**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC ĐẠI NAM**

Logo, company name

Description automatically generated

**BÁO CÁO HỌC PHẦN**

**TÊN HỌC PHẦN: TRÍ TUỆ NHÂN TẠO**

**ĐỀ TÀI: TÌM HIỂU VỀ THUẬT TOÁN CUCKOO SEARCH.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| STT | Mã Sinh Viên | Họ và Tên | Ngày Sinh | Lớp |
| 1 | 1771020412 | Trần Văn Lâm | 10/07/2005 | 17-01 |
| 2 | 1771020451 | Đoàn Duy Mạnh | 11/07/2005 | 17-01 |
| 3 |  |  |  | 17-01 |

### 

**Hà Nội, năm 2025**

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC ĐẠI NAM**

Logo, company name

Description automatically generated

**BÁO CÁO HỌC PHẦN**

**TÊN HỌC PHẦN: TRÍ TUỆ NHÂN TẠO**

**ĐỀ TÀI: TÌM HIỂU VỀ THUẬT TOÁN CUCKOO SEARCH.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| STT | Mã Sinh Viên | Họ và Tên | Ngày Sinh | Điểm | |
| Bằng Số | Bằng Chữ |
| 1 | 1771020412 | Trần Văn Lâm | 10/07/2005 |  |  |
| 2 | 1771020451 | Đoàn Duy Mạnh | 11/07/2005 |  |  |
| 3 |  |  |  |  |  |

### 

### CÁN BỘ CHẤM THI 1 CÁN BỘ CHẤM THI 2

**Hà Nội, năm 2025**

**LỜI NÓI ĐẦU**

Trong bối cảnh khoa học công nghệ phát triển mạnh mẽ, Trí tuệ nhân tạo (AI) đã và đang trở thành một lĩnh vực then chốt, mang lại những bước tiến vượt bậc trong nhiều ngành nghề. Từ những hệ thống nhận diện giọng nói, xử lý ngôn ngữ tự nhiên, đến các ứng dụng tối ưu hóa phức tạp, AI ngày càng khẳng định vai trò không thể thiếu trong cuộc sống hiện đại.

Môn học Trí tuệ nhân tạo không chỉ cung cấp những kiến thức nền tảng về các thuật toán và mô hình AI, mà còn mở ra cơ hội để chúng ta khám phá và ứng dụng những công nghệ tiên tiến này vào giải quyết các vấn đề thực tiễn. Trong đó, các thuật toán tối ưu hóa, như thuật toán Cuckoo Search (CS), đóng vai trò quan trọng trong việc tìm kiếm giải pháp tối ưu cho các bài toán phức tạp.

Đề tài "Ứng dụng Thuật toán Cuckoo Search vào Bài toán Tối ưu hóa" được thực hiện nhằm mục đích nghiên cứu và triển khai thuật toán CS để giải quyết ba bài toán tối ưu hóa thực tế: tối ưu hóa tham số cho máy học (SVM), tối ưu hóa trọng số mạng nơ-ron và giải bài toán định tuyến (TSP). Qua đề tài này, chúng tôi mong muốn không chỉ củng cố kiến thức về thuật toán CS, mà còn góp phần vào việc ứng dụng AI vào giải quyết các vấn đề thực tiễn.

Chúng tôi xin gửi lời cảm ơn chân thành đến giảng viên đã tận tình hướng dẫn và hỗ trợ chúng tôi trong quá trình thực hiện đề tài. Chúng tôi cũng xin cảm ơn các bạn sinh viên đã cùng nhau trao đổi, chia sẻ kiến thức và kinh nghiệm, giúp chúng tôi hoàn thành đề tài này.

**MỤC LỤC**

[LỜI NÓI ĐẦU 3](#_Toc193450764)

[MỤC LỤC 4](#_Toc193450765)

[MỤC LỤC HÌNH ẢNH 6](#_Toc193450766)

[CHƯƠNG 1. GIỚI THIỆU VỀ THUẬT TOÁN CUCKOO SEARCH 7](#_Toc193450767)

[1.1 Giới thiệu về thuật toán Cuckoo Search 7](#_Toc193450768)

[1.2 Mô tả chi tiết thuật toán Cuckoo Search 7](#_Toc193450769)

[*1.2.1 Cơ chế hoạt động của thuật toán* 7](#_Toc193450770)

[*1.2.2 Các bước thực hiện thuật toán* 8](#_Toc193450771)

[*1.2.3 Bước bay Lévy trong Cuckoo Search* 8](#_Toc193450772)

[*1.2.4 Lợi ích và ứng dụng của thuật toán Cuckoo Search* 8](#_Toc193450773)

[*1.2.5 So sánh với các thuật toán tối ưu hóa khác* 9](#_Toc193450774)

[*1.2.6 Nhược điểm của thuật toán Cuckoo Search* 9](#_Toc193450775)

[1.3 Kết luận 9](#_Toc193450776)

[CHƯƠNG 2. NGUYÊN LÝ LÀM VIỆC CỦA THUẬT TOÁN 11](#_Toc193450777)

[2.1. Quá trình khởi tạo và biểu diễn cá thể 11](#_Toc193450778)

[2.2. Lévy Flight – Công cụ tìm kiếm ngẫu nhiên có định hướng 12](#_Toc193450779)

[2.3. Lựa chọn chim mồi và quy tắc thay thế tổ 13](#_Toc193450780)

[2.4. Cơ chế khám phá và khai thác song song trong thuật toán 14](#_Toc193450781)

[2.5. Các thuật toán phụ trợ và điều chỉnh tham số 14](#_Toc193450782)

[2.6. Mô hình hóa toán học trong quá trình tìm kiếm 15](#_Toc193450783)

[2.7. Vai trò của yếu tố ngẫu nhiên và định hướng trong quá trình tối ưu 15](#_Toc193450784)

[2.8. Sự cạnh tranh và đồng đẳng trong quần thể 16](#_Toc193450785)

[2.9. Ứng dụng của quy tắc thay thế tổ trong các bài toán thực tiễn 16](#_Toc193450786)

[2.10. Kết luận 16](#_Toc193450787)

[CHƯƠNG 3. ỨNG DỤNG THUẬT TOÁN CUCKOO SEARCH TRONG CÁC BÀI TOÁN TỐI ƯU HÓA 18](#_Toc193450788)

[3.1. Giới thiệu 18](#_Toc193450789)

[3.2. Lý thuyết nền tảng của Thuật toán Cuckoo Search (CS) 18](#_Toc193450790)

[3.3. Triển khai và Ứng dụng CS trong các bài toán tối ưu hóa 18](#_Toc193450791)

[*3.3.1. Tối ưu hóa Tham số SVM* 18](#_Toc193450792)

[*3.3.2. Tối ưu hóa Trọng số Mạng Nơ-ron* 20](#_Toc193450793)

[*3.3.3. Giải Bài toán Định tuyến (TSP)* 21](#_Toc193450794)

[3.4. Đánh giá và Thảo luận 25](#_Toc193450795)

[3.5. Link Github: 25](#_Toc193450796)

[KẾT LUẬN 26](#_Toc193450797)

[DANH MỤC TÀI LIỆU THAM KHẢO 27](#_Toc193450798)

**MỤC LỤC HÌNH ẢNH**

[*Hình 1- Bảng so sánh các thuật toán tối ưu 9*](#_Toc193450536)

[*Hình 2- Quá trình thuật toán tìm kiếm Cuckoo 11*](#_Toc193450537)

[*Hình 3- Quá trình Lévy Flight 13*](#_Toc193450538)

[*Hình 4- Đồ thị hội tụ Cuckoo Search - SVM. 19*](#_Toc193450539)

[*Hình 5- Kết quả tối ưu SVM. 20*](#_Toc193450540)

[*Hình 6- Đồ thị hội tụ Cuckoo Search - ANN. 21*](#_Toc193450541)

[*Hình 7- Kết quả tối ưu hóa trọng số mạng nơ-ron, hiển thị loss tối ưu. 21*](#_Toc193450542)

[*Hình 8- Đồ thị hội tụ Cuckoo Search - TSP. 23*](#_Toc193450543)

[*Hình 9- Kết quả tối ưu hóa lộ trình TSP, hiển thị lộ trình và tổng khoảng cách. 24*](#_Toc193450544)

[*Hình 10- Đồ thị TSP tối ưu. 24*](#_Toc193450545)

**CHƯƠNG 1. GIỚI THIỆU VỀ THUẬT TOÁN CUCKOO SEARCH**

**1.1 Giới thiệu về thuật toán Cuckoo Search**

Thuật toán Cuckoo Search (CS) là một thuật toán tối ưu hóa metaheuristic được phát triển bởi Xin-She Yang và Suash Deb vào năm 2009, lấy cảm hứng từ hành vi sinh sản đặc biệt của loài chim cúc cu (Cuckoo). Loài chim này có tập tính đẻ trứng vào tổ của các loài chim khác để tận dụng nguồn lực nuôi dưỡng. Một số loài chim chủ có khả năng phát hiện ra trứng của chim cúc cu và sẽ loại bỏ chúng hoặc bỏ đi tổ. Dựa trên đặc điểm này, thuật toán Cuckoo Search được xây dựng để tìm kiếm các lời giải tốt hơn trong không gian tối ưu.

Thuật toán này đã được chứng minh là có hiệu suất cao trong việc giải quyết các bài toán tối ưu hóa phi tuyến, bài toán có ràng buộc và nhiều bài toán thực tế khác trong các lĩnh vực như kỹ thuật, tài chính, khoa học dữ liệu, và trí tuệ nhân tạo.

Cuckoo Search có thể hoạt động tốt trên các bài toán có không gian tìm kiếm lớn và phức tạp nhờ vào cơ chế cập nhật lời giải thông qua bước bay Lévy, giúp cải thiện khả năng hội tụ và mở rộng phạm vi tìm kiếm.

**1.2 Mô tả chi tiết thuật toán Cuckoo Search**

***1.2.1 Cơ chế hoạt động của thuật toán***

Thuật toán Cuckoo Search dựa trên ba quy tắc chính:

* Mỗi con chim cúc cu đẻ một trứng (tạo ra một giải pháp mới) vào một tổ được chọn ngẫu nhiên.
* Các tổ có trứng chất lượng cao hơn (tức là giải pháp tốt hơn) sẽ có nhiều cơ hội tồn tại và được truyền sang thế hệ sau.
* Số lượng tổ có sẵn là cố định, và một số tổ có thể bị phát hiện bởi chim chủ. Khi đó, trứng trong tổ có thể bị loại bỏ hoặc tổ có thể bị thay thế bằng một tổ mới.

Cuckoo Search sử dụng một kỹ thuật gọi là bước bay Lévy (Lévy Flight) để cập nhật vị trí của các cá thể, giúp thuật toán tránh bị kẹt trong cực trị cục bộ và có thể khám phá không gian tìm kiếm hiệu quả hơn.

***1.2.2 Các bước thực hiện thuật toán***

1. Khởi tạo quần thể tổ: Một số tổ ban đầu được tạo ngẫu nhiên trong không gian tìm kiếm.
2. Sinh trứng mới bằng bước bay Lévy: Một số tổ sẽ được cập nhật bằng cách sinh ra trứng mới và sử dụng bước bay Lévy để tìm kiếm các nghiệm mới.
3. Đánh giá chất lượng: Các nghiệm mới được so sánh với các nghiệm hiện có, và những tổ có trứng kém chất lượng hơn có thể bị thay thế.
4. Xác suất phát hiện: Với một xác suất nhất định, một số tổ sẽ bị thay thế bởi tổ mới được tạo ngẫu nhiên để đảm bảo sự đa dạng trong không gian tìm kiếm.
5. Lặp lại quá trình: Quá trình này tiếp tục cho đến khi đạt điều kiện dừng (ví dụ: số vòng lặp tối đa hoặc độ hội tụ đạt yêu cầu).

***1.2.3 Bước bay Lévy trong Cuckoo Search***

Bước bay Lévy là một phương pháp quan trọng trong Cuckoo Search giúp các cá thể trong thuật toán có khả năng di chuyển xa, tránh bị mắc kẹt trong cực trị cục bộ. Công thức của bước bay Lévy có thể được biểu diễn như sau:

Trong đó:

* là vị trí hiện tại của cá thể.
* là hệ số điều chỉnh bước bay.
* là một bước bay Lévy được tạo theo phân phối Lévy.

Bước bay Lévy có đặc điểm là tạo ra các bước nhảy lớn, giúp thuật toán có khả năng tìm kiếm mạnh mẽ trong không gian lớn, tránh bị mắc kẹt tại các cực trị cục bộ.

***1.2.4 Lợi ích và ứng dụng của thuật toán Cuckoo Search***

Thuật toán Cuckoo Search có một số ưu điểm quan trọng:

* Khả năng tìm kiếm toàn cục mạnh mẽ: Nhờ bước bay Lévy, thuật toán có khả năng tìm ra giải pháp tối ưu mà không bị kẹt trong cực trị cục bộ.
* Ít tham số cần điều chỉnh: So với các thuật toán tối ưu hóa khác như Genetic Algorithm (GA) hoặc Particle Swarm Optimization (PSO), Cuckoo Search chỉ có một vài tham số quan trọng, giúp giảm thiểu công sức điều chỉnh.
* Áp dụng hiệu quả trong nhiều lĩnh vực: Thuật toán này được sử dụng trong tối ưu hóa thiết kế kỹ thuật, tối ưu hóa tài chính, phân loại dữ liệu, đào tạo mạng nơ-ron nhân tạo và nhiều bài toán thực tế khác.

***1.2.5 So sánh với các thuật toán tối ưu hóa khác***

|  |  |  |  |
| --- | --- | --- | --- |
| **Đặc điểm** | **Cuckoo Search** | **Genetic Algorithm (GA)** | **Particle Swarm Optimization (PSO)** |
| Cơ chế tìm kiếm | Bước bay Lévy | Chọn lọc, lai ghép, đột biến | Dựa trên quỹ đạo của cá thể |
| Khả năng tránh cực trị cục bộ | Cao | Trung bình | Trung bình |
| Số tham số cần điều chỉnh | Ít | Nhiều | Trung bình |
| Khả năng hội tụ | Nhanh | Chậm hơn | Nhanh |

Hình 1- Bảng so sánh các thuật toán tối ưu

***1.2.6 Nhược điểm của thuật toán Cuckoo Search***

Mặc dù có nhiều ưu điểm, Cuckoo Search cũng có một số hạn chế:

* Nếu không điều chỉnh đúng, bước bay Lévy có thể khiến thuật toán di chuyển quá xa và bỏ lỡ các vùng tiềm năng.
* Thuật toán có thể cần số vòng lặp lớn để hội tụ đối với các bài toán có không gian tìm kiếm phức tạp.
* Việc tìm kiếm giải pháp mới đôi khi có thể bị ảnh hưởng bởi lựa chọn ngẫu nhiên của tổ chim, làm giảm hiệu suất trên một số bài toán nhất định.

**1.3 Kết luận**

Thuật toán Cuckoo Search là một phương pháp tối ưu hóa mạnh mẽ, lấy cảm hứng từ tự nhiên, có khả năng tìm kiếm hiệu quả và ít bị mắc kẹt trong cực trị cục bộ. Với các quy tắc đơn giản nhưng hiệu quả, thuật toán này đã chứng tỏ được tính ứng dụng cao trong nhiều lĩnh vực khác nhau. Trong chương tiếp theo, chúng ta sẽ áp dụng thuật toán này vào một bài toán tối ưu cụ thể để hiểu rõ hơn về cách nó hoạt động trong thực tế.

**CHƯƠNG 2. NGUYÊN LÝ LÀM VIỆC CỦA THUẬT TOÁN**

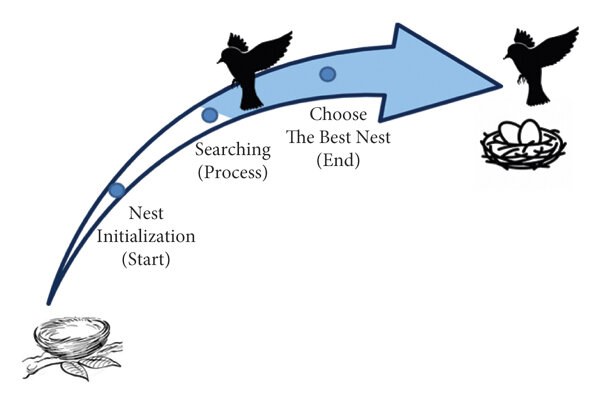
Thuật toán Cuckoo Search (CS) hoạt động dựa trên một số nguyên lý sinh học mô phỏng hành vi sinh sản và tìm kiếm của loài chim Cuckoo, với mục tiêu khai thác và khám phá không gian giải pháp một cách hiệu quả. Phần này sẽ trình bày chi tiết các bước chính trong quá trình tìm kiếm, cách thức lựa chọn chim mồi, cũng như cơ chế cạnh tranh giữa các tổ – những thành phần cốt lõi tạo nên hiệu quả của thuật toán.

## 2.1. Quá trình khởi tạo và biểu diễn cá thể

Mỗi cá thể trong thuật toán Cuckoo Search thể hiện một giải pháp khả thi, được biểu diễn dưới dạng một vector các giá trị của các biến quyết định. Quá trình khởi tạo bắt đầu với một tập hợp các cá thể (hay còn gọi là quần thể), được sinh ra ngẫu nhiên trong không gian giải pháp. Mỗi cá thể ban đầu sẽ trải qua các bước tìm kiếm, từ đó dần dần cải thiện giá trị hàm mục tiêu. Quá trình này được thực hiện theo các bước sau:

Khởi tạo ngẫu nhiên: Một quần thể có kích thước N được tạo ra với các cá thể có giá trị ngẫu nhiên nằm trong biên giới cho phép của bài toán.

Đánh giá ban đầu: Mỗi cá thể được đánh giá thông qua hàm mục tiêu để xác định độ phù hợp (fitness) của các giải pháp. Cơ chế đánh giá giúp xác định những cá thể có khả năng tiến hóa tốt hơn.



Hình 2- Quá trình thuật toán tìm kiếm Cuckoo

*Nguồn :* [*researchgate.net*](https://www.researchgate.net/figure/Cuckoo-search-algorithm-process_fig6_361218076)

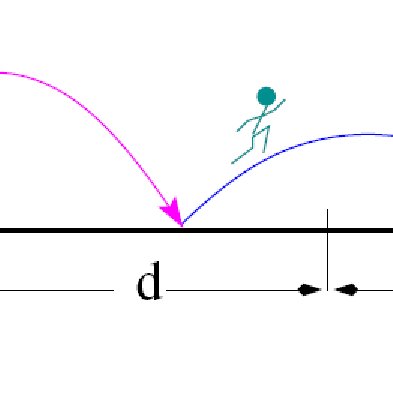
## 2.2. Lévy Flight – Công cụ tìm kiếm ngẫu nhiên có định hướng

Một trong những điểm đặc sắc của thuật toán CS là cơ chế di chuyển của các cá thể thông qua một quá trình ngẫu nhiên có định hướng gọi là Lévy Flight. Lévy Flight được mô phỏng dựa trên một chuỗi các bước ngẫu nhiên mà khoảng cách di chuyển có thể rất lớn, giúp cá thể chuyển vị từ vị trí này sang vị trí khác trong không gian giải pháp một cách không tuyến tính. Điều này giúp CS có khả năng:

* Khám phá không gian giải pháp rộng: Trong những trường hợp bài toán có nhiều cực trị cục bộ, Lévy Flight cho phép nhảy ra khỏi vùng tối ưu cục bộ để tìm kiếm vùng mới có khả năng chứa cực trị toàn cục.
* Cân bằng giữa khai thác và khám phá: Sự kết hợp giữa bước di chuyển lớn và bước di chuyển ngẫu nhiên với kích thước nhỏ giúp thuật toán vừa khai thác các giải pháp tiềm năng hiện có vừa có khả năng khám phá những vùng mới, từ đó cải thiện kết quả tìm kiếm.

Các bước cụ thể trong quá trình Lévy Flight bao gồm:

* Xác định bước nhảy: Độ dài bước di chuyển được lựa chọn dựa trên phân phối Lévy, thường biểu diễn dưới dạng hàm mũ mũ số, có khả năng tạo ra bước nhảy ngẫu nhiên với khoảng cách lớn.
* Cập nhật vị trí của cá thể: Mỗi cá thể thực hiện bước nhảy từ vị trí cũ sang vị trí mới dựa trên công thức di chuyển liên quan đến Lévy Flight. Quá trình này được lặp đi lặp lại cho đến khi các điều kiện dừng được thỏa mãn (ví dụ như số lần lặp tối đa hoặc độ cải thiện hàm mục tiêu không còn đáng kể).



Hình 3- Quá trình Lévy Flight

*Nguồn :* [*researchgate.net*](https://www.researchgate.net/publication/228030356_Introduction_to_the_Theory_of_Levy_Flights)

## 2.3. Lựa chọn chim mồi và quy tắc thay thế tổ

Một điểm đặc trưng độc đáo của thuật toán Cuckoo Search là quá trình lựa chọn chim mồi và cơ chế thay thế tổ. Cụ thể, mỗi cá thể được coi là một “tổ” chứa một giải pháp. Khi một cá thể mới được tạo ra thông qua Lévy Flight, nó sẽ được so sánh với một tổ (hoặc cá thể) được chọn ngẫu nhiên từ quần thể hiện có. Quy trình này được thực hiện theo các bước sau:

* So sánh hàm mục tiêu: Nếu giải pháp mới (tổ mới) có giá trị hàm mục tiêu tốt hơn so với giải pháp hiện có trong tổ được chọn, thì giải pháp cũ sẽ bị thay thế. Ngược lại, giải pháp mới có thể bị loại bỏ.
* Áp dụng quy tắc chọn lọc: Quá trình so sánh nhằm mục tiêu loại bỏ các giải pháp kém hiệu quả. Trong trường hợp tổ cũ có chất lượng tốt hơn, nó sẽ được giữ lại, đảm bảo rằng chất lượng của quần thể không bị xuống cấp theo từng bước lặp.
* Sự cạnh tranh giữa các người “nuôi” tổ: Khi nhiều cá thể cùng cạnh tranh giành chỗ đứng trong quần thể, việc loại bỏ những cá thể có hiệu quả thấp giúp quần thể tiến hóa theo hướng tối ưu hơn. Cơ chế cạnh tranh này không chỉ đảm bảo sự tốt lên của hàm mục tiêu mà còn tăng tính đa dạng của các giải pháp hiện có, từ đó tránh bẫy cục bộ hiệu quả.

## 2.4. Cơ chế khám phá và khai thác song song trong thuật toán

Trong quá trình tối ưu hóa, sự cân bằng giữa việc khai thác các giải pháp tốt hiện có (exploitation) và khám phá các giải pháp mới (exploration) là yếu tố quyết định hiệu quả của thuật toán. Cuckoo Search có cơ chế đặc biệt để duy trì sự cân bằng này:

* Khai thác: Các cá thể với giá trị hàm mục tiêu cao được dùng làm cơ sở để tiếp tục tìm kiếm các giải pháp xung quanh. Việc này giúp cải thiện hơn nữa hàm mục tiêu của các vùng có định hướng tốt.
* Khám phá: Thông qua Lévy Flight và cơ chế lựa chọn tổ, thuật toán không chỉ tập trung vào khu vực có hiệu quả cao mà còn thường xuyên “nhảy” ra khỏi các vùng đã khai thác để tìm kiếm những khu vực mới. Điều này đặc biệt hữu ích khi không gian nghiệm chứa nhiều cực trị cục bộ.
* Cơ chế thay đổi ngẫu nhiên: Một đặc điểm khác của CS là một tỷ lệ phần trăm các tổ (giải pháp) sẽ bị thay thế hoàn toàn bằng các giải pháp ngẫu nhiên theo tỷ lệ p\_a (tỷ lệ phát hiện chim mồi). Điều này giúp tăng tính đa dạng của quần thể, đảm bảo rằng thuật toán không vướng vào bẫy địa phương lâu dài.

## 2.5. Các thuật toán phụ trợ và điều chỉnh tham số

Ngoài các bước chính đã đề cập, một số yếu tố phụ trợ cũng góp phần làm tăng hiệu quả của quá trình tìm kiếm:

* Điều chỉnh tham số Lévy Flight: Các tham số của phân phối Lévy, như chỉ số mũ và bước di chuyển trung bình, có thể được điều chỉnh để phù hợp với đặc tính của từng bài toán. Điều này cho phép thuật toán linh hoạt trong việc duy trì sự cân bằng giữa khám phá và khai thác.
* Tỷ lệ phát hiện và thay thế tổ (p\_a): Tỷ lệ p\_a đóng vai trò quan trọng trong việc đảm bảo rằng có một số lượng tổ nhất định luôn được thay thế bằng giải pháp ngẫu nhiên. Việc này giúp làm mới quần thể, ngăn ngừa hiện tượng hội tụ quá sớm vào cực trị cục bộ.
* Cập nhật đồng bộ và không đồng bộ: Một số phiên bản của thuật toán CS có thể cập nhật tất cả các cá thể đồng thời hoặc theo cách không đồng bộ (từng cá thể được cập nhật riêng lẻ). Lựa chọn cách thức cập nhật này dựa vào mục tiêu tối ưu hóa và đặc điểm của bài toán cụ thể.

## 2.6. Mô hình hóa toán học trong quá trình tìm kiếm

Để hiểu sâu hơn về nguyên lý hoạt động của thuật toán Cuckoo Search, chúng ta cần xem xét một số công thức toán học làm cơ sở cho các bước di chuyển và cập nhật giải pháp:

Công thức cập nhật vị trí: Một cá thể i với vị trí hiện tại x\_i(t) sẽ được cập nhật thành vị trí mới x\_i(t+1) theo công thức:  
 x\_i(t+1) = x\_i(t) + α ⊕ Lévy(λ)  
 Trong đó, α là hệ số bước ảnh hưởng đến độ dài của bước di chuyển và Lévy(λ) là một bước ngẫu nhiên theo phân phối Lévy với chỉ số λ. Dấu ⊕ biểu thị phép nhân thành phần theo quy luật của con số ngẫu nhiên.

Cơ chế thay thế: Nếu một cá thể x\_new được tạo ra có giá trị hàm mục tiêu cải thiện so với một cá thể x\_j ngẫu nhiên đã chọn, thì x\_j sẽ bị thay thế bởi x\_new. Công thức đánh giá này có thể biểu diễn dưới dạng:  
 Nếu f(x\_new) > f(x\_j) thì x\_j ← x\_new  
(với f(x) là hàm mục tiêu cần tối ưu hóa).

## 2.7. Vai trò của yếu tố ngẫu nhiên và định hướng trong quá trình tối ưu

Một trong những điểm mạnh quan trọng của thuật toán Cuckoo Search chính là khả năng kết hợp yếu tố ngẫu nhiên với định hướng dựa trên kinh nghiệm – được thể hiện rõ qua Lévy Flight và quy tắc thay thế tổ. Cụ thể:

* Yếu tố ngẫu nhiên: Các bước nhảy theo Lévy Flight đảm bảo rằng vùng tìm kiếm luôn được mở rộng, tránh việc chỉ tập trung vào một vùng cụ thể của không gian giải pháp, từ đó giảm nguy cơ rơi vào bẫy tối ưu cục bộ.
* Định hướng dựa trên kinh nghiệm: Thông qua quy tắc trao đổi tổ, các cá thể có kinh nghiệm (tức là những tổ có hàm mục tiêu tốt) sẽ được duy trì và phát triển, giúp thuật toán dần dần hội tụ về phía giải pháp tối ưu.

## 2.8. Sự cạnh tranh và đồng đẳng trong quần thể

Cơ chế cạnh tranh giữa các cá thể được coi là yếu tố động lực thúc đẩy quá trình hội tụ của thuật toán, qua đó đạt được sự tối ưu hóa hiệu quả:

* Chiến lược đánh giá và sàng lọc: Mỗi lần một cá thể mới được tạo ra, một cá thể hiện có trong quần thể được chọn ngẫu nhiên để so sánh. Nếu cá thể mới mang lại kết quả cải thiện, cá thể hiện có sẽ bị loại bỏ. Chiến lược này liên tục “lọc” ra những giải pháp yếu kém, giữ lại những tổ tiềm năng nhất cho quá trình tìm kiếm ở những vòng lặp tiếp theo.
* Đồng đẳng trong cạnh tranh: Các cá thể trong quần thể được xem là bình đẳng trong việc cạnh tranh giành chỗ đứng. Điều này giúp duy trì sự đa dạng của giải pháp, mặc dù một số cá thể có giá trị hàm mục tiêu kém sẽ bị loại bỏ, nhưng số lượng đủ lớn các cá thể mới liên tục được giới thiệu thông qua các bước ngẫu nhiên sẽ đảm bảo sự đa dạng cần thiết cho quá trình tìm kiếm.

## 2.9. Ứng dụng của quy tắc thay thế tổ trong các bài toán thực tiễn

Quy tắc thay thế tổ không chỉ giúp thuật toán duy trì sự cải thiện liên tục mà còn có ứng dụng thực tiễn trong việc giải quyết các bài toán phức tạp:

* Bài toán tối ưu hóa hàm số: Bằng cách liên tục thay thế các giải pháp xấu bằng những giải pháp tốt hơn, CS đã chứng minh khả năng hội tụ nhanh và chính xác trong các bài toán tối ưu hóa hàm số có nhiều cực trị.
* Bài toán hoạch định và thiết kế: Trong các trường hợp cần tìm kiếm thiết kế tối ưu hoặc nghiệm của một hệ thống phức tạp, cơ chế cạnh tranh giữa các tổ giúp thuật toán khai thác và cập nhật những vùng có khả năng chứa nghiệm tốt nhất.
* Ứng dụng trong khoa học máy tính: CS đã được áp dụng cho các bài toán liên quan đến xử lý dữ liệu, học máy và tối ưu hóa các hệ thống mạng, nhờ vào khả năng linh hoạt và hiệu quả của thuật toán.

## 2.10. Kết luận

Qua đó, nguyên lý hoạt động của thuật toán Cuckoo Search xây dựng dựa trên một cơ chế kết hợp giữa yếu tố ngẫu nhiên và quy tắc chọn lọc tự nhiên, tạo nên một quy trình tìm kiếm hiệu quả, mở rộng khả năng khám phá không gian giải pháp đồng thời đảm bảo khai thác các giải pháp tiềm năng để hướng đến kết quả tối ưu. Những cơ chế này không chỉ giúp thuật toán hội tụ nhanh mà còn đảm bảo độ ổn định trong quá trình chạy, đặc biệt trong các bài toán có số lượng cực trị cục bộ nhiều, qua đó mở ra nhiều hướng ứng dụng và nghiên cứu tương lai trong lĩnh vực tối ưu hóa.

**CHƯƠNG 3. ỨNG DỤNG THUẬT TOÁN CUCKOO SEARCH TRONG CÁC BÀI TOÁN TỐI ƯU HÓA**

**3.1. Giới thiệu**

Chương này đi sâu vào việc ứng dụng thuật toán Cuckoo Search (CS) để giải quyết ba bài toán tối ưu hóa thực tế: tối ưu hóa tham số cho máy học (SVM), tối ưu hóa trọng số mạng nơ-ron và giải bài toán định tuyến (TSP). Chúng ta sẽ khám phá lý thuyết cơ bản của CS, phân tích chi tiết mã Python được sử dụng và đánh giá kết quả thực nghiệm.

**3.2. Lý thuyết nền tảng của Thuật toán Cuckoo Search (CS)**

CS lấy cảm hứng từ hành vi sinh sản của chim cu cu, đặc biệt là chiến lược ký sinh tổ. Thuật toán hoạt động dựa trên ba quy tắc chính:

1. Mỗi con cu cu đẻ một quả trứng (giải pháp) tại một tổ được chọn ngẫu nhiên.
2. Những tổ có trứng chất lượng cao (giải pháp tốt hơn) sẽ được giữ lại cho thế hệ tiếp theo.
3. Số lượng tổ có sẵn là cố định. Chim chủ có thể phát hiện trứng lạ với xác suất pa​, và khi đó, chúng sẽ vứt bỏ trứng hoặc xây tổ mới.

CS sử dụng bước nhảy Levy flight để tạo ra các giải pháp mới, giúp thuật toán khám phá không gian tìm kiếm rộng hơn và tránh bị mắc kẹt ở các cực trị cục bộ.

**3.3. Triển khai và Ứng dụng CS trong các bài toán tối ưu hóa**

***3.3.1. Tối ưu hóa Tham số SVM***

a. Mã Python và Giải thích:

*def ham\_muc\_tieu\_svm(tham\_so):*

*C, gamma = tham\_so*

*model = SVC(C=C, gamma=gamma, kernel='rbf')*

*model.fit(X\_train, y\_train)*

*y\_du\_doan = model.predict(X\_test)*

*return -accuracy\_score(y\_test, y\_du\_doan)*

Hàm ham\_muc\_tieu\_svm đánh giá hiệu suất của mô hình SVM dựa trên các tham số C và gamma. CS sẽ tối thiểu hóa giá trị âm của độ chính xác, tương đương với việc tối đa hóa độ chính xác.

b. Các bước thực hiện và đánh giá kết quả:

1. Dữ liệu Iris được chia thành tập huấn luyện và tập kiểm tra.
2. CS được khởi tạo với hàm mục tiêu ham\_muc\_tieu\_svm, số chiều là 2 (C, gamma), và giới hạn tìm kiếm phù hợp.
3. CS thực hiện quá trình tối ưu hóa, tìm ra các giá trị C và gamma tối ưu.
4. Mô hình SVM được huấn luyện với các tham số tối ưu và đánh giá trên tập kiểm tra.
5. Kết quả đầu ra hiển thị các giá trị C, gamma tối ưu và độ chính xác tương ứng.
6. Vẽ đồ thị hội tụ của thuật toán CS.

c. Hình ảnh minh họa:

A graph with a line

AI-generated content may be incorrect.

*Hình 4- Đồ thị hội tụ Cuckoo Search - SVM.*

A number on a white background

AI-generated content may be incorrect.

Hình 5- Kết quả tối ưu SVM.

***3.3.2. Tối ưu hóa Trọng số Mạng Nơ-ron***

a. Mã Python và Giải thích:

*def ham\_muc\_tieu\_ann(trong\_so):*

*W1 = trong\_so[:20].reshape(4, 5)*

*b1 = trong\_so[20:25]*

*W2 = trong\_so[25:40].reshape(5, 3)*

*b2 = trong\_so[40:]*

*hidden = 1 / (1 + np.exp(-(np.dot(X\_train, W1) + b1)))*

*output = 1 / (1 + np.exp(-(np.dot(hidden, W2) + b2)))*

*return np.mean((output - y\_train\_onehot) \*\* 2)*

Hàm ham\_muc\_tieu\_ann tính toán loss của mạng nơ-ron dựa trên các trọng số đầu vào. CS sẽ tối thiểu hóa loss này để tìm ra bộ trọng số tối ưu.

b. Các bước thực hiện và đánh giá kết quả:

1. Dữ liệu Iris được mã hóa one-hot.
2. CS được khởi tạo với hàm mục tiêu ham\_muc\_tieu\_ann, số chiều tương ứng với tổng số trọng số, và giới hạn tìm kiếm cho các trọng số.
3. CS thực hiện quá trình tối ưu hóa, tìm ra bộ trọng số tối ưu.
4. Loss của mạng nơ-ron với trọng số tối ưu được tính toán trên tập huấn luyện.
5. Kết quả đầu ra hiển thị giá trị loss tối ưu.
6. Vẽ đồ thị hội tụ của thuật toán CS.

c. Dữ liệu đầu vào:

* Dữ liệu Iris đã được mã hóa one-hot.

d. Hình ảnh minh họa:

A graph with numbers and a line

AI-generated content may be incorrect.

Hình 6- Đồ thị hội tụ Cuckoo Search - ANN.

A black text on a white background

AI-generated content may be incorrect.

*Hình 7- Kết quả tối ưu hóa trọng số mạng nơ-ron, hiển thị loss tối ưu.*

***3.3.3. Giải Bài toán Định tuyến (TSP)***

a. Mã Python và Giải thích:

*def ham\_muc\_tieu\_tsp(duong\_di):*

*duong\_di = duong\_di.astype(int)*

*return sum(distance\_matrix[duong\_di[i], duong\_di[i+1]] for i in range(len(duong\_di)-1))*

Hàm ham\_muc\_tieu\_tsp tính toán tổng khoảng cách của một lộ trình dựa trên ma trận khoảng cách. CS sẽ tối thiểu hóa tổng khoảng cách này.

*# Vẽ đồ thị TSP tối ưu*

*G = nx.Graph()*

*for i in range(num\_cities):*

*for j in range(i+1, num\_cities):*

*G.add\_edge(i, j, weight=distance\_matrix[i, j])*

*pos = nx.spring\_layout(G)*

*nx.draw(G, pos, with\_labels=True, node\_color='lightblue', edge\_color='gray')*

*# Vẽ lộ trình tối ưu trên đồ thị*

*edges = [(best\_tsp\_path[i], best\_tsp\_path[i+1]) for i in range(len(best\_tsp\_path)-1)]*

*nx.draw\_networkx\_edges(G, pos, edgelist=edges, edge\_color='red', width=2)*

*plt.title('Lộ trình TSP tối ưu')*

*plt.show()*

Đoạn mã này thực hiện việc vẽ đồ thị TSP tối ưu:

1. Tạo đồ thị: Sử dụng networkx.Graph() để tạo một đồ thị vô hướng.
2. Thêm cạnh: Thêm các cạnh vào đồ thị, với trọng số là khoảng cách giữa các thành phố.
3. Sắp xếp vị trí: Sử dụng nx.spring\_layout(G) để sắp xếp vị trí các thành phố trên đồ thị.
4. Vẽ đồ thị: Sử dụng nx.draw(G, pos, with\_labels=True, node\_color='lightblue', edge\_color='gray') để vẽ đồ thị với nhãn thành phố, màu nút và màu cạnh.
5. Vẽ lộ trình tối ưu: Sử dụng nx.draw\_networkx\_edges(G, pos, edgelist=edges, edge\_color='red', width=2) để vẽ lộ trình tối ưu trên đồ thị, với màu cạnh đỏ và độ rộng 2.
6. Hiển thị đồ thị: Sử dụng plt.show() để hiển thị đồ thị.

b. Các bước thực hiện và đánh giá kết quả:

1. Ma trận khoảng cách distance\_matrix được định nghĩa.
2. CS được khởi tạo với hàm mục tiêu ham\_muc\_tieu\_tsp, số chiều là số thành phố, và giới hạn tìm kiếm cho các thành phố.
3. CS thực hiện quá trình tối ưu hóa, tìm ra lộ trình tối ưu.
4. Tổng khoảng cách của lộ trình tối ưu được tính toán.
5. Kết quả đầu ra hiển thị lộ trình tối ưu và tổng khoảng cách.
6. Vẽ đồ thị hội tụ của thuật toán CS và đồ thị TSP tối ưu.

c. Hình ảnh minh họa:

A graph with blue lines

AI-generated content may be incorrect.

Hình 8- Đồ thị hội tụ Cuckoo Search - TSP.

A black and white text

AI-generated content may be incorrect.

*Hình 9- Kết quả tối ưu hóa lộ trình TSP, hiển thị lộ trình và tổng khoảng cách.*

A red and grey lines with blue dots and numbers

AI-generated content may be incorrect.

Hình 10- Đồ thị TSP tối ưu.

d. Phân tích thêm:

* Đồ thị TSP tối ưu giúp chúng ta trực quan hóa lộ trình tìm được, dễ dàng nhận thấy thứ tự các thành phố và tổng khoảng cách.
* Việc vẽ đồ thị cũng giúp chúng ta kiểm tra tính hợp lý của kết quả, đảm bảo rằng lộ trình tìm được là một chu trình hợp lệ và có tổng khoảng cách nhỏ nhất.
* Trong đồ thị TSP, các nút (nodes) đại diện cho các thành phố, và các cạnh (edges) đại diện cho khoảng cách giữa chúng. Lộ trình tối ưu được hiển thị bằng các cạnh màu đỏ, giúp chúng ta dễ dàng theo dõi đường đi.
* Việc sử dụng networkx giúp chúng ta dễ dàng thao tác với đồ thị, vẽ và hiển thị kết quả.

**3.4. Đánh giá và Thảo luận**

Kết quả thực nghiệm cho thấy CS là một thuật toán hiệu quả trong việc giải quyết các bài toán tối ưu hóa khác nhau. Khả năng khám phá không gian tìm kiếm rộng lớn của CS, nhờ bước nhảy Levy flight, giúp thuật toán tìm ra các giải pháp tốt trong thời gian hợp lý.

Tuy nhiên, hiệu suất của CS có thể bị ảnh hưởng bởi các tham số như số lượng tổ, số vòng lặp, xác suất pa và hệ số alpha. Việc điều chỉnh các tham số này một cách cẩn thận là cần thiết để đạt được kết quả tốt nhất.

**3.5. Link Github:**

*Link github code bằng colab:* [*https://github.com/TVLlam/1771020412\_CNTT17-01\_TRAN\_VAN\_LAM/blob/8d6ee6a39f7787a2fe4ac5d65a576187fbe7bb5e/NHOM2\_TTNT.ipynb*](https://github.com/TVLlam/1771020412_CNTT17-01_TRAN_VAN_LAM/blob/8d6ee6a39f7787a2fe4ac5d65a576187fbe7bb5e/NHOM2_TTNT.ipynb)

*Link github code bằng vscode:*

**KẾT LUẬN**

Thuật toán Cuckoo Search (CS) nổi bật với khả năng khám phá không gian tìm kiếm rộng lớn, nhờ vào cơ chế bước nhảy Levy flight độc đáo. Điều này cho phép CS tìm ra các giải pháp tối ưu trong các bài toán phức tạp, nơi không gian tìm kiếm có nhiều cực trị cục bộ. Tính đơn giản trong cài đặt và sử dụng cũng là một ưu điểm lớn, giúp CS trở thành lựa chọn hấp dẫn cho nhiều nhà nghiên cứu và kỹ sư. Ngoài ra, CS đã chứng minh được hiệu quả trong việc giải quyết các bài toán tối ưu hóa đa mục tiêu, mở ra nhiều ứng dụng tiềm năng trong các lĩnh vực khác nhau.

Mặc dù CS mang lại nhiều lợi ích, hiệu suất của thuật toán phụ thuộc đáng kể vào việc lựa chọn các tham số như số lượng tổ, số vòng lặp, xác suất pa và hệ số alpha. Việc điều chỉnh các tham số này đòi hỏi kinh nghiệm và thử nghiệm, có thể gây khó khăn cho người mới sử dụng. Trong một số trường hợp, CS có thể bị mắc kẹt ở các cực trị cục bộ nếu không điều chỉnh tham số phù hợp, dẫn đến kết quả không tối ưu. Đối với các bài toán có kích thước lớn, CS có thể yêu cầu thời gian tính toán đáng kể, đặc biệt là khi số vòng lặp tăng lên.

Để nâng cao hiệu suất và mở rộng ứng dụng của CS, có nhiều hướng phát triển tiềm năng. Một trong số đó là kết hợp CS với các thuật toán tối ưu hóa khác, như thuật toán di truyền hoặc thuật toán đàn kiến, để tận dụng ưu điểm của từng thuật toán và khắc phục nhược điểm. Việc áp dụng CS vào các bài toán tối ưu hóa phức tạp hơn, như tối ưu hóa mạng nơ-ron sâu hoặc tối ưu hóa chuỗi cung ứng, cũng là một hướng đi đầy hứa hẹn. Ngoài ra, việc nghiên cứu và phát triển các phương pháp tối ưu hóa tham số tự động cho CS sẽ giúp giảm bớt gánh nặng cho người sử dụng và nâng cao hiệu quả của thuật toán.

**DANH MỤC TÀI LIỆU THAM KHẢO**

[1] Russell, S. J., & Norvig, P. (2016). *Artificial intelligence: a modern approach. Pearson Education.*

[2] Engelbrecht, A. P. (2007). *Computational intelligence: an introduction.* John Wiley & Sons.

[3] Yang, X. S. (2014). *Nature-inspired optimization algorithms*. Elsevier.

[4] Mitchell, T. M. (1997). *Machine learning*. McGraw-Hill.

[5] Larose, D. T., & Larose, C. D. (2019). *Data science using Python and R. John Wiley & Sons*.