàn, dựa trên nguyên lý tự tổ chức trong thiên nhiên. Cơ chế này cho phép thuật toán tìm ra các giải pháp tối ưu mà không cần sự can thiệp từ bên ngoài. Các đom đóm trong bầy không cần biết về vị trí của các đom đóm khác ngoài việc quan sát ánh sáng phát ra từ chúng. Điều này giúp thuật toán có thể khám phá không gian tìm kiếm một cách đồng đều và hiệu quả mà không gặp phải vấn đề "mắc kẹt" ở các cực trị cục bộ, điều mà nhiều thuật toán tối ưu hóa khác có thể gặp phải.

4. Khả năng áp dụng và linh hoạt

Thuật toán Firefly không phụ thuộc vào thông tin về đạo hàm của bài toán tối ưu và có thể được áp dụng linh hoạt cho nhiều loại bài toán tối ưu hóa khác nhau, bao gồm nhưng không giới hạn ở:

- Tối ưu hóa hàm số (single-objective optimization)
- Tối ưu hóa đa mục tiêu (multi-objective optimization)
- Các bài toán tối ưu hóa trong các hệ thống phân tán hoặc các vấn đề tối ưu hóa trong thực tế.

Động lực này thúc đẩy việc phát triển Firefly Algorithm như một công cụ mạnh mẽ và dễ dàng áp dụng cho nhiều bài toán phức tạp trong lĩnh vực khoa học và công nghệ.

II. MỤC ĐÍCH

Mục đích của thuật toán **Firefly Algorithm (FA)** là **tìm kiếm tối ưu** cho các bài toán tối ưu hóa, đặc biệt là các bài toán có không gian tìm kiếm lớn và phức tạp. Thuật toán này mô phỏng hành vi của các con đom đóm trong tự nhiên, trong đó các con đom đóm thu hút nhau bằng cách phát ra ánh sáng và điều chỉnh vị trí của mình để tìm kiếm tối ưu. Dưới đây là một số mục đích chính và ứng dụng của thuật toán Firefly: ❖ Tối Ưu Hóa Toán Học

- Thuật toán Firefly được thiết kế để giải quyết các bài toán tối ưu hóa, tức là tìm ra điểm tối ưu (tối thiểu hoặc tối đa) của một hàm mục tiêu.
- Thông qua việc mô phỏng hành vi di chuyển của đom đóm, thuật toán này tìm kiếm trong không gian giải pháp để xác định giá trị tối ưu cho các bài toán có hàm mục tiêu phức tạp.

❖ Giải Quyết Các Bài Toán Có Không Gian Tìm Kiếm Lớn

Thuật toán Firefly có thể áp dụng cho các bài toán tối ưu hóa có không gian tìm kiếm rất lớn, nơi các phương pháp tìm kiếm thông thường khó khăn hoặc không hiệu quả. Đặc biệt, Firefly có khả năng tìm kiếm một cách hiệu quả trong không gian có nhiều cực trị cục bộ và không đều.

❖ Tránh Tối Ưu Cục Bộ

Một trong những ưu điểm của thuật toán Firefly là khả năng giúp tránh bị mắc kẹt trong các cực trị cục bộ. Thông qua cơ chế ánh sáng, các đom đóm "tự động" điều chỉnh hành vi của mình để tránh rơi vào các cực trị không tối ưu.

Điều này giúp thuật toán có khả năng tìm kiếm tối ưu toàn cục, đặc biệt trong các bài toán có nhiều cực trị cục bộ.

❖ Ứng Dụng Trong Các Lĩnh Vực Khác Nhau

Hệ thống tối ưu hóa trong công nghiệp và kỹ thuật: Tối ưu hóa các tham số trong thiết kế kỹ thuật, điều khiển hệ thống, v.v.

Tối ưu hóa trong học máy (machine learning): Firefly Algorithm có thể được sử dụng để tối ưu hóa các siêu tham số của các mô hình học máy, giúp cải thiện độ chính xác của mô hình.

Quản lý tài nguyên và phân phối: Các bài toán tối ưu hóa về quản lý tài nguyên, phân phối sản phẩm, v.v., cũng có thể được giải quyết bằng thuật toán này.

Hệ thống mạng và truyền thông: Tối ưu hóa các tham số trong mạng lưới, ví dụ như cân bằng tải trong mạng, phân bổ tài nguyên trong hệ thống mạng viễn thông, v.v.

Tối ưu hóa trong thiết kế mạng neuron (neural network): FA có thể được áp dụng trong tối ưu hóa trọng số của mạng neuron, điều này có thể giúp cải thiện hiệu quả học của mạng.

❖ Tối Ưu Hóa Các Tham Số Phi Tuyến

Các bài toán tối ưu hóa với các hàm mục tiêu phi tuyến (nonlinear) và không liên tục (discontinuous) là một trong những vấn đề mà thuật toán Firefly có thể xử lý tốt hơn so với các phương pháp truyền thống như **Gradient Descent** hay **Simulated Annealing**.

❖ **Khả Năng Mở Rộng và Điều Chỉnh Dễ Dàng**

Thuật toán Firefly có thể được điều chỉnh để thích nghi với các bài toán tối ưu khác nhau, với khả năng điều chỉnh các tham số như tốc độ di chuyển (alpha), cường độ ánh sáng (beta), và tham số hấp dẫn giữa các đom đóm (gamma).

Điều này giúp thuật toán linh hoạt trong việc giải quyết nhiều bài toán tối ưu khác nhau với các yêu cầu và điều kiện khác nhau.

III. NỘI DUNG VÀ CÁC BƯỚC CỦA THUẬT TOÁN FIREFLY ALGORITHM (FA)

1. Nội Dung

Thuật toán Firefly là một thuật toán tối ưu hóa bầy đàn, trong đó mỗi cá thể (đom đóm) di chuyển trong không gian tìm kiếm dựa trên sự hấp dẫn của ánh sáng. Ánh sáng của mỗi đom đóm phản ánh chất lượng của giải pháp mà nó đại diện, và đom đóm sáng hơn (tốt hơn) sẽ thu hút những đom đóm kém sáng hơn về phía mình.

Thuật toán Firefly mô phỏng hành vi của đom đóm trong tự nhiên, trong đó mỗi đom đóm đại diện cho một giải pháp trong không gian tìm kiếm và độ sáng của chúng được tính theo giá trị của hàm mục tiêu. Đom đóm sáng hơn sẽ thu hút các đom đóm kém sáng hơn, và quá trình di chuyển sẽ tiếp tục cho đến khi tìm được giải pháp tối ưu.

Để đơn giản khi mô tả thuật toán, chúng tôi sử dụng ba quy tắc lý tưởng sau:

a) tất cả đom đóm đều là phi giới tính nên một con đom đóm sẽ bị thu hút bởi những con đom đóm khác bất kể giới tính của chúng;

b) Độ hấp dẫn tỷ lệ thuận với độ sáng của chúng, do đó đối với bất kỳ hai con đom đóm nhấp nháy nào, con kém sáng hơn sẽ di chuyển về phía con sáng hơn. Độ hấp dẫn tỷ lệ thuận với độ sáng và cả hai đều giảm khi khoảng cách của chúng tăng lên. Nếu không có con nào sáng hơn một con đom đóm cụ thể, nó sẽ di chuyển ngẫu nhiên;

c) Độ sáng của đom đóm bị ảnh hưởng hoặc xác định bởi cảnh quan của hàm mục tiêu. Đối với một bài toán tối đa hóa, độ sáng có thể chỉ đơn giản là tỷ lệ thuận với giá trị của các hàm mục tiêu.

Trong thuật toán đom đóm, có hai vấn đề quan trọng: sự thay đổi cường độ ánh sáng và công thức của sự hấp dẫn. Để đơn giản, chúng ta luôn có thể giả định rằng sự hấp dẫn của đom đóm được xác định bởi độ sáng của nó, độ sáng này lại liên quan đến hàm mục tiêu được mã hóa.

Trong trường hợp đơn giản nhất đối với các vấn đề tối ưu hóa tối đa, độ sáng I của một con đom đóm tại một vị trí **x** cụ thể có thể được chọn là **I(x) f(x).** Tuy nhiên, sức hấp dẫn β là tương đối, nó phải được nhìn thấy trong mắt người nhìn hoặc được đánh giá bởi những con đom đóm khác.

Do đó, nó sẽ thay đổi theo khoảng cách r_{ij} giữa đom đóm i và đom đóm j. Ngoài ra, cường độ ánh sáng giảm theo khoảng cách từ nguồn sáng và ánh sáng cũng được hấp thụ trong môi trường, vì vậy chúng ta nên cho phép sức hấp dẫn thay đổi theo mức độ hấp thụ. Ở dạng đơn giản nhất, cường độ ánh sáng I(r) thay đổi theo định luật nghịch đảo bình phương $I(r) = I_S / r^2$ trong đó Is là cường độ tại nguồn sáng.

Đối với một môi trường nhất định có hệ số hấp thụ ánh sáng y cố định cường độ ánh sáng I thay đổi theo khoảng cách r. Nghĩa là:

$$I = I_0 e^{-\gamma r}$$

Trong đó I0 là cường độ ánh sáng ban đầu.

Vì sức hấp dẫn của đom đóm tỉ lệ thuận với cường độ ánh sáng mà đom đóm bên cạnh nhìn thấy, nên giờ đây chúng ta có thể định nghĩa sức hấp dẫn β của đom đóm bằng

$$\beta = \beta_0 e^{\hat{}}(-\gamma r^2)$$

Trong đó β0 là sức hấp dẫn tại r = 0.

2. Các bước thực hiện

Dưới đây là các bước cơ bản trong thuật toán Firefly:

B1.Khởi tạo Bầy Đom Đóm:

- Khởi tạo một bầy đom đóm ngẫu nhiên trong không gian tìm kiếm. Mỗi đom đóm đại diện cho một giải pháp khả thi trong bài toán tối ưu.
- Tính toán độ sáng (hay fitness) của mỗi đom đóm dựa trên hàm mục tiêu của bài toán.

B2.Tính Toán Độ Sáng (Fitness):

- Độ sáng của mỗi đom đóm được tính từ giá trị hàm mục tiêu tại vị trí của nó. Giải pháp càng tốt, độ sáng càng cao.
- Độ sáng có thể được tính bằng công thức: $I_i = f(x_i)$,

  trong đó $f(x_i)$ là giá trị hàm mục tiêu tại vị trí x_i của đom đóm thứ i.

B3.Cập Nhật Vị Trí của Đom Đóm:

- Mỗi đom đóm sẽ di chuyển đến một vị trí mới theo công thức:

  $$x_i(t+1) = x_i(t) + \beta \cdot (x_j(t) - x_i(t)) + \alpha \cdot (\text{rand}() - 0.5)$$

  Trong đó:

  - $x_i(t)$ là vị trí của đom đóm i tại vòng lặp t.
  - $x_j(t)$ là vị trí của đom đóm j (là đom đóm sáng hơn).
  - β là tham số hấp dẫn (tính theo độ sáng của đom đóm sáng hơn).
  - α là tham số ngẫu nhiên để tạo sự đa dạng trong quá trình tìm kiếm.
  - rand()là một số ngẫu nhiên trong khoảng [-0.5, 0.5].
- Đom đóm sẽ di chuyển về phía đom đóm sáng hơn, nhưng cũng có một yếu tố ngẫu nhiên để tránh bị mắc kẹt trong các cực trị cục bộ.

B4.Cập Nhật Độ Sáng:

- Sau khi cập nhật vị trí, tính lại độ sáng của đom đóm ở vị trí mới. Nếu độ sáng của đom đóm mới tốt hơn, vị trí sẽ được giữ lại.

B5.Kiểm Tra Điều Kiện Dừng:

- Thuật toán sẽ tiếp tục cập nhật vị trí các đom đóm cho đến khi đạt được một trong các điều kiện dừng, ví dụ như:
  - Đạt số vòng lặp tối đa.
  - Không có sự thay đổi đáng kể trong chất lượng giải pháp.
  - Đã tìm được giải pháp tối ưu.

⇨ **Kết Quả:**

- Sau khi thuật toán dừng lại, giải pháp tốt nhất (tức là đom đóm có độ sáng cao nhất) sẽ được trả về.

IV. THUẬT TOÁN

Hàm mục tiêu $f_p(\mathbf{x})$, $\mathbf{x} = (x_1,\ldots,x_d)^T$

Quần thể đom đóm ban đầu x_i (i = 1, ..., n)

Cường độ ánh sáng I_i tại x_i được xác định bởi $f_p(\mathbf{x}_i)$

Xác định hệ số hấp thụ ánh sáng γ

while (t <MaxGeneration)

For i = 1 : n tất cả n đom đóm

For j = 1 : i tất cả n đom đóm

If ($I_j > I_i$)

Di chuyển đom đóm i về phía j (chiều d)

End if

Thay đổi β thông qua exp[-γr]

Đánh giá các giải pháp mới và cập nhật

End for j

End for i

Xếp hạng các con đom đóm và tìm ra kết thúc tốt nhất hiện tại

End while

Kết quả

● Ứng dụng và Đánh giá Kết quả:

Để thực nghiệm thuật toán đom đóm, bạn cần một bài toán tối ưu cụ thể. Ví dụ, một bài toán tối ưu hàm mục tiêu, tối ưu các tham số trong một mô hình học máy hoặc bài toán tối ưu hóa chức năng thực nghiệm.

1. Ví dụ về bài toán tối ưu:

Giả sử bạn cần tối ưu hóa một hàm số đơn giản, ví dụ hàm Ackley, có dạng:

$$f(x,y) = -20\,.\,epx(-0.2\,.\,\sqrt{0.5\,.\,(x^2 + y^2)}) - epx\,(0.5\,.\,(\cos(2\pi x)) + \cos(2\pi x))) + e + 20$$

Đây là một bài toán tối ưu với một cực tiểu toàn cục tại điểm (0,0).

2. Thực nghiệm thuật toán đom đóm:

- **Khởi tạo quần thể**: Chọn một số lượng đom đóm ngẫu nhiên, ví dụ 50 đom đóm.
- **Hàm mục tiêu**: Sử dụng hàm Ackley làm hàm mục tiêu để đánh giá chất lượng mỗi đom đóm.
- **Cập nhật vị trí**: Di chuyển các đom đóm về phía những đom đóm có giá trị ánh sáng tốt hơn (tức là hàm mục tiêu nhỏ hơn).
- **Điều chỉnh tham số**: Thử nghiệm với các tham số như hệ số hấp dẫn, tỷ lệ di chuyển ngẫu nhiên, và số vòng lặp.

Kết quả:

- Sau khi thực hiện thuật toán, kiểm tra xem thuật toán có tìm được giá trị tối ưu (hoặc gần tối ưu) của hàm Ackley không.
- So sánh kết quả với các thuật toán khác như Thuật toán Di truyền (Genetic Algorithm), Thuật toán tối ưu bầy đàn (PSO), hoặc thuật toán Gradient Descent.
- Đánh giá thời gian chạy, chất lượng của kết quả (sai số so với giá trị tối ưu) và tính ổn định của thuật toán.

3. Đánh giá kết quả:

- **Chất lượng kết quả**: Đo lường sai số giữa giá trị tối ưu tìm được và giá trị tối ưu lý thuyết.
- **Thời gian chạy**: Đo thời gian tính toán cần thiết để đạt được kết quả, đặc biệt trong các bài toán lớn.
- **Khả năng hội tụ**: Đánh giá độ ổn định của thuật toán trong việc tìm kiếm giá trị tối ưu (ví dụ, chạy thuật toán nhiều lần và so sánh kết quả).

● Ví dụ mã giả cho thuật toán đom đóm:

```
import numpy as np

# Ham muc tiêu Ackley
def ham_ackley(x, y):
    return -20 * np.exp(-0.2 * np.sqrt(0.5 * (x**2 + y**2))) - np.exp(0.5 * (np.cos(2 * np.pi * x) + np.cos(2 * np.pi * y))) + np.e + 20

# Thuat toan dom dom
def thuat_toan_dom_dom(soluong_dom_dom, so_vong_lap, alpha, beta, gamma):
    # Khoi tao vi tri cua cac dom dom  ngau nhien
    dom_doms = np.random.uniform(-5, 5, (soluong_dom_dom, 2))
    # Tính toán do sáng cua các dom dóm (dánh giá hàm muc tiêu)
    do_sang = np.array([ham_ackley(dom_dom[0], dom_dom[1]) for dom_dom in dom_doms])

    # lap qua các vong lap
    for _ in range(so_vong_lap):
        for i in range(soluong_dom_dom):
            for j in range(soluong_dom_dom):
                # Neu dom dom i ít sáng hon dom dom j
                if do_sang[i] > do_sang[j]:
                    # Tính khoang cách giua 2 dom dóm
                    r = np.linalg.norm(dom_doms[i] - dom_doms[j])
                    # Cap nhat vi trí cua dom dóm i
                    dom_doms[i] = dom_doms[i] + beta * np.exp(-gamma * r**2) * (dom_doms[j] - dom_doms[i]) + alpha * np.random.uniform(-1, 1, 2)

        # Cap nhat lai do sáng sau khi thay doi vi trí
        do_sang = np.array([ham_ackley(dom_dom[0], dom_dom[1]) for dom_dom in dom_doms])

    # Tìm dom dom có do sáng tot nhat (chi so toi uu)
    chi_so_toi_uu = np.argmin(do_sang)
    return dom_doms[chi_so_toi_uu], do_sang[chi_so_toi_uu]

# Vi du su dung thuat toán
vi_tri_toi_uu, gia_tri_toi_uu = thuat_toan_dom_dom(soluong_dom_dom=50, so_vong_lap=1000, alpha=0.2, beta=1, gamma=1)

# In ket qua
print(f"Vi trí toi uu: {vi_tri_toi_uu}, Giá tri toi uu: {gia_tri_toi_uu}")
```

Thuật toán đom đóm có khả năng tối ưu hóa hiệu quả cho nhiều loại bài toán khác nhau. Tuy nhiên, kết quả thực nghiệm và hiệu quả của thuật toán có thể thay đổi tùy

thuộc vào các tham số điều chỉnh như kích thước quần thể, số vòng lặp, và các tham số alpha, beta, gamma. Việc điều chỉnh và đánh giá thuật toán dựa trên các bài toán cụ thể sẽ cho phép đánh giá chính xác khả năng của thuật toán này trong thực tế.

V. KẾT LUẬN

Thuật toán Firefly Algorithm (FA) là một phương pháp tối ưu hóa mạnh mẽ, được lấy cảm hứng từ hành vi tự nhiên của các con đom đóm, trong đó các đom đóm sử dụng ánh sáng để thu hút và di chuyển về phía nhau. Mục tiêu chính của thuật toán là tìm kiếm giải pháp tối ưu trong các bài toán tối ưu hóa phức tạp, đặc biệt là những bài toán có không gian tìm kiếm rộng lớn, phi tuyến hoặc có nhiều cực trị cục bộ. Với cơ chế di chuyển linh hoạt và sự hấp dẫn giữa các đom đóm, thuật toán có khả năng tránh mắc kẹt ở các cực trị cục bộ và tìm ra các giải pháp tối ưu toàn cục. Firefly có thể được áp dụng rộng rãi trong nhiều lĩnh vực như học máy, kỹ thuật, tối ưu hóa mạng, tối ưu hóa tài nguyên, và thiết kế hệ thống phức tạp.

Ưu điểm của thuật toán này là khả năng hội tụ nhanh trong các bài toán tối ưu hóa không có cấu trúc rõ ràng và có thể xử lý các hàm mục tiêu phức tạp hoặc không đều. Tuy nhiên, một trong những hạn chế của Firefly là thời gian chạy có thể dài đối với các bài toán phức tạp hoặc khi kích thước không gian tìm kiếm rất lớn. Điều này có thể làm giảm hiệu quả trong các tình huống yêu cầu thời gian tính toán nhanh. Bên cạnh đó, tinh chỉnh tham số (như tốc độ di chuyển, độ sáng và tham số hấp dẫn) là yếu tố quan trọng để thuật toán hoạt động hiệu quả. Nếu không lựa chọn đúng tham số, thuật toán có thể không đạt được kết quả tối ưu như mong muốn.

Mặc dù vậy, Firefly Algorithm vẫn là một công cụ tối ưu hóa mạnh mẽ và linh hoạt, có thể áp dụng hiệu quả cho các bài toán phức tạp trong các lĩnh vực đa dạng. Với khả năng tìm kiếm tối ưu toàn cục và tránh các cực trị cục bộ, thuật toán này chứng minh được giá trị của mình trong việc giải quyết các vấn đề tối ưu hóa, đồng thời cung cấp một phương pháp hiệu quả trong môi trường không có nhiều thông tin về hàm mục tiêu hoặc khi không gian tìm kiếm quá lớn.

CHƯƠNG 2: CUCKOO SEARCH ALGORITHM (THUẬT TOÁN TÌM KIẾM CHIM BỒ CÂU)

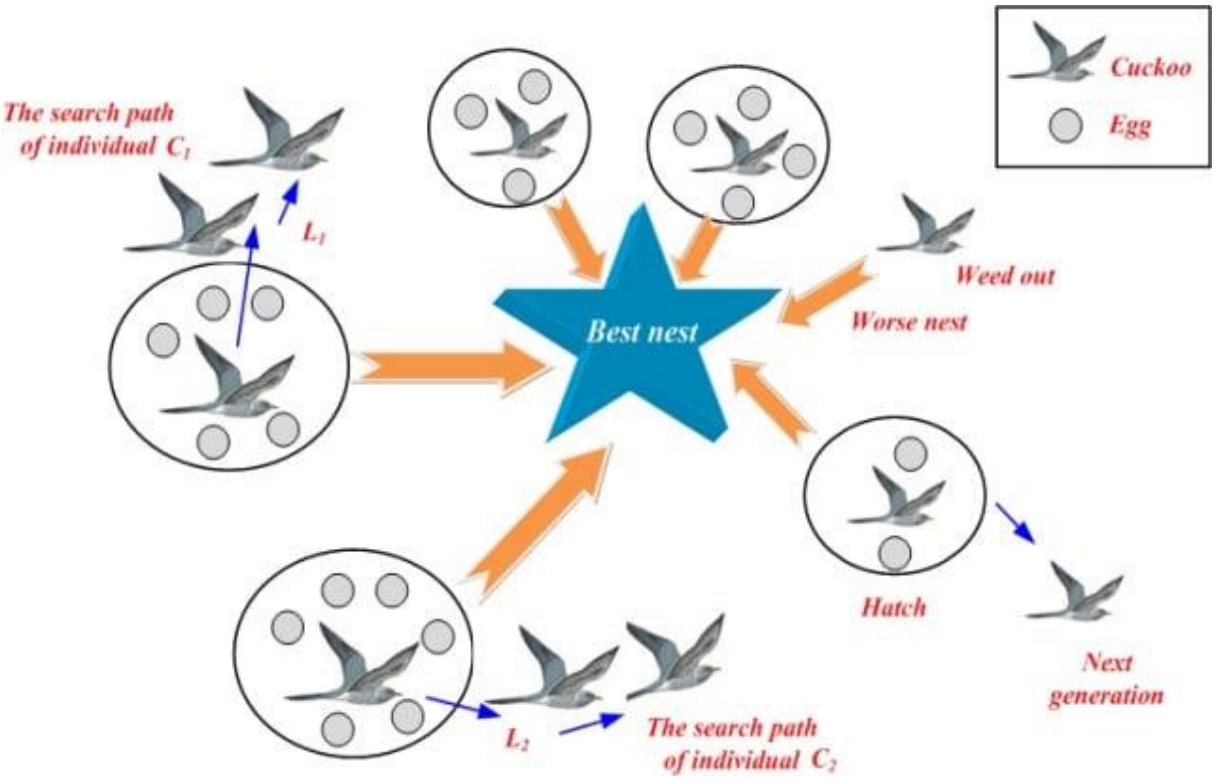
I. GIỚI THIỆU

Thuật toán tìm kiếm Cuckoo đã được sử dụng để tối ưu hóa các giải pháp trong điện toán đám mây, khai thác dữ liệu, thử nghiệm phần mềm, IoT và nhận dạng mẫu. Vì chỉ có một tham số, Pa, nên thuật toán này rất dễ triển khai.

Yang và Deb đã tạo ra thuật toán tìm kiếm chim cu vào năm 2009. Thuật toán này được lấy cảm hứng từ loài chim cu. Chim cu đẻ trứng trong tổ của những loài chim chủ khác. Động lực cơ bản đầu tiên để phát triển thuật toán tối ưu hóa mới là việc đẻ trứng và sinh sản của chim cu. Nếu chim chủ nhận ra những quả trứng không phải của mình thì nó sẽ vứt trứng ra khỏi tổ hoặc chỉ cần làm trống tổ và xây một tổ mới.

Mỗi quả trứng trong tổ đại diện cho một giải pháp. Trứng chim cu đại diện cho một lựa chọn mới và tốt. Câu trả lời thu được là một lựa chọn mới dựa trên lựa chọn hiện có với một số đặc điểm được sửa đổi. Ở dạng cơ bản nhất, mỗi tổ có một quả trứng chim cu, và mỗi tổ có nhiều trứng đại diện cho một tập hợp các lựa chọn. Tìm kiếm chim cu đã lý tưởng hóa hành vi sinh sản này. Thuật toán tìm kiếm chim cu có thể được áp dụng cho nhiều vấn đề tối ưu hóa. Thuật toán tối ưu hóa này cải thiện hiệu quả, độ chính xác và tỷ lệ hội tụ.

Để bắt đầu, mỗi con chim cu chỉ có thể đẻ một quả trứng tại một thời điểm. Sau đó, đẻ trứng vào một tổ được chọn ngẫu nhiên. Thứ hai, những tổ tốt nhất với những quả trứng chất lượng cao sẽ được truyền lại cho các thế hệ tương lai. Thứ ba, số lượng tổ vật chủ có sẵn được cố định:



II. ĐỘNG LỰC ĐỀ XUẤT THUẬT TOÁN CUCKOO SEARCH

Thuật toán Cuckoo Search ra đời với mục tiêu mô phỏng hành vi thông minh của loài chim cúc cu và tận dụng đặc điểm độc đáo của Levy Flight để khắc phục các hạn chế của các thuật toán tối ưu hóa truyền thống. Tính đơn giản, hiệu quả và khả năng giải quyết các bài toán phức tạp đã khiến CS trở thành một công cụ mạnh mẽ trong lĩnh vực tối ưu hóa. Động lực chính của việc đề xuất thuật toán này xuất phát từ các yếu tố sau:

1. Hành vi sinh sản độc đáo của loài chim cúc cu

- Chim cúc cu có chiến lược sinh sản rất đặc biệt:
  - Chúng **đặt trứng của mình vào tổ của các loài chim khác**.
  - Một số loài chim chủ nhận ra sự xâm nhập và phá hủy trứng cúc cu hoặc bỏ tổ.
  - Trứng cúc cu có xác suất sống sót cao hơn khi chúng giống với trứng của chim chủ.
- **Ý tưởng áp dụng**:
  - Hành vi sinh sản của cúc cu có thể được mô phỏng như quá trình tìm kiếm giải pháp tối ưu trong một không gian giải pháp phức tạp.

2. Khả năng tìm kiếm hiệu quả thông qua bước Levy Flight

- **Levy Flight** là một kiểu chuyển động ngẫu nhiên, trong đó các bước nhảy dài được xen kẽ với các bước ngắn. Điều này phản ánh các hành vi tìm kiếm tài nguyên tự nhiên (như thức ăn) của các loài động vật.
- **Ý tưởng áp dụng**:
  - Levy Flight giúp khám phá không gian tìm kiếm một cách hiệu quả, cân bằng giữa việc **khai thác (exploitation)** và **thăm dò (exploration)**.
  - Những bước nhảy dài tương ứng với khả năng tìm kiếm các khu vực mới trong không gian giải pháp.
  - Những bước ngắn giúp tập trung khai thác các khu vực tiềm năng.

3. Khắc phục hạn chế của các thuật toán tối ưu hóa truyền thống

- **Vấn đề trong các thuật toán khác**:
  - Các thuật toán như GA (Genetic Algorithm), PSO (Particle Swarm Optimization) thường dễ bị **rơi vào cực trị địa phương (local optima)**.
  - Các thuật toán tối ưu truyền thống yêu cầu nhiều thông tin về bài toán hoặc dựa trên các nguyên tắc toán học phức tạp.
- **Cuckoo Search**:
  - Có khả năng **thoát khỏi cực trị địa phương** nhờ các bước Levy Flight.

- Đơn giản và dễ triển khai nhưng vẫn mạnh mẽ trong việc tìm kiếm giải pháp.

4. Tính đơn giản và linh hoạt

- Thuật toán CS sử dụng ít tham số:
  - Tỷ lệ thay thế tổ (pap_apa) và phân phối Levy Flight.
- Điều này làm cho thuật toán:
  - Dễ áp dụng vào nhiều bài toán tối ưu khác nhau.
  - Linh hoạt và hiệu quả khi giải quyết các bài toán phi tuyến tính hoặc có nhiều ràng buộc.

5. Hiệu quả trong nhiều lĩnh vực

- Cuckoo Search đã chứng minh được tính hiệu quả trong nhiều ứng dụng thực tế, ví dụ:
  - Tối ưu hóa thiết kế kỹ thuật.
  - Xử lý tín hiệu và hình ảnh.
  - Quản lý năng lượng và tài nguyên.
  - Điều khiển hệ thống và robot.

III. MỤC ĐÍCH

Thuật toán Cuckoo Search (CS) được phát triển nhằm giải quyết các bài toán tối ưu hóa phức tạp trong các lĩnh vực khoa học, kỹ thuật và công nghiệp. Mục tiêu cụ thể của thuật toán bao gồm:

1. Tìm kiếm giải pháp tối ưu (Optimization)

Thuật toán Cuckoo Search (CS) được thiết kế để tìm giải pháp tốt nhất hoặc gần tốt nhất trong không gian tìm kiếm của một bài toán tối ưu hóa. Điểm nổi bật của CS nằm ở việc **xác định và cải thiện giá trị hàm mục tiêu** thông qua mô phỏng hành vi tìm tổ của loài chim cu cu. Các bước nhảy **Levy Flight** trong CS cho phép quá trình tìm kiếm đi xa hơn các thuật toán truyền thống, từ đó đạt được các giải pháp chất lượng cao hơn.

Ứng dụng trong thực tế:

- **Tối ưu hóa công suất mạng điện:** CS được sử dụng để tính toán cách phân phối công suất hiệu quả trong lưới điện, giảm tổn thất điện năng.
- **Quản lý tài nguyên nước:** Dùng để tối ưu hóa dòng chảy trong các hệ thống thủy lợi hoặc hồ chứa.
- **Kỹ thuật hàng không:** Trong việc tối ưu hóa thiết kế khí động học để giảm lực cản và tiêu hao nhiên liệu.

2. Khắc phục cực trị địa phương (Avoiding Local Optima)

Trong các bài toán tối ưu hóa phi tuyến tính hoặc có nhiều cực trị địa phương, các thuật toán truyền thống như Gradient Descent thường dễ bị mắc kẹt trong các cực trị địa phương thay vì đạt cực trị toàn cục. Cuckoo Search giải quyết vấn đề này bằng cách sử dụng **Levy Flight**, một phương pháp mô phỏng chuyển động ngẫu nhiên với bước nhảy lớn.

- **Levy Flight hoạt động như thế nào?**

  Trong CS, Levy Flight giúp các cá thể (giải pháp) "nhảy" xa hơn khỏi khu vực hiện tại để tìm kiếm giải pháp tốt hơn ở các vùng không gian khác. Điều này làm tăng khả năng đạt được cực trị toàn cục.
- **Ưu điểm của Levy Flight:**
  - Dễ dàng thoát khỏi vùng cực trị địa phương.
  - Cải thiện hiệu quả tìm kiếm không gian rộng lớn.

Ứng dụng trong thực tế:

- **Tối ưu hóa dữ liệu y sinh học:** CS được dùng để phát hiện các mẫu (patterns) trong bộ dữ liệu phức tạp của y học.
- **Tìm kiếm trong không gian lớn:** CS được sử dụng để tối ưu hóa các thông số cho hệ thống robot trong môi trường có nhiều trở ngại.

3. Cân bằng giữa khai thác và thăm dò (Exploitation vs. Exploration)

CS cung cấp sự cân bằng hiệu quả giữa việc khai thác các khu vực tiềm năng (exploitation) và khám phá các khu vực mới (exploration):

- **Khai thác (Exploitation):** Tìm kiếm cục bộ xung quanh các giải pháp tiềm năng để cải thiện chúng.
- **Thăm dò (Exploration):** Di chuyển đến các khu vực chưa được khám phá để tìm các giải pháp mới.

Điểm nổi bật của CS là khả năng điều chỉnh linh hoạt giữa hai yếu tố này dựa trên xác suất **pa (probability of abandonment)**. Giá trị **pa** đại diện cho khả năng từ bỏ một giải pháp kém hiệu quả, từ đó tăng cơ hội khám phá các giải pháp tốt hơn.

Lợi ích:

- Giảm thiểu nguy cơ lặp lại các giải pháp không hiệu quả.
- Tăng tốc độ hội tụ đến giải pháp tối ưu.

Ứng dụng:

- **Tối ưu hóa đường đi:** Trong các bài toán logistic (tìm đường đi ngắn nhất cho xe vận tải).

- **Tối ưu hóa phần mềm:** CS được dùng để tìm thông số tốt nhất cho các phần mềm tự động hóa.

4. Giải quyết bài toán phi tuyến tính và không liên tục

Một số bài toán thực tế có các hàm mục tiêu phức tạp, không liên tục hoặc không thể tính toán đạo hàm. CS được thiết kế để làm việc hiệu quả với các bài toán này bằng cách:

- **Không yêu cầu tính đạo hàm:** Khác với Gradient Descent, CS chỉ cần giá trị hàm mục tiêu, không yêu cầu tính đạo hàm hoặc gradient.
- **Xử lý không gian tìm kiếm phức tạp:** CS có thể làm việc với các hàm mục tiêu phi tuyến tính, đa chiều, hoặc có nhiều ràng buộc.

Ứng dụng trong thực tế:

- **Tối ưu hóa thiết kế cơ khí:** CS được dùng để tìm ra các cấu trúc cơ khí tối ưu mà không cần phải giải các hệ phương trình đạo hàm.
- **Phân tích tài chính:** Trong việc dự báo giá cổ phiếu hoặc tối ưu hóa danh mục đầu tư với các ràng buộc phi tuyến.

5. Đơn giản hóa và khả năng mở rộng (Simplicity and Scalability)

CS được thiết kế với một cấu trúc đơn giản, sử dụng ít tham số và dễ triển khai. Điều này làm cho thuật toán dễ dàng được mở rộng hoặc cải tiến để giải quyết các bài toán lớn hơn, phức tạp hơn.

Lợi ích chính:

- **Dễ dàng cài đặt:** Các nhà nghiên cứu và kỹ sư có thể nhanh chóng triển khai CS trong các dự án thực tế.
- **Mở rộng linh hoạt:** CS có thể kết hợp với các thuật toán khác (như Genetic Algorithm hoặc Particle Swarm Optimization) để tạo ra các thuật toán lai (hybrid algorithms).

Ứng dụng:

- **Hệ thống thông minh:** CS được sử dụng trong hệ thống IoT và các ứng dụng tự động hóa để tối ưu hóa hiệu suất.
- **Big Data:** CS được dùng để tối ưu hóa việc xử lý dữ liệu lớn và phân tích thông tin.

III. CÁC BƯỚC CỦA THUẬT TOÁN CUCKOO SEARCH (CS)

B1. Khởi tạo

Chim cu gáy thích đẻ trứng vào tổ của các loài chim khác.

B2. Chuyến bay Levy

Đây là một chuyến bay hoặc đi bộ ngẫu nhiên. Các bước được xác định theo chiều dài bước có phân phối xác suất nhất định với các hướng ngẫu nhiên. Kiểu bay này được quan sát thấy ở các loài động vật và côn trùng khác nhau. Chuyển động tiếp theo được xác định bởi vị trí hiện tại.

B3. Tính toán thể lực

Tính toán độ thích nghi đạt được bằng cách sử dụng hàm độ thích nghi để tìm ra giải pháp tốt nhất. Tổ được chọn ngẫu nhiên. Độ thích nghi của trứng chim cu gáy (giải pháp mới) sau đó được so sánh với độ thích nghi của trứng chủ (giải pháp) trong tổ. Nếu giá trị hàm độ thích nghi của trứng chim cu gáy nhỏ hơn hoặc bằng giá trị hàm độ thích nghi của tổ được chọn ngẫu nhiên, tổ được chọn ngẫu nhiên sẽ được thay thế bằng giải pháp mới.

B4. Chấm dứt

Hàm thể lực so sánh các giải pháp trong lần lặp hiện tại và chỉ có giải pháp tốt nhất được truyền tiếp. Nếu số lần lặp ít hơn số lần lặp tối đa, tổ chim tốt nhất sẽ được giữ lại. Tất cả các loài chim cu gáy đều sẵn sàng cho các hành động tiếp theo của chúng sau khi hoàn tất các quá trình khởi tạo, bay lên và tính toán thể lực. Thuật toán tìm kiếm chim cu gáy sẽ bị chấm dứt khi đạt đến số lần lặp tối đa. Các bước này có thể áp dụng cho bất kỳ bài toán tối ưu hóa nào. Trong những trường hợp như vậy, mỗi quả trứng chim cu gáy và tổ chim cu gáy đều đóng vai trò quan trọng.

Tối ưu hóa chi phí và lỗ hổng trên đám mây bằng thuật toán tìm kiếm Cuckoo với các chuyến bay Levy: Khi giảm thiểu chi phí và lỗ hổng, một số rủi ro và mối đe dọa được đưa vào. Do đó, bộ kỹ thuật được sử dụng để giảm số lượng lỗ hổng và mối đe dọa bảo mật trên các hệ thống thông tin dựa trên đám mây. Để giảm nguy cơ tấn công proxy, khoảng cách giữa nút dễ bị tấn công và nút nạn nhân tiềm năng được tăng lên.

Thuật toán tìm kiếm Cuckoo cho thiết kế mảng ăng-ten: Mảng ăng-ten hình nón hữu ích trong liên lạc vệ tinh, liên lạc tàu ngầm và liên lạc điểm-đến-điểm. Thuật toán tìm kiếm Cuckoo có thể được sử dụng để giảm các tham số hoặc thành phần của mảng ăng-ten như dòng điện tương đối, pha của các thành phần và các thành phần xen kẽ. Mục tiêu chính là ngăn chặn các thùy bên và điều khiển null theo các hướng cụ thể của mẫu bức xạ.

Lựa chọn tính năng sử dụng thuật toán tìm kiếm Cuckoo: Các phần tử dữ liệu được gọi là các tính năng, biến hoặc thuộc tính. Lựa chọn tính năng là một phương pháp xử lý trước dữ liệu được sử dụng trong phân loại. Nó được sử dụng để loại bỏ các thuộc tính không cần thiết và trùng lặp khỏi các tập dữ liệu được cung cấp. Độ chính xác được tăng lên và thời gian đào tạo được giảm đáng kể. Mục tiêu chính là giảm thiểu số lượng đặc điểm trong khi cải thiện hiệu suất phân loại. Vectơ nhị phân được sử dụng để giải quyết vấn đề chọn hoặc không chọn tính năng nào trong một vấn đề cụ thể. Trong trường hợp này, 0 chỉ ra không chọn và 1 chỉ ra chọn tính năng có liên quan.

IV. THUẬT TOÁN CUCKOO SEARCH

Bài toán: Giả sử tìm Min của hàm số trong khoảng sau:

$$F(x) = x.cos(x) - (0.5xcos(0.6x\text{^}2)) \ biết \ x \ thuộc \ [5; 13]$$

1. **Phân tích bài toán**

· **Hàm mục tiêu**: $F(x) = xcos(x) - (0.5xcos(0.6x\text{^}2))$.

· **Phạm vi tìm kiếm**: x∈[5,13]
· **Mục tiêu**: Tìm x sao cho F(x) đạt giá trị nhỏ nhất.

2. Thuật toán

Cuckoo Search là một thuật toán metaheuristic được lấy cảm hứng từ hành vi sinh sản của loài chim cúc cu, bao gồm:

- Khởi tạo tổ ban đầu: Xác định một số tổ ban đầu với vị trí ngẫu nhiên trong không gian tìm kiếm.

- Tính fitness của tổ: Tính giá trị hàm F(x) tại vị trí mỗi tổ

- Sinh tổ mới: Các tổ mới được tạo ra bằng cách sử dụng phân phối Gaussian, đại diện cho sự đột biến.

- Thay thế tổ xấu: Một số tổ có fitness kém bị thay thế bằng tổ mới sinh ra ngẫu nhiên.

- Cập nhật tổ tốt nhất: Sau mỗi vòng lặp, chọn tổ có giá trị fitness tốt nhất.

- Kiểm tra điều kiện dừng: Dừng khi đạt số vòng lặp tối đa hoặc không có cải thiện đáng kể.

3. Các bước thực hiện thuật toán

Bước 1: Khởi tạo
+Số lượng tổ: n(nests)= 25
+ Số vòng lặp tối đa: max_iter= 100
+Xác suất thay thế tổ: pa= 0.25
Khởi tạo ngẫu nhiên n(nests) tổ trong khoảng [5;13]
Bước 2: Đột biến tổ

Mỗi tổ sinh ra tổ mới dựa trên công thức:

new_nest=nest+noise

với noise~N(0,1) là phân phối Gaussian.
Bước 3: Thay thế tổ kém
Một số tổ kém fitness bị thay thế bằng tổ mới ngẫu nhiên trong khoảng [5,13]
Bước 4: Cập nhật tổ tốt nhất
So sánh giá trị F(x) của tất cả tổ để chọn tổ tốt nhất (giá trị nhỏ nhất).
Bước 5: Kiểm tra điều kiện dừng
Lặp lại các bước trên cho đến khi đạt số vòng lặp tối đa hoặc giá trị tốt nhất không cải thiện thêm.
-Code chạy:

```
clc,clear all;
S=10;
```

```
N=10;
Beta= 1.5;
iter=3000;
G= 100; XG =10;
pa= 0.25;
x=rand(1,10);
xmin= 5;
xmax=13;
for i=1:10
        p(i)=100; xB(i)= 10; pa(i) = 0.25;
end
for j=1:iter
for i=1:10
        u=randn (1)*2;
        v=randn(1);
        levy=u/abs(v)^(1/Beta);
        x(i)=x(i)+0.5*levy;
        if  x(i)>xmax x(i)=xmax;
                end;
            if x(i) < xmin x(i)=xmin;
                end;
p(i) = test(x(i));
if p(i) < pa (i) pa(i) =p(i); xB(i)=x(i);
        end
end
        for i=1:10
             if p(i)< G XG=x(i);
                end
        end
end
```

-Kết quả:

+ Giá trị x **tối ưu**: Điểm mà F(x) đạt giá trị nhỏ nhất trong [5,13]: x≈9.706

+ Giá trị F(x) **tối ưu**: Giá trị nhỏ nhất của hàm: F(x)≈−14.176

Đánh giá giải thuật

- **Ưu điểm**

Tính đơn giản: CS dễ dàng triển khai và hiểu, với các bước đơn giản gồm: sinh tổ mới, đánh giá fitness, và thay thế tổ kém.

Khả năng khám phá mạnh mẽ: Phân phối Levy giúp thuật toán tìm kiếm trên toàn bộ không gian giải pháp hiệu quả, giảm nguy cơ rơi vào cực tiểu cục bộ.

Hiệu suất tốt trên nhiều bài toán: CS hoạt động tốt trong các bài toán tối ưu hóa liên tục và tổ hợp, cạnh tranh với nhiều thuật toán khác như Particle Swarm Optimization (PSO) hoặc Genetic Algorithm (GA).

Cân bằng giữa khai thác và khám phá: CS sử dụng xác suất thay thế tổ **pa** để điều chỉnh giữa việc tìm kiếm xung quanh các giải pháp tiềm năng và nhảy đến các vùng không gian mới.

Tương thích với nhiều dạng bài toán: Có thể áp dụng CS cho các bài toán đa ngành như tối ưu hóa tham số trong kỹ thuật, học máy, tài chính, và xử lý ảnh.

● **Nhược điểm**

Nhạy cảm với tham số: Hiệu suất của CS phụ thuộc nhiều vào các tham số như pa (xác suất thay thế tổ) và β\betaβ (tham số Levy). Việc tinh chỉnh các tham số này không dễ dàng.

Chậm hội tụ trong một số trường hợp: Do tập trung vào khám phá, CS có thể mất nhiều thời gian để đạt hội tụ, đặc biệt trong các bài toán yêu cầu độ chính xác cao.

Không phù hợp cho bài toán phức tạp cao: Với không gian tìm kiếm có độ phức tạp cao (nhiều chiều hoặc ràng buộc phi tuyến), CS có thể gặp khó khăn so với các thuật toán tối ưu khác.

Thiếu sự tương thích ràng buộc: CS không được thiết kế chuyên biệt để xử lý các bài toán ràng buộc, cần thêm cơ chế điều chỉnh ràng buộc.

● **Ứng dụng thực tế**

Kỹ thuật và công nghiệp: Tối ưu hóa tham số hệ thống, thiết kế mạch điện tử, và tối ưu hóa mạng cảm biến.

Xử lý tín hiệu và ảnh: Tối ưu hóa bộ lọc tín hiệu, phân đoạn ảnh, và trích xuất đặc trưng.

Học máy: Tối ưu hóa hyperparameter trong các mô hình như Neural Network, SVM.

Kinh tế và tài chính: Lập kế hoạch sản xuất, tối ưu hóa danh mục đầu tư.

Sinh học và y học: Tối ưu hóa quá trình điều trị, phân tích gen.

V. KẾT LUẬN

Thuật toán Cuckoo Search (CS) là một thuật toán metaheuristic hiện đại, lấy cảm hứng từ hành vi sinh học của chim cu cu kết hợp với cơ chế Levy Flight để thực hiện tìm kiếm tối ưu hóa. CS ra đời nhằm giải quyết các bài toán tối ưu hóa phức tạp và đã nhanh chóng khẳng định được hiệu quả vượt trội trong nhiều lĩnh vực.

Một trong những điểm mạnh của CS là sự kết hợp hài hòa giữa tìm kiếm toàn cục và khai thác cục bộ. Điều này giúp thuật toán tránh được hiện tượng rơi vào cực trị địa phương, đồng thời khai thác triệt để không gian tìm kiếm để tìm ra lời giải tối ưu. Bên cạnh đó, CS có cấu trúc đơn giản, dễ triển khai và không đòi hỏi nhiều tham số phức tạp, khiến nó trở thành lựa chọn hấp dẫn trong các ứng dụng thực tiễn.

Tuy nhiên, thuật toán cũng gặp phải một số thách thức, như hiệu suất có thể giảm trong các bài toán tối ưu hóa có không gian tìm kiếm lớn hoặc các vấn đề đòi hỏi độ chính xác cao. Ngoài ra, hiệu quả của CS phụ thuộc nhiều vào cách thiết lập các tham số như kích thước quần thể, xác suất thay thế tổ trứng, và bước nhảy Levy Flight. Những yếu tố này đòi hỏi sự nghiên cứu và điều chỉnh kỹ lưỡng để đạt hiệu suất tối ưu.

Trong ứng dụng, CS đã được áp dụng thành công trong nhiều lĩnh vực như tối ưu hóa kỹ thuật, mạng lưới cảm biến, xử lý tín hiệu, học máy, và nhiều ngành công nghiệp

khác. Với khả năng mở rộng và cải tiến linh hoạt, CS cũng tạo nền tảng để phát triển các phiên bản lai ghép, kết hợp với các thuật toán khác nhằm cải thiện hiệu suất.

Nhìn chung, CS không chỉ là một thuật toán mạnh mẽ trong tối ưu hóa mà còn là nguồn cảm hứng cho các nghiên cứu sâu hơn về thuật toán tiến hóa và tìm kiếm metaheuristic. Với sự phát triển không ngừng của khoa học và công nghệ, CS được dự đoán sẽ tiếp tục khẳng định vai trò của mình trong việc giải quyết các bài toán thực tiễn và nghiên cứu trong tương lai.