TRƯỜNG ĐẠI HỌC CÔNG NGHỆ GIAO THÔNG VẬN TẢI

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO BÀI TẬP LỚN MÔN HỌC**

**TRÍ TUỆ NHÂN TẠO**

**Tên đề tài:**

**Ant colony optimization (ACO)**

**Và Seagull optimization algorithm (SOA)**

|  |  |
| --- | --- |
| **GIÁO VIÊN HƯỚNG DẪN:** | **TS. ĐỖ BẢO SƠN** |
| **SINH VIÊN THỰC HIỆN:** | **NINH THỊ THU HUYỀN**  **BÙI THỊ PHƯƠNG**  **NGUYỄN DUY PHÚ**  **ĐỚI SỸ NAM** |
| **LỚP:** | **74DCTT26** |

**HÀ NỘI 2025**

**MỤC LỤC**

[DANH MỤC CÁC TỪ VIẾT TẮT 1](#_Toc200561284)

[MỞ ĐẦU 2](#_Toc200561285)

[CHƯƠNG 1. CÁC THUẬT TOÁN LẤY CẢM HỨNG TỪ TỰ NHIÊN 3](#_Toc200561286)

[**1.1. Giới thiệu chung** 3](#_Toc200561287)

[**1.2. Phân loại** 3](#_Toc200561288)

[CHƯƠNG 2. THUẬT TOÁN ANT COLONY OPTIMIZATION 5](#_Toc200561289)

[**2.1. Tổng quan về thuật toán Ant colony optimization** 5](#_Toc200561290)

[***2.1.1. Lịch sử ra đời*** 5](#_Toc200561291)

[***2.1.2. Động lực hình thành của thuật toán*** 6](#_Toc200561292)

[***2.1.3. Ứng dụng của thuật toán*** 6](#_Toc200561293)

[**2.2. Cấu trúc và nguyên lý hoạt động của thuật toán** 7](#_Toc200561294)

[***2.2.1. Sơ đồ thuật toán*** 7](#_Toc200561295)

[***2.2.3. Nguyên lý hoạt động thuật toán*** 9](#_Toc200561296)

[***2.2.4. Các bước giải quyết bài toán đàn kiến*** 10](#_Toc200561297)

[***2.2.5. Các sơ đồ thuật toán khác phát triển trên mô hình ACO*** 11](#_Toc200561298)

[CHƯƠNG 3. THUẬT TOÁN SEAGULL OPTIMIZATION ALGORITHM 19](#_Toc200561299)

[**3.1. Tổng quan về thuật toán Seagull optimization algorithm** 19](#_Toc200561300)

[**3.2. Cấu trúc và nguyên lý hoạt động của thuật toán** 20](#_Toc200561301)

[***3.2.1. Sơ đồ thuật toán*** 20](#_Toc200561302)

[***3.2.2. Mô hình thuật toán*** 22](#_Toc200561303)

[***3.2.3. Nguyên lý hoạt động thuật toán*** 27](#_Toc200561304)

[***3.2.4. Các thuật toán khác phát triển trên mô hình SOA*** 28](#_Toc200561305)

[CHƯƠNG 4. MÔ PHỎNG THUẬT TOÁN 29](#_Toc200561306)

[**4.1. Thuật toán Ant colony optimization** 29](#_Toc200561307)

[***4.1.1. Bài toán người du lịch*** 29](#_Toc200561308)

[***4.1.2. Mô phỏng thuật toán*** 32](#_Toc200561309)

[**4.2. Thuật toán Seagull optimization algorithm** 32](#_Toc200561310)

[***4.2.1. Bài toán người giao hàng*** 32](#_Toc200561311)

[***4.2.2. Mô phỏng thuật toán*** 34](#_Toc200561312)

[KẾT LUẬN 36](#_Toc200561313)

# **DANH MỤC CÁC TỪ VIẾT TẮT**

|  |  |  |
| --- | --- | --- |
| **STT** | **Kí hiệu từ viết tắt** | **Chữ viết đầy đủ** |
| 1 | ACO | Ant Colony Optimization |
| 2 | SOA | Seagull Optimization Algorithm |
| 3 | NP-hard | Non-deterministic Polynomial-time hard |
| 4 | AS | Ant System |
| 5 | TSP | Traveling Salesman Problem |
| 6 | QAP | Quadratic Assignment Problem |
| 7 | MMAS | MAX-MIN Ant System |
| 8 | ACS | Ant Colony System |
| 9 | RBAS | Rank-Based Ant System |
| 10 | BWAS | Best-Worst Ant System |
| 11 | PSO | Particle Swarm Optimization |
| 12 | GA | Genetic Algorithm |
| 13 | MO-SOA | Multi-objective SOA |
| 14 | DE | Differential Evolution |

# **MỞ ĐẦU**

Hiện nay, các bài toán tối ưu ngày càng xuất hiện nhiều trong các lĩnh vực thực tế như kỹ thuật, logistics, tài chính, quản lý sản xuất, … Phần lớn trong số đó là những bài toán phức tạp, thuộc nhóm NP-hard, tức là không thể giải quyết triệt để bằng các phương pháp truyền thống trong thời gian hợp lý khi kích thước bài toán tăng lên. Các phương pháp tìm kiếm vét cạn chỉ phù hợp với bài toán quy mô nhỏ, trong khi đó các bài toán thực tế thường đòi hỏi lời giải xấp xỉ tối ưu trong thời gian ngắn. Chính vì vậy, các thuật toán mô phỏng tự nhiên – lấy cảm hứng từ hành vi thích nghi và sinh tồn của các sinh vật trong tự nhiên đã và đang được ứng dụng rộng rãi trong lĩnh vực tối ưu hóa. Trong số đó, Ant Colony Optimization (ACO) – Thuật toán tối ưu đàn kiến mô phỏng quá trình tìm đường đi ngắn nhất của đàn kiến, và Seagull Optimization Algorithm (SOA) – Thuật toán tối ưu chim mòng biển dựa trên hành vi săn mồi của chim mòng biển, là hai thuật toán tiêu biểu với nhiều tiềm năng ứng dụng.

Đề tài này tập trung nghiên cứu tổng quan về các thuật toán mô phỏng tự nhiên, đi sâu vào Ant Colony Optimization và Seagull Optimization Algorithm, từ nguyên lý hoạt động, sơ đồ giải thuật đến các biến thể và ứng dụng thực tiễn. Đồng thời, đề tài còn xây dựng chương trình mô phỏng hai thuật toán nhằm kiểm chứng hiệu quả thông qua các bài toán tối ưu cụ thể.

# **CHƯƠNG 1. CÁC THUẬT TOÁN LẤY CẢM HỨNG TỪ TỰ NHIÊN**

## **1.1. Giới thiệu chung**

Tự nhiên từ lâu đã trở thành nguồn cảm hứng phong phú cho con người, đặc biệt trong việc phát triển các hệ thống thông minh. Nhìn vào cách các loài sinh vật tồn tại, sinh trưởng và thích nghi qua hàng triệu năm tiến hóa, các nhà khoa học đã nhận ra rằng những cơ chế trong tự nhiên – dù đơn giản hay phức tạp – đều có thể gợi mở cách giải quyết cho các vấn đề khó nhằn trong thế giới hiện đại. Từ việc quan sát bầy kiến tìm đường đến tổ, đàn chim bay theo đội hình, hay cách gen di truyền trong quá trình tiến hóa – tất cả đều mang lại ý tưởng cho các thuật toán mô phỏng hành vi tự nhiên.

Kể từ khi máy tính ra đời, các nhà nghiên cứu không ngừng tìm cách bắt chước tự nhiên để xây dựng các thuật toán thông minh. Những thuật toán này, gọi chung là thuật toán lấy cảm hứng từ tự nhiên, có khả năng tự học hỏi, thích nghi và giải quyết hiệu quả các bài toán phức tạp – đặc biệt là những bài toán tối ưu hóa mà các phương pháp truyền thống gặp khó khăn.

Điểm đặc biệt của các thuật toán này là chúng không cần biết rõ công thức bài toán, không phụ thuộc vào đạo hàm và có thể mở rộng tốt khi làm việc với lượng dữ liệu lớn. Chúng thường mang các đặc tính giống tự nhiên như: có yếu tố ngẫu nhiên, hoạt động phi tuyến tính, tự tổ chức, linh hoạt với môi trường thay đổi, và có tính song song cao. Nhờ vậy, các thuật toán này đang được ứng dụng ngày càng rộng rãi trong nhiều lĩnh vực như trí tuệ nhân tạo, điều khiển robot, xử lý ảnh, y học, thiết kế kỹ thuật và nhiều lĩnh vực tối ưu hóa khác.

## **1.2. Phân loại**

Để hiểu rõ hơn, các thuật toán lấy cảm hứng từ tự nhiên thường được chia thành 4 nhóm chính: dựa trên tiến hóa, dựa trên sinh học, dựa trên hành vi xã hội – văn hóa, và dựa trên các quy luật khoa học tự nhiên. Mỗi nhóm phản ánh một khía cạnh của tự nhiên và đều có những ứng dụng thực tế rất đa dạng.

|  |  |  |
| --- | --- | --- |
| **Nhóm chính** | **Phân nhóm phụ/ hành vi mô phỏng** | **Một số thuật toán tiêu biểu** |
| Dựa trên sự tiến hóa (Evolution-based) | - Lập trình di truyền (Genetic programming)  - Chiến lược tiến hóa (Evolutionary strategies)  - Thuật toán di truyền (Genetic algorithms)  - Lập trình tiến hóa (Evolutionary programming) | - Genetic Algorithm (GA)  - Genetic Programming (GP)  - Evolutionary Strategy (ES)  - Evolutionary Programming (EP) |
| Dựa trên sinh học (Biological-based) | \* Dựa trên hành vi xã hội (Social behavior-based):  - Hành vi kiếm ăn (Foraging behavior)  - Hành vi định hướng (Navigation behavior)  - Hành vi ký sinh (Parasitic behavior)  - Hành vi cạnh tranh (Competitive behavior)  - Hành vi sinh sản (Reproductive behavior)  \*Dựa trên hệ miễn dịch (Immune system-based):  - Hệ miễn dịch nhân tạo (Artificial immune system) | \*Hành vi xã hội:  - PSO (Particle Swarm Optimization)  - ACO (Ant Colony Optimization)  - ABC (Artificial Bee Colony)  - GWO (Grey Wolf Optimizer)  - CS (Cuckoo Search)  \*Hệ miễn dịch:  - Artificial Immune System (AIS) |
| Dựa trêm hành vi xã hội – văn hóa (Social-culture-based) | - Động não (Brain storm-based)  - Dựa trên văn hóa (Culture-based)  - Cạnh tranh đế quốc (Imperialist competitive-based)  - Dạy – học (Teaching–learning-based) | - Teaching–Learning-Based Optimization (TLBO)  - Brain Storm Optimization (BSO)  - Imperialist Competitive Algorithm (ICA)  - Cultural Algorithm |
| Dựa trên khoa học tự nhiên (Science-based) | - Dựa trên vật lý (Physics-based)  - Dựa trên địa lý (Geography-based)  - Dựa trên toán học (Mathematics-based) | - Simulated Annealing (SA)  - Gravitational Search Algorithm (GSA)  - Sine Cosine Algorithm (SCA)  - Chaos-based Algorithms |

# **CHƯƠNG 2. THUẬT TOÁN ANT COLONY OPTIMIZATION**

## **2.1. Tổng quan về thuật toán Ant colony optimization**

### ***2.1.1. Lịch sử ra đời***

ACO (Ant Colony Optimization Tối ưu đàn kiến) là một phương pháp nghiên cứu lấy cảm hứng từ việc mô phỏng hành vi của đàn kiến trong tự nhiên nhằm mục tiêu giải quyết các bài toán tối ưu phức tạp.

Hệ thống ACO lần đầu tiên được Marco Dorigo giới thiệu trong luận văn của mình vào năm 1992, và được gọi là Hệ thống kiến (Ant System, hay AS). AS là kết quả của việc nghiên cứu trên hướng tiếp cận trí tuệ máy tính nhằm tối ưu tổ hợp mà Dorigo được hướng dẫn ở Politecnico di milano với sự hợp tác của Alberto Colorni và Vittorio Maniezzo. AS ban đầu được áp dụng cho bài toán người du lịch (TSP) và QAP

Cũng vào năm 1992, tại hội nghị sự sống nhân tạo lần đầu tiên ở châu Âu, Dorigo và các cộng sự đã công bố bài: sự tối ưu được phân bố bởi đàn kiến.

Tiếp theo tại hội nghị quốc tế thứ hai về giải quyết các vấn đề song song trong tự nhiên ở Hà Lan (1992), ông và các cộng sự đã công bố bài: nghiên cứu về các đặc tính của một giải thuật kiến.

Kể từ năm 1995 Dorigo, Gambardella và Stützle đã phát triển các sơ đồ AS khác nhau. Dorigo và Gambardella đã đề xuất Hệ thống bầy kiến (Ant Colony System, hay ACS) trong khi Stützle and Hoos đề xuất MAX-MIN Ant System (MMAS). Tất cả đều áp dụng cho bài toán người du lịch đối xứng hay không đối xứng và cho kết quả mỹ mãn. Dorigo, Gambardella and Stützle cũng đề xuất những phiên bản lai của ACO với tìm kiếm địa phương.

Vào năm 1995, L.M. Gambardella và M. Dorigo đã đề xuất hệ thống Ant-Q, là một cách tiếp cận học tăng cường cho cho bài toán TSP.Và nó được áp dụng trong Học Máy.

Tiếp đó, vào năm 1996, trong bài báo công nghệ của mình tại Bruxelles M. Dorigo và L.M. Gambardella đã công bố hệ thống Ant Conoly System. Đây là hệ thống đề cập đến cách học phối hợp áp dụng cho bài toán TSP.

Cũng trong năm 1996 này, T.Stützle và H. H. Hoos đã đề xuất hệt thống Max-Min Ant System . Đây là một hệ thống cải tiến hệ thống AntSystem ban đầu và được đánh giá là hệ thống tính toán trong tương lai.

Sau đó, vào năm 1997, G. Di Caro và M. Dorigo đã đề xuất hệ thống AntNet. Đây là cách tiếp cận về định hướng sự thích nghi. Và phiên bản cuối cùng của hệ thống AntNet về điều khiển mạng truyền thông đã được công bố vào năm 1998.

Cũng trong năm 1997, hệ thống Rank-based Ant System, một hệ thống cải tiến hệ thống kiến ban đầu về nghiên cứu hệ thống tính toán đã được đề xuất bởi B. Bullnheimer, R. F. Hartl và C. Strauss. Phiên bản cuối cùng của hệ thống này được công bố vào năm 1999.

Vào năm 2001, C. Blum, A. Roli, và M. Dorigo đã cho công bố về hệ thống kiến mới là Hyper Cube – ACO. Phiên bản mở rộng tiếp đó đã được công bố vào năm 2004.

Hầu hết các nghiên cứu gần đây về ACO tập trung vào việc phát triển các thuật toán biến thể để làm tăng hiệu năng tính toán của thuật toán Ant System ban đầu.

### ***2.1.2. Động lực hình thành của thuật toán***

Giải thuật Đàn Kiến (Ant Colony Optimization - ACO) ra đời từ nhu cầu tìm kiếm các phương pháp giải quyết hiệu quả các bài toán tối ưu tổ hợp, đặc biệt là những bài toán có không gian tìm kiếm rộng lớn và độ phức tạp cao như Bài toán Người bán hàng (Travelling Salesman Problem - TSP), Bài toán lập lịch, và các bài toán định tuyến mạng.

Cảm hứng cho ACO đến từ hành vi tìm đường đến nguồn thức ăn của loài kiến trong tự nhiên. Mặc dù mỗi cá thể kiến có trí tuệ hạn chế, nhưng thông qua cơ chế tập thể và sự trao đổi thông tin gián tiếp thông qua pheromone (chất hóa học do kiến tiết ra trên đường đi), cả đàn kiến có thể tìm ra được con đường ngắn nhất từ tổ đến nguồn thức ăn. Điều này thể hiện khả năng thích nghi và học tập phân tán hiệu quả, thu hút sự chú ý của các nhà nghiên cứu trong lĩnh vực trí tuệ nhân tạo và tính toán tự nhiên.

### ***2.1.3. Ứng dụng của thuật toán***

- Bài toán lập lịch tổng quát

- Phân luồng giao thông

- Định tuyến mạng

- Tìm kiếm thông tin trên mạng

- Phân công lao động

…

Đây đều là các bài toán NP-hard, rất khó giải nhanh với độ chính xác cao bằng các thuật toán truyền thống.

## **2.2. Cấu trúc và nguyên lý hoạt động của thuật toán**

### ***2.2.1. Sơ đồ thuật toán***



*Hình 2.2.1. Sơ đồ thuật toán ACO*

*Thuật toán:*

Procedure Thuật toán ACO;

Begin

Khởi tạo tham số, ma trận mùi, khởi tạo m con kiến;

**repeat**

**for** k=1 **to** m **do**

Kiến k xây dựng lời giải;

**end-for**

Cập nhật mùi;

Cập nhật lời giải tốt nhất;

**until** (Điều kiện kết thúc);

Đưa ra lời giải tốt nhất;

**End;**

**\* Đánh giá độ phức tạp của thuật toán**

Giả sử bài toán có các tham số:

* n là số lượng đỉnh
* m là số lượng kiến
* t số vòng lặp
* k số lượng lân cận được xét khi kiến chọn các bước tiếp theo

Mỗi con kiến cần đi qua toàn bộ các đỉnh, mỗi bước phải chọn một đỉnh mới trong số các đỉnh còn lại

* Mỗi con kiến mất O() để hoàn thành chuyến đi.

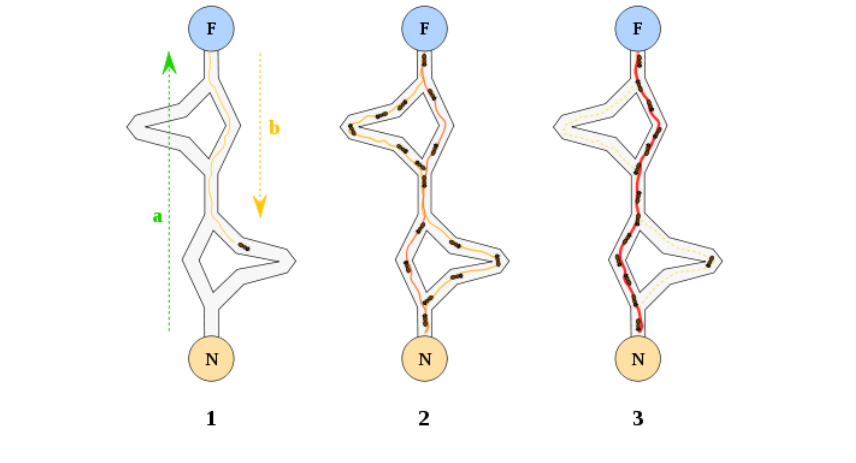
Sau khi tất cả kiến hoàn thành chuyến đi, pheromone trên các cạnh được cập nhật. Có thể là trên tất cả các cạnh O(), hoặc chỉ trên các canhh được sử dụng bởi kiến.

Như vậy, tổng thời gian cho mỗi vòng lặp là:

O(m. + ) = O(m.).

* Tổng thời gian cho toàn bộ thuật toán (với t vòng lặp): O(t.m.)

### ***2.2.3. Nguyên lý hoạt động thuật toán***



*Hình 2.2.3. Nguyên lý tìm đường đi của kiến*

**Quy luật đi của đàn kiến:**

- Kiến điều hướng từ nơi tổ của chúng tới nguồn thức ăn. Và chúng bị mù do đó chúng định hướng bằng dấu vết mà chúng để lại (pheromone)!

- Đường đi ngắn nhất được phát hiện thông qua pheromone mà chúng đã để lại trên đường đi.

- Mỗi con kiến đều di chuyển một cách ngẫu nhiên.

- Pheromone được kiến lưu lại trên toàn bộ quãng đường đi của chúng.

- Đoạn đường nào có nhiều pheromone hơn sẽ được đi theo nhiều hơn.

**Mỗi con kiến sẽ chọn đường đi theo các nguyên tắc:**

- Dựa vào thông tin mùi để lại có trên các đoạn đường để tính xác suất chọn đoạn đường tiếp theo.

- Đường đi nào có lượng mùi nhiều hơn thì sẽ được gán một xác suất lớn hơn.

- Ngược lại các đoạn đường đi có lượng thông tin mùi thấp hơn sẽ có xác suất được chọn thấp hơn.

=> Con kiến sẽ tìm đường cho tới khi hoàn thành một đường đi của nó (thoả mãn điều kiện dừng của con kiến).

- Một đường đi đầy đủ và hoàn chỉnh được gọi là một lời giải đúng cho bài toán đặt ra.

- Các đường đi tìm được sẽ được phân tích và đánh giá để tìm phương án tối ưu nhất có thể. Đó là lời giải tối ưu của bài toán.

- Sau khi tất cả kiến trong đàn hoàn thành việc tìm đường đi của nó, ta sẽ tiến hành cập nhật thông tin mùi cho các cung.

=> Số lượng của mùi sẽ được tính toán và điều chỉnh để tìm được phương án tối ưu tốt hơn:

- Các lời giải có đường đi ngắn hơn sẽ có khối lượng pheromone lớn hơn để đặt trên các cung đã được đi qua. Ngược lại, các lời giải cho đường đi dài hơn sẽ có khối lượng pheromone bé hơn.

- Con kiến sẽ chọn đường đi có lượng mùi lớn hơn và xác suất chọn đường đi đó cũng cao hơn.

### ***2.2.4. Các bước giải quyết bài toán đàn kiến***

Từ thuật toán trên ta có thể rút ra các bước giải quyết một bài toán ứng dụng với thuật toán đàn kiến:

**Bước 1:** Thể hiện bài toán trong khung của tập các thành phần và sự chuyển đổi hoặc bởi một đồ thị được đánh dấu bởi kiến đề xây dựng cách giải quyết.

**Bước 2:** Định nghĩa thích hợp cho mùi lạ τrs là một xu hướng quyết định. Đó là một bước chủ yếu trong việc hình thanhg thuật toán ACO và xác định rõ mùi lạ không là một nhiệm vụ tầm thường và nó tính toán yêu cầu bên trong của bài toán sau đáp án.

**Bước 3:** Định nghĩa thích hợp kinh nghiệm cho mỗi quyết định để một con kiến có thể xây dựng cách giải quyết, ví dụ: định nghĩa thông tin kinh nghiệm ηrs kết hợp mỗi thành phần hoặc trạng thái chuyển đổi. Thông tin kinh nghiệm chủ yếu cho việc tìm kiếm tốt nếu thuật toán tìm kiếm vùng không có sẵn hoặc không thể ứng dụng.

**Bước 4:** Nếu thực hiện được, tạo ra một vùng thuật toán tìm kiếm hiệu quả cho bài toán sau đáp án bởi vì kết quả của nhiều ứng dụng ACO cho bài toán tổ hợp tối ưu NP-hard thể hiện qua sự tìm kiếm tốt nhất đạt được khi ACO có vùng lạc quan.

**Bước 5:** Lựa chọn một thuật toán ACO và ứng dụng nó vào những bài toán cần giải quyết.

**Bước 6:** Các tham số phù hợp của thuật toán ACO. Một điểm bắt đầu tốt cho tham số phù hợp là sử dụng cài đặt tham số để tìm kiếm tốt khi ứng dụng thuật toán ACO vào bài toán đơn giản hoặc các bài toán khác nhau. Một vấn đề khác chi phối thời gian trong nhiệm phù hợp là để sử dụng thủ tục động cho tham số phù hợp.

Nó nên xoá các bước tiếp có thể chỉ đưa ra một hướng dẫn sử dụng thuật toán ACO. Thêm nữa, việc sử dụng là sự kết hợp các quá trình ở đó với vài bài toán sâu hơn và hoạt động của thuật toán, vài lựa chọn ban đầu cần phải sửa lại. Cuối cùng, chúng ta muốn trên thực tế, điều quan trọng nhất của các bước là đầu tiên phải khớp bởi vì lựa chọn tồi ở trạng thái này tính không thể tính với một tham số gốc phù hợp tốt.

**Chú ý:** Khi áp dụng phương pháp ACO cho mỗi bài toán cụ thể, có ba yếu tố quyết định hiệu quả thuật toán:

1. *Xây dựng đồ thị cấu trúc thích hợp.* Việc xây dựng đồ thị cấu trúc để tìm được lời giải cho bài toán theo thủ tục tuân tự không khó. Khó khăn chính là với các bài toán cỡ lớn thì không gian tìm kiếm quá rộng, đòi hỏi ta sử dụng các ràng buộc 2 một cách hợp lý để giảm miền tìm kiếm cho mỗi con kiến. Cách xử lý bài toán suy diễn haplotype ở chương 4 minh họa cho điều này.
2. *Chọn thông tin heuristic.* Thông tin heuristic tốt sẽ tăng hiệu quả thuật toán. Tuy nhiên, nhiều bài toán ta không có thông tin này thì có thể đánh giá chúng như nhau. Khi đó lúc ban đầu, thuật toán chỉ đơn thuần chạy theo phương thức tìm kiếm ngẫu nhiên, vết mùi thể hiện định hướng của học tăng cường và thuật toán vẫn thực hiện được.
3. *Chọn quy tắc cập nhật mùi.* Quy tắc cập nhật mùi thể hiện chiến lược học của thuật toán. Nếu đồ thị cấu trúc và thông tin heuristic luôn phụ thuộc vào từng bài toán cụ thể thì quy tắc cập nhật mùi là yếu tố phổ dụng và thường dùng để đặt tên cho thuật toán. Có nhiều quy tắc cập nhật mùi đã được đề xuất, trong luận án này chúng tôi sẽ tìm quy tắc thích hợp cho hai loại bài toán tùy theo thông tin heuristic ảnh hưởng nhiều hay ít tới thủ tục tìm kiếm lời giải.

### ***2.2.5. Các sơ đồ thuật toán khác phát triển trên mô hình ACO***

***a. Thuật toán Ant System (AS)***

Được phát triển bởi Dorigo, Maniezzo và Colorni năm 1991, là thuật toán ACO đầu tiên. Ban đầu có 3 biến thể khác nhau là: AS-Density, ASQuantity và AS-Cycle khác nhau bởi cách thức cập nhật thông tin Pheromone.

Trong đó:

* AS-Density: Thì đàn kiến sẽ đặt thêm pheromone trong quá trình xây dựng lời giải (online step-by-step pheromone update), lượng pheromone để cập nhật là một hằng số.
* AS-Quantity: Thì đàn kiến sẽ đặt thêm pheromone trong quá trình xây dựng lời giải (online step-by-step pheromone update), lượng pheromone để cập nhật là phụ thuộc vào độ mong muốn (thông tin heuristic) với đoạn đường đi qua ηij
* AS-Cycle: Thông tin pheromone sẽ được cập nhật khi lời giải đã hoàn thành (online delayed pheromone update). Đây là mô hình cho kết quả tốt nhất và được coi như là thuật toán AS.
* **Quy tắc di chuyển của kiến**

Trong thuật toán AS, kiến xây dựng một đường đi bắt đầu tại một đỉnh được chọn ngẫu nhiên.

Tại đỉnh i, một con kiến k sẽ chọn đỉnh j chưa được đi qua trong tập láng giềng của i theo công thức sau:

, j∈

Trong đó:

: xác suất con kiến k lựa chọn cạnh (i,j);

: nồng độ vét mùi trên cạnh (i,j);

α : hệ số điều chỉnh ảnh hưởng trên

**:** thông tin heuristic giúp đánh giá chính xác

β : hệ số điều chỉnh ảnh hưởng trên

: tập các đỉnh láng giềng của i mà con kiến k chưa đi qua

Sự lựa chọn của con khi quyết định đi từ đỉnh i qua đỉnh j và được tính  
theo công thức:

: khoảng cách giữa đỉnh i và đỉnh j

* Quy tắc cập nhật thông tin mùi

Trong quá trình di chuyển tìm đường đi của đàn kiến, chúng thực hiện cập nhật thông tin mùi trên những đoạn đường mà chúng đã đi qua. Gắn với mỗi cạnh (i,j) nồng độ vết mùi và thông số heuristic trên cạnh đó.

Ban đầu nồng độ mùi trên mỗi cạnh (i,j) được khởi tạo một hằng số c, hoặc được xác định theo công thức:

, ∀(i,j);

Việc cập nhập pherpmone được tiến hành như sau:

* Đầu tiên tất cả pheromone trên các cung sẽ được giảm đi bởi một lượng:

Với p trong khoảng (0,1) là tốc độ bay hơi của pheromone.

* Tiếp theo mỗi con kiến trong đàn sẽ đặt thêm một lượng thông tin pheromone trên những cung mà chúng đã đi qua trong hành trình của chúng.

Trong đó: deta là lượng pheromone mà con kiến k đặt lên cạnh mà nó đã đi qua và được tính như sau:

Với: Ck là độ dài đường đi của con kiến thứ k sau khi hoàn thành đường đi, tức  
là bằng tổng các cung thuộc đường đi mà kiến đã đi qua.

***b. Thuật toán Ant Colony System (ACS)***Phát triển từ thuật toán AS  
❖***Quy tắc di chuyển của kiến***Trong thuật toán ACS, con kiến k đang ở đỉnh i, việc kiến chọn đỉnh j để di  
chuyển đến được xác định bằng quy luật như sau:  
⮚Cho q0 là một hằng số cho trước (0< q0 <1)  
⮚Chọn ngẫu nhiên một giá trị q trong khoảng [0,1]  
⮚Nếu q < q0 kiến k chọn điểm j di chuyển tiếp theo dựa trên giá trị lớn nhất của  
thông tin mùi và thông tin heuristic có trên cạnh tương ứng với công thức:

)

* Nếu q > q0 kiến k sẽ chọn đỉnh j chưa được đi qua trong tập láng giềng của i theo

một quy luật phân bổ xác xuất được xác định theo công thức sau:

, j∈

* **Quy tắc cập nhật thông tin mùi**

Cập nhật thông tin mùi toàn cục:

Một con kiến có đường đi tốt nhất sau mỗi lần lặp thì được phép cập nhật thông

tin pheromone. Việc cập nhật được thực hiện theo công thức sau:

Cập nhật thông tin mùi cục bộ:

Công thức sau:

Với p: là tham số bay hơi năm trong khoảng (0,1)

=1/(nCnm)

n: là số dỉnh hay là số thành phố

Cnm: chiều dài hành trình cho bởi phương pháp tìm kiếm gần nhát

***c. Thuật toán Max–Min Ant System (MMAS)***

Được phát triển bởi Stutzle và Hooss vào năm 1996, được mở rộng lên từ hệ thống AS. Luật di chuyển của kiến được thực hiện tương tự như trong thuật toán ACS.

* **Quy tắc cập nhật thông tin mùi**

Thuật toán MMAS thực hiện việc cập nhật thông tin mùi khi toàn bộ kiến trong

đàn hoàn thành lời giải và lượng thông tin mùi chỉ cập nhật trên các cạnh thuộc lời giải

tối ưu nhất. Ban đầu cũng thực hiện bay hơi thông tin mùi trên các cạnh thuộc lơi giải

tối ưu với một lượng được xác định tại công thức (2.4).

Lượng pheromone trên một cạnh được xác định như sau:

với

Cbest là độ dài đường đi ngắn nhất của con kiến thứ k sau khi cả đàn hoàn thành đường đi.

* Khởi tạo và khởi tạo lại thông tin mùi

Thuật toán MMAS đã thêm vào giá trị cận trên và giá trị cận dưới cho thông tin pheromone gọi là τmin và τmax

Sau mỗi lần cập nhật giá trị thông tin mùi τij nếu τij < τmin thì sẽ gán τij = τmin và nếu τij > τmax thì gán τij = τmax

Giá trị cận trên τmax thường được thiết lập với công thức sau:

τmax = 1 ⁄𝜌𝐶𝑏𝑒𝑠𝑡

Giá trị cận dưới τmin được xác định bằng công thức:

τmin = τmax /2n.

***d. Thuật toán Rank-Based Ant System (RBAS)***

Đây cũng là một thuật toán được mở rộng phát triển từ hệ thống AS đưa ra bởi Bullnheimer, Hartl và Strauss vào năm 1997. Thuật toán này đưa vào ý tưởng xếp hạng cho các lời giải khi thực hiện cập nhật pheromone. Cụ thể như sau:

⮚ Đầu tiên, m con kiến được xếp hạng theo thứ tự giảm dần dựa theo chất lượng  
lời giải mà nó thu được. Ví dụ: (S1, S2, … Sm-1, Sm) trong đó S1 là phương án tốt nhất.

⮚ Pheromone chỉ được đặt thêm trên các cung của б -1 con kiến có lời giải tốt nhất. Lượng pheromone cũng phụ thuộc trực tiếp vào thứ hạng sắp xếp của con kiến.

⮚ Các đoạn đường đi của lời giải tốt nhất được nhận thêm một lượng pheromone  
phụ thuộc vào chất lượng lời giải.

Các công thức như sau:

Trong đó:

Và

Tóm tắt thủ tục cập nhật mùi pheromone của thuật toán này:

1 **Procedure** daemon\_actions

2 **for** **each** Sk **do** local\_search ( Sk ) {optional}

3 rank() in decreasing order of solution

quality into (,...,)

4 **if** (best ())

5  =

6 **end if**

7 **for** μ = 1 to σ = 1 **do**

8 **for** each edge  **do**

9

10 **end for**

11 **end for**

12 **for** each edge **do**

13

14 **end for**

15 **end Procedure**

**e. Thuật toán Best-Worst Ant System (BWAS)**

Thuật toán được đưa ra bởi Cordon vào năm 1999. Thuật toán này bao gồm một thuật toán mở rộng khác của AS là MMAS (về luật di chuyển và việc bay hơi của pheromone). Bên cạnh đó trong thuật toán này còn quan tâm tới của việc tối ưu cục bộ một cách hệ thống để nâng cao chất lượng lời giải của con kiến. Trong thuật toán BWAS **có 3 daemon action** thêm vào gồm có:

* Đầu tiên, áp dụng luật có tên *best-worst pheromone update* để tăng cường pheromone trên các đoạn đường đi qua bởi lời giải tốt nhất toàn cục (global best solution). Thêm vào đó luật này sẽ phạt những cạnh của lời giải tồi nhất trong lần lặp Scurrent-worst.
* Áp dụng *Pheromone trail mutation* để đi theo các hướng khác nhau trong quá trình tìm kiếm.
* BWAS có cơ chế khởi tạo lại thông tin pheromone khi thuật toán bị đình trệ, bằng cách thiết lập pheromone trail cho tất cả các thành phần bằng .

Mô hình thủ tục Daemon action của thuật toán BWAS như sau:

1 **Procedure** daemon\_actions

2 **for each** Sk **do** local\_search ( Sk )

3  = best\_solution ( Sk )

4 **if** (best (,))

5

6 **end if**

**7 for** each edge **do**

8

9

10 **end for**

11

12 = worst\_solution (Sk)

13 **for** each edge and **do**

14

15 **end for**

16 mut = mut (it, )

17 **for** each nút / component r ∈{1,..., l} **do**

18 z = generate\_random\_value\_in\_[0,1]

19 **if** ( z <= Pm)

20 s = generate\_random\_value\_in\_[1,…, 1]

21 a = generate\_random\_value\_in\_[0,1]

22 if ( a = 0)

23 else

24 **end if**

25 **end for**

26 **if** (stagnation\_condition)

27 **for** each **do**

1. **end if**
2. **end Procedure**

# **CHƯƠNG 3. THUẬT TOÁN SEAGULL OPTIMIZATION ALGORITHM**

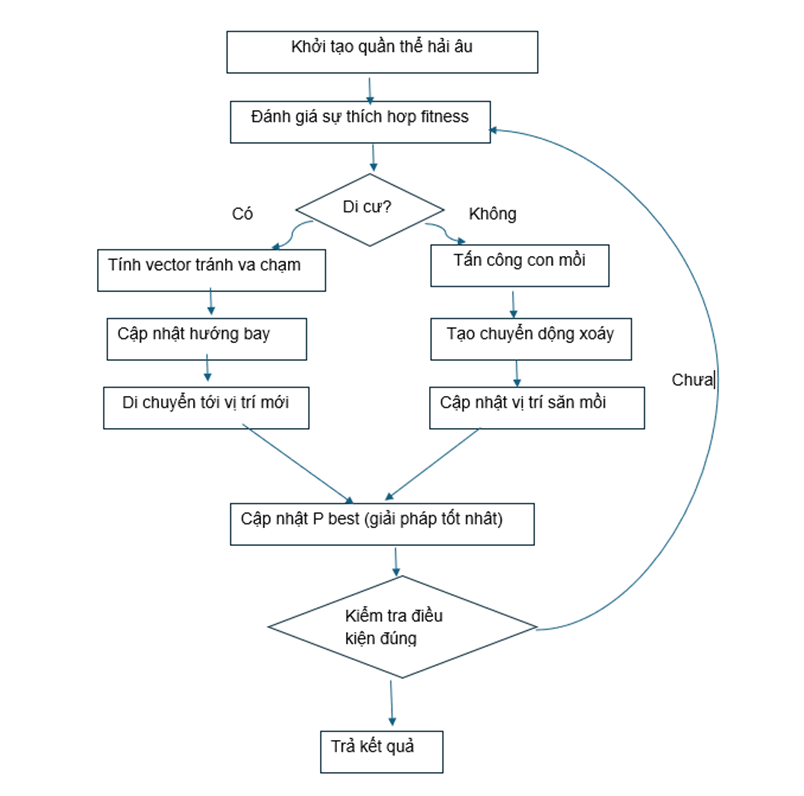
## **3.1. Tổng quan về thuật toán Seagull optimization algorithm**

Trong những năm gần đây, cùng với sự phát triển mạnh mẽ của công nghệ máy tính, các lĩnh vực như máy học và học thuật toán đã trở thành những trụ cột quan trọng trong tiến trình phát triển của khoa học máy tính và các ứng dụng công nghệ hiện đại. Một xu hướng đáng chú ý trong nghiên cứu tối ưu hóa là việc lấy cảm hứng từ các đặc điểm hành vi của động vật và thực vật trong tự nhiên để mô phỏng và phát triển các thuật toán tối ưu hóa trí tuệ bầy đàn (Swarm Intelligence). Trong xu hướng này, thuật toán tối ưu hóa chim mòng biển (Seagull Optimization Algorithm – SOA) được Gaurav Dhiman, Vijay Kumar và cộng sự đề xuất vào năm 2019, đã nhanh chóng thu hút sự quan tâm của cộng đồng nghiên cứu. SOA mô phỏng hành vi di cư và tấn công con mồi của chim mòng biển để thực hiện quá trình tìm kiếm lời giải trong không gian tối ưu. Cụ thể, quá trình khám phá (exploration) được mô phỏng theo hành vi di cư theo đàn của chim mòng biển nhằm bao quát không gian tìm kiếm, trong khi quá trình khai thác (exploitation) mô phỏng cách chim mòng biển tấn công con mồi theo quỹ đạo xoắn ốc nhằm tiến gần đến nghiệm tối ưu.

Thuật toán SOA đã được kiểm chứng hiệu quả thông qua việc so sánh với nhiều thuật toán tối ưu hóa nổi tiếng trên các bộ hàm kiểm tra chuẩn. Và đặc biệt, SOA đã được ứng dụng để giải quyết nhiều bài toán thực tế có tính ràng buộc cao trong công nghiệp, chẳng hạn như tối ưu hóa thiết kế kỹ thuật, phân bổ tài nguyên, lập lịch sản xuất, điều phối robot, cũng như các bài toán trong lĩnh vực xử lý dữ liệu và phân loại. Nhờ khả năng mô hình hóa linh hoạt và tính tổng quát cao, SOA đã chứng minh tiềm năng ứng dụng rộng rãi trong các hệ thống kỹ thuật và công nghệ hiện đại đòi hỏi giải pháp tối ưu với chi phí tính toán hợp lý.

## **3.2. Cấu trúc và nguyên lý hoạt động của thuật toán**

### ***3.2.1. Sơ đồ thuật toán***



*Hình 3.2.1. Sơ đồ thuật toán Seagull optimization algorithm*

**\* Thuật toán:**

**Input**: Seagull population

**Output**: Optimal search agent

* 1. **procedure** SOA
  2. Initialize the parameters A, B, and Maxiteration
  3. Set fc ← 2
  4. Set u ← 1
  5. Set v ← 1
  6. **while** (x < Maxiteration) **do**
  7. ← **ComputeFitness** (  ) /\* Calculate the fitness values of each search agent using **ComputeFitness** function\*/

/\* Migration behavior \*/

* 1. rd ← Rand(0, 1) /\* To generate the random number in range [0, 1] \*/
  2. k ← Rand(0, 2π) /\* To generate the random number in range [0, 2π] \*/

/\* Attacking behavior \*/

* 1. r ← u × ekv /\* To generate the spiral behavior during migration \*/
  2. Calculate the distance using Eq. (9)
  3. P ← x′ × y′ × z′ /\* Compute x, y, z planes using Eqs. (10) - (13) \*/

* 1. t ← t + 1
  2. **end while**
  3. return
  4. **end procedure**

1. **procedure** ComputeFitness ()
2. **for** i ← 1 to n **do** /\* Here, n represents the dimension of a given problem \*/
3. FITs[i] ← FitnessFunction() /\* Calculate the fitness of each individual \*/
4. **end for**
5. ← **BEST**(FITs[]) /\* Calculate the best fitness value using **BEST** function \*/
6. return
7. **end procedure**
8. **procedure** BEST(FITs[])
9. Best ← FITs[0]
10. **for** i ← 1 to n **do**
11. **if** (FITs[i] < Best) **then**
12. Best ← FITs[i]
13. **end if**
14. **end for**
15. return Best /\* Return the best fitness value \*/
16. **end procedure**

**\* Đánh giá độ phức tạp của thuật toán:**

*- Các ký hiệu:*

• P\_s: quần thể chim hải âu, số lượng cá thể là N

• n: số chiều của mỗi cá thể (độ dài vector nghiệm)

• MaxIteration: số vòng lặp tối đa

• FitnessFunction: hàm đánh giá chất lượng nghiệmật toán theo từng phần:

*- Vòng lặp chính:* while (x < MaxIteration)

• Gọi I là số vòng lặp (tối đa là MaxIteration)

*- Hàm ComputeFitness(P\_s)*

• Duyệt qua toàn bộ quần thể (kích thước N) → với mỗi cá thể, tính FitnessFunction

Nếu mỗi hàm fitness chạy trong O(n) (vì cá thể có n chiều):

→ O(N · n)

*- Tính Best(FITs[]):* duyệt qua mảng FITs[] có N phần tử → O(N)

*- Hành vi di cư và tấn công*

• Gồm nhiều phép toán vector: nhân, cộng, … → mỗi phép với vector chiều n

• Các phép này thực hiện cho toàn bộ quần thể: O(N · n)

***Vậy:***

Tổng thời gian cho mỗi vòng lặp: O(N · n) + O(N) + O(N · n) = O(N · n)

Tổng thời gian toàn thuật toán (qua I vòng lặp): O(I · N · n)

### ***3.2.2. Mô hình thuật toán***

Mòng biển là một nhóm động vật rất thông minh tồn tại trên Trái đất và hiện phân bố trên toàn cầu với nhiều loài và kích thước khác nhau. Mòng biển là loài chim di cư theo mùa, di cư theo những thay đổi khí hậu theo mùa để có đủ nguồn thức ăn. Ý tưởng chính của thuật toán tối ưu hóa mòng biển là mô phỏng hành vi di cư và hung dữ của chính loài mòng biển và tìm ra giải pháp tối ưu bằng cách liên tục cập nhật vị trí của mòng biển.

***a. Đặc điểm sinh học***

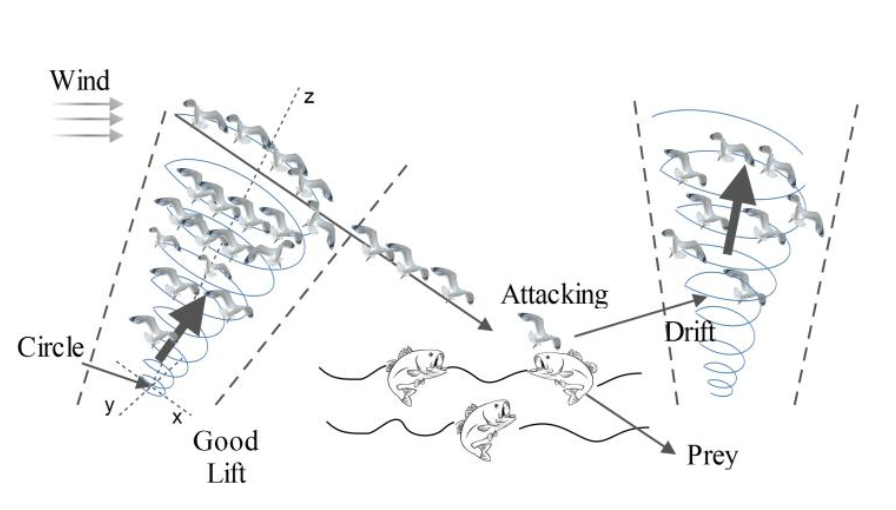
Chim mòng biển (tên khoa học: *Liridae*) là loài chim biển phổ biến, có mặt ở hầu hết các thành phố ven biển trên toàn thế giới. Chúng sống theo bầy đàn và nổi bật với trí thông minh đặc biệt trong việc tìm kiếm và săn mồi. Mòng biển là loài ăn tạp, có thể ăn nhiều loại thức ăn như côn trùng, cá, bò sát, lưỡng cư, ... và đặc biệt có khả năng uống cả nước ngọt lẫn nước mặn nhờ hệ thống tuyến đặc biệt phía trên mắt giúp bài tiết muối.

Hai hành vi đặc trưng nổi bật của chim mòng biển là di cư và tấn công. Hành vi di cư thể hiện qua việc chúng bay theo đàn trong các đợt di chuyển xa để thích nghi với điều kiện khí hậu và tìm kiếm nguồn thức ăn dồi dào. Trong quá trình này, mỗi cá thể giữ vị trí khác nhau để tránh va chạm, đồng thời có xu hướng di chuyển về phía vị trí tối ưu hơn trong đàn. Hành vi tấn công, thường xảy ra trong quá trình tìm mồi, thể hiện sự hung hăng và quyết liệt. Đáng chú ý, khi tấn công, mòng biển thường bay theo hình xoắn ốc – một kiểu bay đặc trưng, giúp cả đàn phối hợp nhịp nhàng và hiệu quả trong việc tiếp cận mục tiêu.

Nói chung mòng biển sống theo bầy đàn. Chúng sử dụng trí thông minh để tìm và tấn công con mồi. Điều quan trọng nhất về loài mòng biển là hành vi di cư và tấn công của chúng. Di cư được định nghĩa như sự di chuyển theo mùa của loài mòng biển từ nơi này đến nơi khác tìm nguồn thục phẩm phong phú và dồi dào nhất sẽ cung cấp năng lượng. Hành vi này được mô tả như sau:

* Trong quá trình di cư, chúng di chuyển theo nhóm. Các vị trí ban đầu của mòng biển khác nhau để tránh va chạm giữa mỗi con khác.
* Trong một nhóm, mòng biển có thể bay theo hướng của con mòng biển có sức sống tốt nhất, tức là con mòng biển có giá trị sức sống thấp hơn so với những con khác.
* Dựa trên con mòng biển khỏe mạnh nhất, những con mòng biển khác có thể cập nhật vị trí ban đầu của chúng.

Mòng biển thường tấn công các loài chim di cư trên biển khi chúng di cư từ nơi này đến nơi khác. Chúng có thể tạo ra hình xoắn ốc chuyển dạng tự nhiên trong khi tấn công. Hành vi này được minh họa trong hình 3.2.2.1. Những hành vi này có thể là được xây dựng theo cách mà nó có thể liên kết với các mục tiêu chức năng được tối ưu hóa. Điều này làm cho nó có thể xây dựng một thuật toán tối ưu hóa.



*Hình 3.2.2.1. Hành vi di cư và tấn công của mòng biển.*

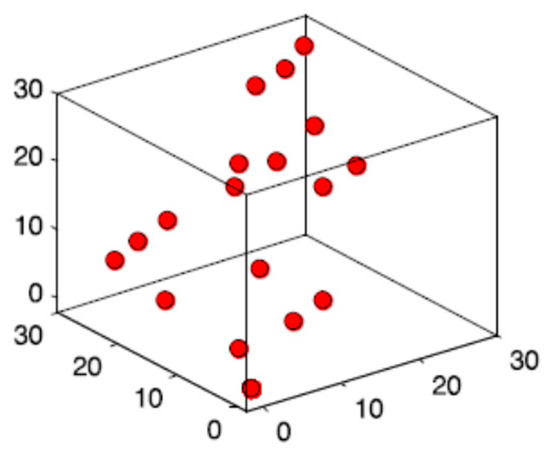
***b. Mô hình toán sinh học***

Thông qua các đặc điểm sinh học của quần thể mòng biển, một cá thể mòng biển được chọn ngẫu nhiên và hành vi di cư cũng như tính hung dữ của nó được chỉ định, mô tả và thực hiện bằng mô hình toán học sau.

***\* Hành vi di cư***

Thuật toán mô phỏng cách quần thể mòng biển di chuyển từ vị trí này sang vị trí khác trong quá trình di cư. Trong giai đoạn này, từng con mòng biển phải đáp ứng ba điều kiện:

1. Tránh va chạm: Để tránh va chạm với những con mòng biển lân cận (những con mòng biển khác), thuật toán tính toán vị trí mới của một con mòng biển riêng lẻ bằng cách sử dụng một biến bổ sung 𝐴. Biến số 𝐴 là sự kiểm soát để tránh va chạm giữa những con mòng biển ở gần nhau. (Hình 3.2.2.2)



Hình 3.2.2.2. Tránh va chạm giữa các con mòng biển

(1)

(2)

Trong đó:

: chỉ ra vị trí mới mà mòng biển thu được khi di chuyển và vị trí tìm kiếm này sẽ không xung đột với các vị trí khác của mòng biển.

: chỉ ra vị trí hiện tại của mòng biển.

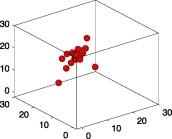
: biểu thị số lần lặp lại của thuật toán tại thời điểm hiện tại ​​​​

Max\_iteration: thể hiện số lần lặp lại lớn nhất của thuật toán đang hoạt động.

𝐴: là một biến bổ sung được xác định để tránh va chạm với những con chim biển lân cận (những con mòng biển khác).

: là một siêu tham số được xác định bởi thuật toán và giá trị của tham số này được cố định là 2, do đó giá trị của 𝐴 giảm tuyến tính từ 2 đến 0 khi số lần lặp lại được chồng lên nhau.

1. Di chuyển theo hướng của người hàng xóm tốt nhất: Để tránh chồng chéo với những con mòng biển khác, từng con mòng biển sẽ di chuyển theo hướng của người hàng xóm tốt nhất của chúng.



*Hình 3.2.2.3. Sự di chuyển về phía hàng xóm tốt nhất.*

Trong đó:

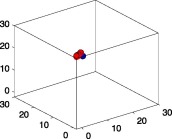
: biểu thị hướng đi tốt nhất của mòng biển

: biểu thị hướng vị trí của người hàng xóm tốt nhất (tức là con mòng biển khoẻ nhất).

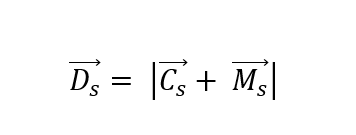
Trên là một số ngẫu nhiên trong phạm vi [0,1], **𝑟𝑑** có thể làm gián đoạn việc thực hiện thuật toán bằng cách thực hiện các bản cập nhật ngẫu nhiên của riêng nó trong phạm vi giá trị đã chọn, ngăn thuật toán rơi vào trạng thái tối ưu cục bộ trong quá trình thực hiện.

B là một số ngẫu nhiên khác dựa trên giá trị của 𝐴 và giá trị của B được sử dụng để cân nhắc tìm kiếm toàn cục hoặc cục bộ của thuật toán.

1. Tiếp cận vị trí tốt nhất của mòng biển: Sau khi di chuyển theo hướng của người hàng xóm tốt nhất, từng con mòng biển sẽ di chuyển theo hướng tối ưu toàn cầu và cuối cùng đến một vị trí mới.



*Hình 3.2.2.4. Sự hội tụ về phía tốt nhất*



Trong đó: kết hợp các điều kiện để mòng biển tránh va chạm và di chuyển đến vị trí tối ưu, có thể được biểu thị bằng một vị trí mới khác mà mòng biển đạt được, cũng như khoảng cách giữa mòng biển hiện tại và mòng biển tối ưu.

***\* Hành vi tấn công***

Mòng biển là một lớp sinh vật rất thông minh, chúng rất giỏi sử dụng lịch sử và kinh nghiệm của mình trong quá trình tìm kiếm. Trong quá trình di cư, do phải săn mồi trong thời gian dài và thường thực hiện hành vi tấn công, khi mòng biển tấn công con mồi, đầu tiên chúng sẽ dựa vào đôi cánh và trọng lượng của mình để duy trì độ ổn định cao, sau đó là hành vi chuyển động xoắn ốc cụ thể khi bay, liên tục thay đổi tốc độ bay và góc tấn công. Khi một con mòng biển tấn công, chuyển động của nó trong mặt phẳng ba chiều 𝑥 , 𝑦 , 𝑧 được mô tả như sau.

Trong đó:

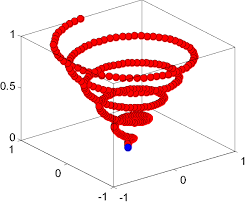
𝑟: chỉ bán kính bay của đường bay xoắn ốc khi mòng biển thực hiện hành vi tấn công.

𝑘 là một số ngẫu nhiên trong khoảng [ 0 ≤ k ≤ 2𝜋].

𝑢 và 𝑣 là các hằng số xác định hình dạng của đường xoắn ốc, thường xác định giá trị của cả hai là 1

𝑒 được biểu thị là cơ số của logarit tự nhiên.

là một biểu thức được cập nhật dựa trên sự kết hợp giữa hành vi di cư và tấn công để xác định vị trí tìm kiếm cuối cùng của mòng biển.



*Hình 3.2.2.5. Hành vi tấn công tự nhiên của chim mòng biển*

### ***3.2.3. Nguyên lý hoạt động thuật toán***

***Quy luật di chuyển và tấn công của chim hải âu:***

* Chim hải âu thường sống theo bầy và di chuyển cùng nhau đến nơi có nhiều thức ăn hơn, đặc biệt trong mùa di cư.
* Khi bay, mỗi con sẽ tự điều chỉnh vị trí để tránh đụng nhau và cố gắng bay gần con có hướng bay tốt nhất trong đàn.
* Khi phát hiện con mồi, cả đàn sẽ bay xoắn ốc để tiếp cận và tấn công – đây là cách giúp chúng nhanh chóng và chính xác bắt được mồi.
* Trong thuật toán, mỗi con chim là một lời giải. Chúng sẽ "bay" (di chuyển) trong không gian tìm kiếm để tìm lời giải tốt nhất.

***Cách mỗi cá thể chim hải âu di chuyển và tìm lời giải:***

* Mỗi con sẽ cập nhật vị trí của mình dựa theo hướng của con chim tốt nhất hiện tại.
* Thuật toán còn sử dụng thêm các yếu tố ngẫu nhiên (rd ← Rand (0, 1) và k ← Rand (0, 2π)) để tăng tính đa dạng cho quần thể và tránh chim bị “kẹt” tại vị trí chưa tối ưu
* Khi gần lời giải tốt, chim sẽ chuyển sang hành vi tấn công xoắn ốc để tinh chỉnh vị trí của mình tốt hơn

=> Mỗi cá thể sẽ cập nhật vị trí trong không gian tìm kiếm cho đến khi đạt đến điều kiện dừng (hết số vòng lặp hoặc hội tụ).

* Một vị trí tốt được xem là một lời giải cho bài toán tối ưu.
* Tất cả các cá thể trong quần thể sau mỗi vòng lặp đều được đánh giá để xác định lời giải tốt nhất.

### ***3.2.4. Các thuật toán khác phát triển trên mô hình SOA***

Thuật toán Chim Hải Âu (Seagull Optimization Algorithm – SOA) đã được nhiều nhà nghiên cứu cải tiến và phát triển thành các biến thể nhằm nâng cao hiệu quả tối ưu. Một số biến thể phổ biến có thể kể đến như:

- SOA cải tiến (Improved SOA – ISOA), trong đó các tham số được điều chỉnh linh hoạt theo từng vòng lặp để cải thiện khả năng hội tụ và tránh bị mắc kẹt tại các điểm cực trị cục bộ.

- Ngoài ra, còn có các phiên bản lai ghép như SOA kết hợp với PSO (Particle Swarm Optimization) để tận dụng khả năng khai thác mạnh của PSO, hay kết hợp với GA (Genetic Algorithm) và DE (Differential Evolution) nhằm tăng tính đa dạng và độ sâu tìm kiếm.

- Một hướng phát triển khác là SOA thích nghi (Adaptive SOA), cho phép các tham số như tốc độ hoặc khoảng cách di chuyển thay đổi linh hoạt để cân bằng giữa quá trình khai phá và khai thác.

- Với các bài toán tối ưu đa mục tiêu, thuật toán SOA cũng được mở rộng thành MO-SOA (Multi-objective SOA) bằng cách áp dụng các chiến lược như lưu trữ lời giải ưu việt và xếp hạng Pareto.

Những biến thể này giúp thuật toán SOA mở rộng phạm vi ứng dụng và giải quyết hiệu quả hơn các bài toán phức tạp trong thực tế.

# **CHƯƠNG 4. MÔ PHỎNG THUẬT TOÁN**

## **4.1. Thuật toán Ant colony optimization**

### ***4.1.1. Bài toán người du lịch***

* ***Phát biểu bài toán***

Một người du lịch đi thăm n thành phố T1…Tn. Xuất phát từ thành phố nào đó, người du lịch muốn đi thăm tất cả các thành phố còn lại, mỗi thành phố đi đúng một lần rồi trở lại thành phố xuất phát. Gọi Cij là chi phí từ Ti đến Tj. Tìm hành trình với tổng chi phí nhỏ nhất.

* ***Tư tưởng giải quyết bài toán người du lịch bằng thuật toán kiến***
* Kiến đi sau sử dụng vết mùi lạ (lớp Trail) và kinh nghiệm (lớp Heuristics) của kiến đi trước để tìm kiếm con đường đi cho mình, sau đó con đường tốt hơn sẽ được cập nhật lại.
* Cứ như vậy sẽ cho ta kết quả con đường đi ngắn nhất cần tìm.***•***
* ***Thuật toán đàn kiến giải bài toán người du lịch***

Áp dụng thuật toán kiến vào bài toán người du lịch với các thông số  
của kiến như sau:

m: số lượng kiến

τij: nồng độ vết mùi trên cạnh (i,j)

*p* : là tham số bay hơi nằm trong khoảng (0,1)  
pijk : xác suất con kiến k lựa chọn cạnh (i,j)  
α: hệ số điều chỉnh ảnh hưởng của τij  
ηij : thông tin heuristic giúp đánh giá chính xác sự lựa chọn của kiến  
đi từ đỉnh i tới đỉnh j  
β : hệ số điều chỉnh ảnh hưởng ηij  
Nik : tập các đinh láng giềng của i mà con kiến k chưa đi qua  
∆τijk là lượng pheromone mà con kiến k đặt lên cạnh mà nó đã đi qua  
NC: là số bước lặp của thuật toán  
Sk: đường đi của kiến k (tập các đỉnh mà kiến k đi qua)  
q : là giá trị ngẫu nhiên trong khoảng [0,1]  
aki : nhận giá trị True, False tương ứng với con kiến k đã thăm hoặc chưa đi qua đỉnh i

**\*Các Bước xây dựng thuật toán như sau:**

* Đầu vào:

Đồ thị G=(V,E), với tập đỉnh V và tập cạnh E, ma trận trọng số D=d[i,j] với i, j∈ 𝑉

Số lượng kiến m.

* Đầu ra: đường đi S với khoảng cách ngắn nhất Cs
* Các bước:

***Bước 1: Khởi tạo***

Khởi tạo các tham số NC, β,α,𝜌, số lượng kiến m và số đỉnh n.

Khởi tạo độ dài đường đi ngắn nhất Cbest là hằng số

Tình độ dài đường đi ngắn nhắt Cnn

Tính giá trị 𝜏max=1/𝜌Cnn và 𝜏min = 𝜏max/2

Khởi tạo giá trị q0 với (0 ≤ q0 ≤ 1)

Khởi gắn điều kiện kết thức stop :=1

Thiết lập ma trận pheromone trên tất cả các cạnh

For i:=1 to n do

For j:=1 to n do 𝜏ij = 𝜏max

***Bước 2: Xây dựng đường đi của kiến***Trường hợp hoàn thành xong tất cả các bước lặp: stop > NC thì xuất ra đường đi ngắn nhất và kết thúc

*2.1. Thiết lập các đỉnh khi kiến chưa đi qua nhận giá trị false*For k:=1 to m do  
For i:=1 to n do aki:=false  
Thiết lập đường đi của kiến Sk =∅

*2.2 Chọn ngẫu nhiên vị trí xuất phát của kiến*For k:=1 to m do  
For i:=1 to n do  
R🡨random{1..n}  
Bổ sung r vào đường đi Sk:={r}, akr:=true;  
Gán độ dài đường đi Ck:=0  
*2.3 Xác định đỉnh đến tiếp theo của kiến k*- Chọn ngẫu nhiên một giá trị q: q🡨random{0..1}  
- Nếu q ≤ q0 kiến k chọn điểm u di chuyển tiếp theo với

u =max(𝜏rl ( ηrl )β )  
- Nếu q > q0 kiến k sẽ chọn đỉnh u chưa được đi qua trong tập láng giềng của r theo công thức sau:

pruk =   
- Chọn đỉnh u là đỉnh tiếp theo, bổ sung đỉnh u vào Sk

Sk:={r,u}  
- Tăng độ dài đường đi Ck:=Ck+dru  
- Gán :=true  
***Bước 3: Xác định đường đi ngắn nhất***Ta có Ck là độ dài đường đi của con kiến k với k = [1..m] thu được từ bước 2.

Nếu Ck < Cbest thì hiệu chỉnh Cbest:=Ck  
***Bước 4: Cập nhật thông tin mùi***Tại bước này, chỉ cập nhật thông tin mùi trên đường đi của con kiến k  
có giá trị Ck nhỏ nhất thu được từ bước 3, tức là giá trị Cbest  
*4.1 Bay hơi thông tin mùi trên các cạnh* For i:=1 to n do

For j:=i to n do 𝜏ij:=(1-𝜌) 𝜏ij  
*4.2 Thêm thông tin mùi trên các cạnh thuộc đường đi tốt nhất Sk,best* Tính giá trị pheromone ∆𝜏=

Nếu cạnh (i,j) ∈ Skbest thì 𝜏ij:= 𝜏ij +∆𝜏

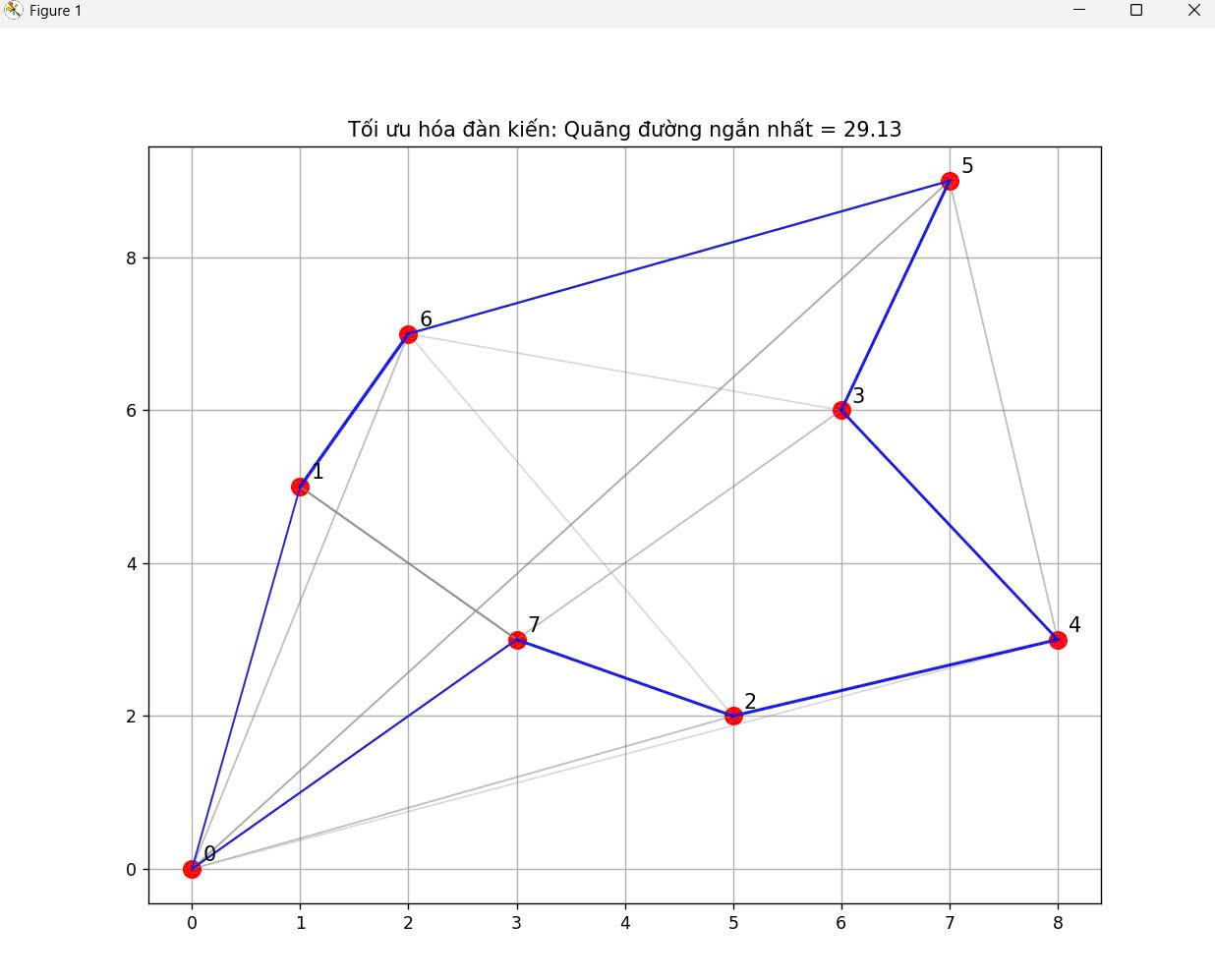
Kiểm tra thông tin pheromone với cận trên và cận dưới

Nếu 𝜏ij< 𝜏min thì 𝜏ij = 𝜏min

Nếu 𝜏ij > 𝜏max thì 𝜏ij= 𝜏max

Tăng giá trị stop: = stop + 1.

### ***4.1.2. Mô phỏng thuật toán***



## **4.2. Thuật toán Seagull optimization algorithm**

### ***4.2.1. Bài toán người giao hàng***

***\* Phát biểu bài toán***

Một công ty giao hàng có một kho chính và cần giao hàng đến 10 địa điểm trong thành phố. Mỗi địa điểm chỉ được thăm một lần duy nhất và tuyến đường phải bắt đầu và kết thúc tại kho.

**Mục tiêu:** Tìm hành trình tối ưu (ngắn nhất) bắt đầu và kết thúc tại kho, đi qua tất cả các điểm còn lại đúng một lần.

***\* Tư tưởng giải quyết bài toán người giao hàng bằng thuật toán SOA:***

* Mỗi cá thể (một con mòng biển) đại diện cho một hoán vị của các địa điểm giao hàng (kể cả kho).
* SOA sử dụng hai giai đoạn chính:

+ Di cư: điều chỉnh cá thể gần hơn với lời giải tốt.

+ Tấn công mồi: khai thác lời giải tốt hơn bằng các phép hoán vị.

Việc tối ưu hóa diễn ra qua nhiều vòng lặp để tìm ra tuyến đường ngắn nhất.

**\* Các bước xây dựng thuật toán:**

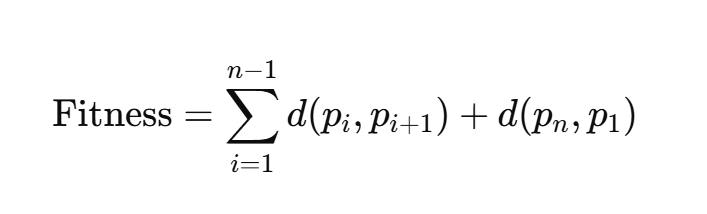
**Bước 1: Biểu diễn lời giải**

Mỗi cá thể (con mòng biển) được biểu diễn bằng một hoán vị của các điểm cần thăm, trong đó:

* + Vị trí đầu tiên là Kho (chỉ số 0) – cố định.
  + Phần còn lại là một hoán vị ngẫu nhiên của các điểm còn lại (A đến J).
  + Tuyến đường là vòng khép kín: xuất phát từ kho, đi qua tất cả các điểm, và quay về kho.

**Bước 2: Xây dựng hàm mục tiêu (fitness function)**

- Đánh giá chất lượng một tuyến đường bằng **tổng chiều dài quãng đường** (dựa vào ma trận khoảng cách Euclidean):



Trong đó ​ là chỉ số của điểm thứ i trong tuyến đường.

**Bước 3: Khởi tạo quần thể ban đầu**

Sinh ngẫu nhiên N cá thể (lời giải ban đầu):

* + Mỗi cá thể là một hoán vị của các điểm (trừ kho), thêm kho vào đầu.
  + Đảm bảo không có cá thể trùng nhau trong lần khởi tạo.

**Bước 4: Cập nhật lời giải (hành vi mòng biển)**

* + Hành vi di cư → sinh lời giải mới bằng các phép hoán đổi, đảo ngược, chèn trong hoán vị.
  + Hành vi tấn công mồi → tiếp tục cải tiến lời giải tốt hơn qua biến đổi cục bộ.

Thực hiện cập nhật lời giải:

* Nếu lời giải mới tốt hơn (fitness nhỏ hơn) → chấp nhận.
* Nếu không → giữ nguyên cá thể cũ.

**Bước 5: Lựa chọn lời giải tốt nhất**

Sau mỗi vòng lặp:

* + Cập nhật cá thể tốt nhất hiện tại.
  + So sánh với lời giải toàn cục tốt nhất và cập nhật lại nếu cần.

**Bước 6: Điều kiện dừng**

Thuật toán dừng khi:

Đạt đến số vòng lặp tối đa (MaxIter), hoặc lời giải hội tụ (không thay đổi sau nhiều vòng).

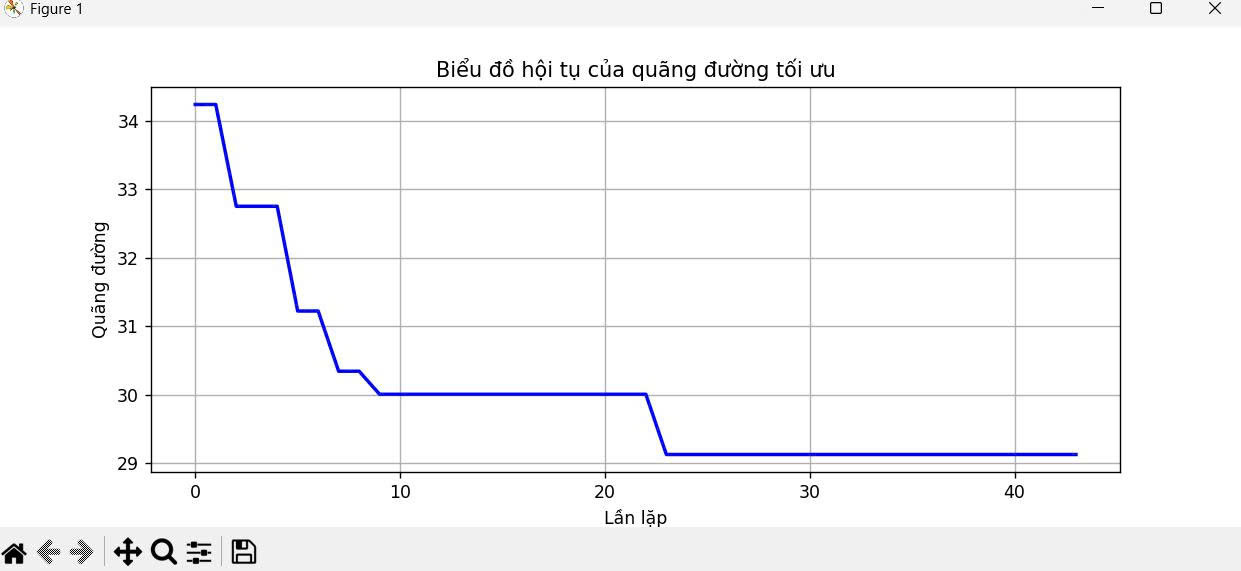
**Bước 7: Xuất kết quả**

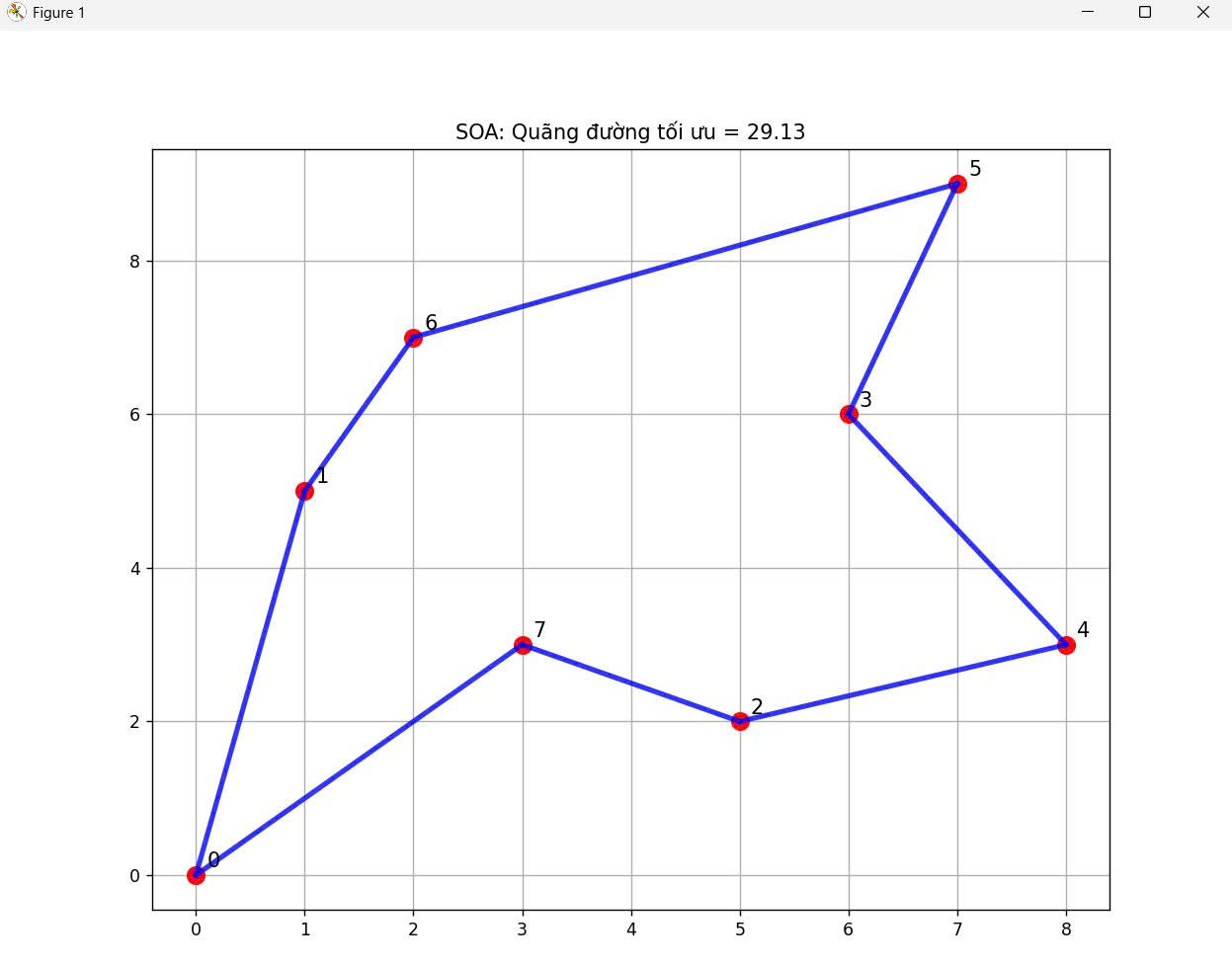
In ra:

* + Tuyến đường tối ưu (theo tên điểm).
  + Tổng quãng đường.

Vẽ sơ đồ trực quan hóa tuyến đường.

### ***4.2.2. Mô phỏng thuật toán***





# **KẾT LUẬN**

Trong quá trình tìm hiểu, ta thấy rằng cả hai thuật toán ACO (Ant Colony Optimization) và SOA (Seagull Optimization Algorithm) đều là những thuật toán mô phỏng hành vi của sinh vật trong tự nhiên, nhưng mỗi thuật toán lại mang một cách tiếp cận riêng để giải quyết các bài toán tối ưu hóa.

ACO được xây dựng dựa trên hành vi kiếm ăn của đàn kiến trong tự nhiên. Bằng cách sử dụng dấu vết pheromone để chỉ đường, thuật toán này cho phép nhiều “kiến ảo” cùng tìm kiếm và học hỏi từ lẫn nhau để dần tìm ra con đường ngắn nhất hoặc phương án tối ưu. Nhờ tính linh hoạt và khả năng xử lý tốt các bài toán tổ hợp, ACO đã được áp dụng hiệu quả trong các bài toán như tìm đường đi ngắn nhất, lập lịch công việc, tối ưu hóa mạng và các hệ thống phân phối.

SOA lại lấy cảm hứng từ hành vi di cư và tấn công con mồi của chim mòng biển. Với hai giai đoạn mô phỏng thực tế là “di chuyển theo nhóm” và “tấn công xoắn ốc”, SOA cho phép cá thể (đại diện cho lời giải) di chuyển linh hoạt trong không gian tìm kiếm để dần tiến gần đến nghiệm tối ưu. Thuật toán này thường được sử dụng trong các bài toán tối ưu hóa liên tục, chọn lựa đặc trưng, điều chỉnh tham số trong mô hình hoặc điều khiển hệ thống.

Từ việc nghiên cứu hai thuật toán này, ta có thể thấy rõ sức mạnh của tự nhiên trong việc truyền cảm hứng cho các phương pháp tính toán hiện đại. Việc mô phỏng các hành vi sinh học không chỉ mang lại những thuật toán hiệu quả mà còn mở ra hướng đi mới cho việc giải quyết các vấn đề phức tạp trong thực tiễn. Trong tương lai, những thuật toán như ACO và SOA có thể được cải tiến thêm hoặc kết hợp với các kỹ thuật khác để phục vụ tốt hơn cho nhu cầu thực tế đang ngày càng đa dạng và thay đổi nhanh chóng.