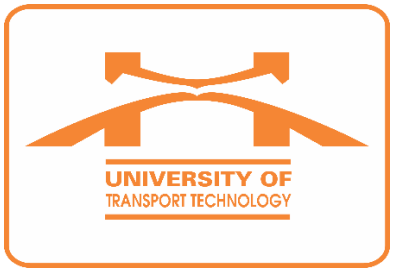
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ GIAO THÔNG VẬN TẢI

KHOA CÔNG NGHỆ THÔNG TIN

---o0o---



TRÍ TUỆ NHÂN TẠO

ĐỀ TÀI: THUẬT TOÁN ACO
(ANT COLONY OPTIMIZATION)

Giảng viên hướng dẫn: Ths.Đỗ Bảo Sơn

Lớp: 73DCTT24

Sinh viên thực hiện: Chu Đình Tạo

Bùi Hữu Trung

Nguyễn Văn Phú

Nguyễn Minh Nghĩa

Nguyễn Quang Trường

Hà Nội, 2024

MỤC LỤC

CHƯƠNG 1: GIỚI THIỆU CHUNG VỀ ACO (ANT COLONY OPTIMIZATION)

1.1. Tổng quan về ACO

Tối ưu hóa đàn kiến (ACO) nghiên cứu các hệ thống nhân tạo lấy cảm hứng từ hành vi của đàn kiến thực và được sử dụng để giải quyết các vấn đề tối ưu hóa rời rạc." Được giới thiệu lần đầu tiên bởi Marco Dorigo vào năm 1991. Ban đầu được áp dụng cho Bài toán Người bán hàng du lịch. Hành vi tự nhiên của loài kiến đã truyền cảm hứng cho các nhà khoa học bắt chước các phương pháp hoạt động của côn trùng để giải quyết các vấn đề tối ưu hóa phức tạp trong đời thực. Bằng cách quan sát hành vi của kiến, các nhà khoa học đã bắt đầu hiểu được phương tiện giao tiếp của chúng. Các mô hình hành vi dựa trên kiến để giải quyết các vấn đề tổ hợp - lần đầu tiên được đề xuất bởi Marco Dorigo. Kiến tiết ra pheromone khi di chuyển từ tổ đến thức ăn và ngược lại để liên lạc với nhau nhằm tìm ra con đường ngắn nhất

1.1.1. Cơ chế hoạt động của kiến tự nhiên:

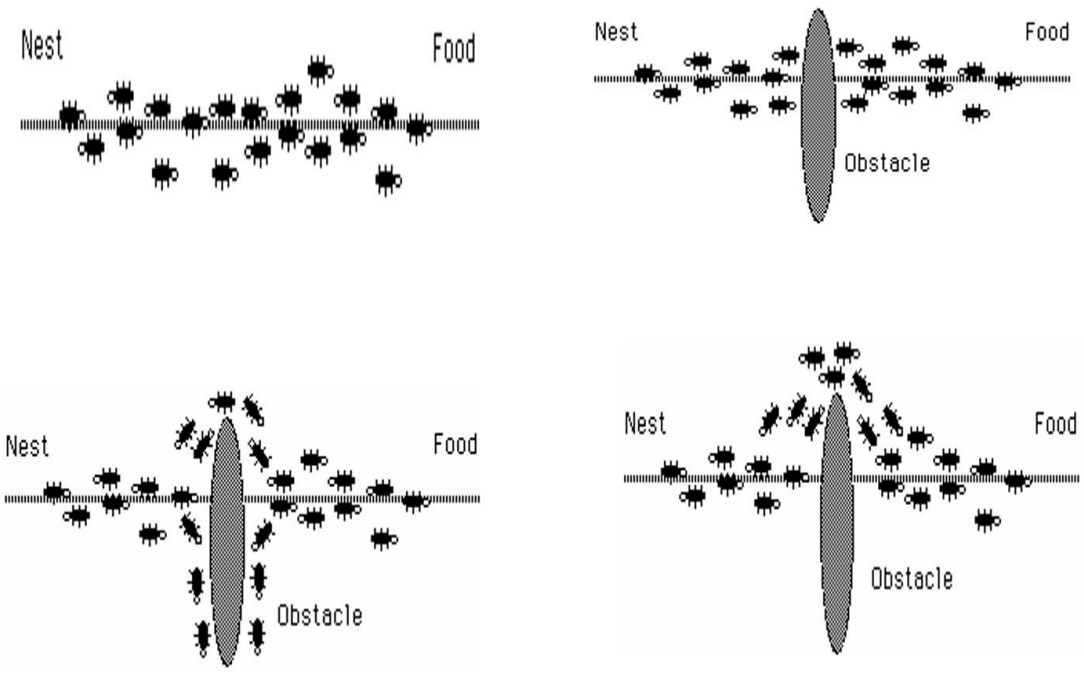
Hành vi của loài kiến trong tự nhiên, đặc biệt là cách chúng tìm kiếm thức ăn, đã truyền cảm hứng sâu sắc cho sự phát triển của thuật toán ACO. Các nghiên cứu chi tiết về hành vi này, điển hình là thí nghiệm của Pierre-Paul Grassé và Deneubourg cùng cộng sự, đã giải thích các cơ chế tự tổ chức hiệu quả của kiến.

*** Cơ chế tìm kiếm thức ăn của loài kiến**

- Kiến sử dụng **pheromone** – một loại hóa chất được tiết ra trên mặt đất – để đánh dấu con đường giữa tổ và nguồn thức ăn.Các con kiến khác, khi di chuyển, sẽ cảm nhận nồng độ pheromone và có xu hướng đi theo đường có lượng pheromone cao hơn.
- Lúc đầu, các con kiến di chuyển một cách ngẫu nhiên từ tổ để tìm nguồn thức ăn.
- Khi một con kiến tìm thấy thức ăn, nó sẽ quay về tổ và tiết ra pheromone trên đường đi.Các con kiến khác bị thu hút bởi dấu hiệu pheromone này và bắt đầu sử dụng đường được đánh dấu.

- Khi nhiều kiến sử dụng cùng một con đường, lượng pheromone tích tụ càng nhiều, làm cho đường đó trở nên hấp dẫn hơn.Dần dần, cả đàn kiến sẽ hội tụ vào con đường ngắn nhất, giúp tối ưu hóa hành trình vận chuyển thức ăn.

ví dụ :



Kiến buộc phải quyết định xem chúng nên đi sang trái hay phải và lựa chọn được đưa ra là một quyết định ngẫu nhiên. Sự tích lũy pheromone nhanh hơn trên con đường ngắn hơn. Sự khác biệt về hàm lượng pheromone giữa hai đường đi theo thời gian khiến đàn kiến chọn đường đi ngắn hơn. Càng có nhiều kiến đi theo một dấu vết thì dấu vết đó càng trở nên hấp dẫn để được theo dõi. Các vấn đề tối ưu hóa khác nhau đã được khám phá bằng cách sử dụng mô phỏng hành vi của con kiến thực này. Con kiến k ở nút r sẽ chọn nút đích s ở giai đoạn sau với xác suất

1.1.2. Thuật toán ACO:

Thuật toán ACO là một phương pháp tối ưu hóa metaheuristic được lấy cảm hứng từ hành vi tự nhiên của loài kiến trong việc tìm kiếm đường đi ngắn nhất giữa tổ và nguồn thức ăn. Thuật toán này được đề xuất lần đầu tiên bởi Marco Dorigo vào năm

1991 để giải quyết bài toán người bán hàng (TSP). Từ đó, nó đã phát triển thành một trong những phương pháp mạnh mẽ nhất cho các bài toán tối ưu hóa tổ hợp.

1.2. Lịch sử và phát triển của ACO

Thuật toán Ant Colony Optimization (ACO) được phát triển từ ý tưởng dựa trên hành vi tự nhiên của loài kiến trong việc tìm kiếm và tối ưu hóa đường đi. Từ khi ra đời, ACO đã trải qua nhiều giai đoạn phát triển và trở thành một trong những thuật toán metaheuristic nổi bật nhất cho các bài toán tối ưu hóa tổ hợp.

1.2.1. Khởi nguồn cảm hứng từ sinh học

Ý tưởng về việc mô phỏng hành vi của loài kiến bắt nguồn từ các nghiên cứu của Pierre-Paul Grasse vào thập niên 1950. Ông là người đầu tiên giới thiệu khái niệm stigmergy, mô tả cách các côn trùng xã hội như kiến và mối sử dụng dấu hiệu môi trường (pheromone) để phối hợp hành vi tập thể.

Nghiên cứu cụ thể về loài kiến: Các thí nghiệm của Jean-Louis Deneubourg và cộng sự (thập niên 1980-1990) đã minh họa cơ chế tìm kiếm thức ăn của loài kiến, đặc biệt là khả năng tự tổ chức và tìm đường ngắn nhất thông qua phản hồi tích cực từ pheromone.

1.2.2. Phát triển thuật toán ACO

Marco Dorigo và hệ thống kiến đầu tiên (Ant System)

- Năm 1991, Marco Dorigo lần đầu tiên giới thiệu Hệ thống kiến (Ant System - AS) trong luận án tiến sĩ của ông tại Đại học Politecnico di Milano, Ý. Thuật toán này được thiết kế để giải quyết bài toán người bán hàng (TSP) – một bài toán NP-hard nổi tiếng.

Đặc điểm nổi bật của Ant System:

- Các kiến nhân tạo sử dụng pheromone ảo để đánh dấu giải pháp.
- Phản hồi tích cực giúp cải thiện giải pháp tốt qua từng vòng lặp.
- Sử dụng thông tin heuristic để hướng dẫn quá trình tìm kiếm.

Các biến thể đầu tiên (1992-1996):

- Sau khi Ant System được giới thiệu, Marco Dorigo và các đồng sự đã phát triển thêm các biến thể nhằm cải thiện hiệu quả của ACO:
- Elitist Ant System: Tăng cường pheromone trên giải pháp tốt nhất.

- Ant-Q: Kết hợp các khái niệm từ học tăng cường (reinforcement learning).

1.2.3. Ant Colony System (ACS - 1996)

Một trong những cải tiến lớn nhất là thuật toán ACS, được phát triển bởi Dorigo và Gambardella.

Đặc điểm mới của ACS:

- Cập nhật pheromone cục bộ trong quá trình xây dựng giải pháp.
- Quy tắc chọn ngẫu nhiên giả (pseudorandom proportional rule) giúp cân bằng giữa khai thác và khám phá.

1.2.4. MAX-MIN Ant System (MMAS - 1997)

- Tối ưu hóa hiệu quả bằng cách giới hạn giá trị pheromone trong khoảng *[τmin,τmax][\tau_{\text{min}}, \tau_{\text{max}}]*[τmin ,τmax].
- Chỉ cho phép kiến tốt nhất cập nhật pheromone.

1.2.5. Các biến thể khác (1998-2005):

- **Rank-Based Ant System:** Sắp xếp kiến theo chất lượng giải pháp trước khi cập nhật pheromone.
- **Best-Worst Ant System (BWAS):** Tăng cường khả năng khám phá bằng cách giảm pheromone trên giải pháp tệ nhất.
- **Hyper-Cube Framework:** Biến đổi cách lưu trữ pheromone thành không gian hyper-cube để cải thiện hiệu quả.

1.3. Mở rộng ứng dụng và lý thuyết (2000-2005)

Mở rộng ứng dụng ACO:

Từ bài toán người bán hàng (TSP), ACO được mở rộng sang các lĩnh vực khác như:

- Định tuyến phương tiện (Vehicle Routing Problem - VRP).
- Lập lịch công việc.
- Bài toán tập hợp con.
- Các bài toán trong mạng viễn thông.

Phát triển lý thuyết:

- Chứng minh hội tụ: Các nghiên cứu lý thuyết (như của Gutjahr) đã chứng minh ACO có khả năng hội tụ đến giải pháp tối ưu với xác suất 1.

Liên kết với các phương pháp khác:

- So sánh với học tăng cường (reinforcement learning).
- Tương đồng với các phương pháp dựa trên mô hình xác suất (probabilistic model-based search).

Ứng dụng của thuật toán ACO trong các bài toán tối ưu

Bài toán NP-hard là các bài toán tối ưu hóa và quyết định mà không có thuật toán chính xác nào có thể giải quyết trong thời gian đa thức khi kích thước đầu vào lớn. ACO đã được áp dụng thành công vào một số bài toán NP-hard điển hình, với khả năng tìm giải pháp gần tối ưu trong thời gian ngắn hơn so với các phương pháp cổ điển.

1.3.1. Bài toán người bán hàng (TSP)

TSP là bài toán tìm hành trình ngắn nhất để đi qua tất cả các thành phố một lần duy nhất và quay về điểm xuất phát.

Cách ACO giải quyết:

- Các kiến nhân tạo di chuyển qua đồ thị biểu diễn các thành phố và chọn lộ trình dựa trên lượng pheromone (dấu hiệu đường dẫn tốt từ các vòng lặp trước) và heuristic (nghịch đảo khoảng cách).
- Sau mỗi vòng lặp, pheromone trên các cạnh thuộc hành trình ngắn nhất sẽ được tăng cường, giúp hội tụ nhanh chóng vào giải pháp tốt nhất.

1.3.2. Bài toán lập lịch

Các bài toán lập lịch đòi hỏi sắp xếp thứ tự thực hiện các công việc trên các tài nguyên (máy móc, nhân lực) sao cho tối ưu hóa tiêu chí (như thời gian hoàn thành hoặc chi phí).

Cách ACO giải quyết:

- Mỗi công việc được coi là một nút trên đồ thị. Kiến sẽ tìm thứ tự sắp xếp công việc dựa trên pheromone và thông tin heuristic.
- Phản hồi tích cực từ các giải pháp tốt giúp cải thiện thứ tự sắp xếp qua các vòng lặp.

1.3.3. Bài toán định tuyến phương tiện (VRP)

VRP yêu cầu tối ưu hóa hành trình cho đội xe giao hàng sao cho tổng chi phí (khoảng cách hoặc thời gian) là nhỏ nhất và thỏa mãn các ràng buộc như tải trọng xe hoặc thời gian giao hàng.

Cách ACO giải quyết:

- Kiến nhân tạo phân chia các đơn hàng thành các tuyến đường và tối ưu hóa từng tuyến.
- Pheromone được cập nhật để tăng cường các tuyến đường ngắn nhất, dẫn đến việc hội tụ vào giải pháp tối ưu hoặc gần tối ưu.

1.3.4. Bài toán tập hợp con (Subset Problems)

Đây là các bài toán yêu cầu chọn một tập hợp con của các phần tử để tối ưu hóa một hàm mục tiêu nào đó.

Ví dụ: Bài toán ba lô (Knapsack Problem) – chọn các vật phẩm để tối đa hóa giá trị mà không vượt quá trọng lượng tối đa của ba lô.

Cách ACO giải quyết:

- Kiến nhân tạo chọn các phần tử thuộc tập hợp con dựa trên pheromone và thông tin heuristic, giúp đạt giải pháp gần tối ưu.
- ACO được sử dụng để tìm các đường dẫn ngắn nhất với chi phí thấp nhất, đồng thời thích nghi nhanh với các thay đổi trong mạng.

CHƯƠNG 2: ĐỘNG LỰC ĐỀ XUẤT THUẬT TOÁN ACO

Thuật toán ACO được phát triển dựa trên cảm hứng từ hành vi tự nhiên của loài kiến. Những nghiên cứu từ giữa thế kỷ 20 đã chỉ ra rằng loài kiến sử dụng một cơ chế giao tiếp gián tiếp qua môi trường, gọi là **stigmergy**, để tổ chức hành vi tập thể một cách hiệu quả. Các đặc tính sinh học này là cơ sở lý thuyết để xây dựng ACO, một thuật toán tối ưu hóa tổ hợp mạnh mẽ.

2.1. Nguồn cảm hứng từ hành vi của loài kiến

Hành vi tìm kiếm thức ăn của loài kiến là một quá trình tự tổ chức đặc biệt hiệu quả, được nghiên cứu chi tiết bởi Pierre-Paul Grassé và các nhà khoa học khác.

2.1.1. Stigmergy – Giao tiếp qua môi trường:

Stigmergy là một cơ chế mà trong đó các sinh vật không giao tiếp trực tiếp với nhau mà tương tác thông qua các tín hiệu để lại trong môi trường.

Ở loài kiến, tín hiệu này là **pheromone**, một loại hóa chất mà chúng tiết ra trên đường đi.

2.1.2. Quá trình tìm kiếm thức ăn:

Bắt đầu ngẫu nhiên:

- Khi tìm kiếm thức ăn, các con kiến di chuyển ngẫu nhiên từ tổ.

Phát hiện nguồn thức ăn:

- Nếu một con kiến phát hiện nguồn thức ăn, nó quay lại tổ và để lại pheromone trên đường đi.

Phản hồi tích cực:

- Các con kiến khác bị hấp dẫn bởi pheromone và bắt đầu sử dụng đường dẫn đó.
- Đường nào ngắn hơn sẽ tích lũy pheromone nhanh hơn, thu hút nhiều kiến hơn, dẫn đến việc hội tụ vào đường ngắn nhất.

2.1.3. Minh họa bằng thí nghiệm:

Thí nghiệm cầu đôi (Double Bridge) đã được thực hiện để minh họa hành vi này.

- Hai cây cầu (một dài, một ngắn) nối tổ kiến với nguồn thức ăn. Ban đầu, các kiến chọn ngẫu nhiên giữa hai cầu.

- Do cầu ngắn có lượng pheromone tích tụ nhanh hơn, đàn kiến nhanh chóng hội tụ vào cầu ngắn, tối ưu hóa hành trình.

Kết luận:

- Hành vi tìm kiếm thức ăn của loài kiến không chỉ cho thấy khả năng tối ưu hóa đường đi mà còn làm nổi bật sự tự tổ chức hiệu quả trong các hệ thống phi tập trung.

2.2. Các đặc điểm nổi bật của ACO

Dựa trên cơ chế của loài kiến, ACO được xây dựng với những đặc tính nổi bật sau:

Phản hồi tích cực:

- Các giải pháp tốt sẽ được củng cố qua mỗi vòng lặp nhờ việc tăng cường pheromone trên các đường dẫn thuộc giải pháp tốt nhất.
- Điều này giúp thuật toán hội tụ nhanh hơn vào các giải pháp tối ưu hoặc gần tối ưu.

Tính ngẫu nhiên:

- Trong quá trình tìm kiếm, các con kiến nhân tạo vẫn duy trì yếu tố ngẫu nhiên, cho phép khám phá các giải pháp mới và tránh rơi vào cực trị cục bộ (local optimum).

Tự tổ chức:

- ACO mô phỏng tính tự tổ chức của loài kiến, giúp thuật toán dễ dàng thích nghi với nhiều loại bài toán khác nhau.
- Các kiến nhân tạo hoạt động độc lập nhưng thông qua cơ chế pheromone, chúng có thể phối hợp để tìm ra giải pháp tối ưu.

2.3. Tại sao ACO lại hiệu quả trong tối ưu hóa?

ACO là một thuật toán hiệu quả trong tối ưu hóa nhờ sự kết hợp cân bằng giữa khai thác thông tin từ các giải pháp tốt nhất và khám phá các giải pháp mới.

2.3.1. Học từ môi trường:

Lượng pheromone được lưu giữ trên các cạnh đại diện cho thông tin tích lũy từ các giải pháp tốt nhất trong các vòng lặp trước đó.

Quá trình này giúp thuật toán học hỏi từ lịch sử tìm kiếm để cải thiện kết quả trong các vòng lặp sau.

2.3.2. Cân bằng giữa khai thác và khám phá:

ACO sử dụng cơ chế xác suất để chọn các đường dẫn, kết hợp:

- **Khai thác:** Chọn các đường dẫn có lượng pheromone lớn để tối ưu hóa dựa trên giải pháp tốt nhất đã tìm được.
- **Khám phá:** Đảm bảo một phần khả năng ngẫu nhiên để tìm kiếm các giải pháp tiềm năng mới.
- Sự cân bằng này giúp ACO tránh hiện tượng hội tụ cục bộ, đồng thời tối ưu hóa không gian tìm kiếm.

2.3.3. Sử dụng thông tin heuristic:

Bên cạnh pheromone, ACO còn tận dụng thông tin heuristic (ví dụ: khoảng cách ngắn nhất trong bài toán người bán hàng) để tăng tốc độ hội tụ.

2.3.4. Tính linh hoạt:

ACO có thể được điều chỉnh và áp dụng cho nhiều bài toán tối ưu hóa khác nhau, từ bài toán NP-hard như TSP đến các bài toán động và không chắc chắn.

CHƯƠNG 3: MÔ TẢ THUẬT TOÁN ACO

Thuật toán Ant Colony Optimization (ACO) là một phương pháp tối ưu hóa metaheuristic dựa trên hành vi kiếm ăn của loài kiến trong tự nhiên. ACO mô phỏng cách các con kiến nhân tạo di chuyển trên một đồ thị, tương tác thông qua pheromone để tìm kiếm và tối ưu hóa giải pháp. Phần này mô tả chi tiết cơ chế hoạt động và các thành phần chính của ACO.

3.1. Tổng quan về cơ chế hoạt động của ACO

Thuật toán ACO mô phỏng các con kiến nhân tạo di chuyển qua các đỉnh và cạnh của một đồ thị, đại diện cho không gian giải pháp của bài toán. Cơ chế hoạt động chính bao gồm:

Xây dựng giải pháp:

- Các con kiến nhân tạo bắt đầu từ một đỉnh khởi đầu và xây dựng giải pháp từng bước bằng cách chọn các cạnh trên đồ thị.
- Quyết định lựa chọn cạnh dựa trên xác suất, tính theo lượng pheromone và thông tin heuristic trên mỗi cạnh.

Cập nhật pheromone:

- Sau khi các con kiến hoàn thành giải pháp, lượng pheromone trên các cạnh sẽ được cập nhật dựa trên chất lượng của giải pháp đó.
- Cơ chế bay hơi pheromone giúp loại bỏ các thông tin không hữu ích, tránh hiện tượng hội tụ sớm vào các giải pháp kém.

Ý nghĩa:

- Cơ chế này cho phép thuật toán khám phá không gian tìm kiếm, học từ các giải pháp tốt trước đó và hội tụ dần vào giải pháp tối ưu.

3.2. Các bước cơ bản trong thuật toán ACO

Bước 1: Khởi tạo pheromone

Mỗi cạnh của đồ thị được khởi tạo một lượng pheromone ban đầu $\tau 0$, thường là một giá trị nhỏ không âm.

Bước 2: Xây dựng giải pháp

Mỗi con kiến nhân tạo sẽ xây dựng một giải pháp bằng cách chọn các cạnh từ một đỉnh đến các đỉnh khác

Công thức xác suất lựa chọn cạnh:

$$P_{ij} = \frac{\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum_{k \in N(i)} \tau_{ik}^{\alpha} \cdot \eta_{ik}^{\beta}}$$

Trong đó:

- Pij : Xác suất chọn cạnh (i,j).
- Tij : Lượng pheromone trên cạnh (i,j).
- ηij : Giá trị heuristic, thường là nghịch đảo của chi phí cạnh (i,j), ví dụ: ηij=1/dij.
- α: Hệ số trọng số cho pheromone, kiểm soát mức độ ảnh hưởng của thông tin pheromone.
- β: Hệ số trọng số cho thông tin heuristic, kiểm soát mức độ ưu tiên các cạnh có heuristic tốt.
- N(i): Tập các đỉnh hợp lệ từ đỉnh i

Ví dụ: Trong bài toán người bán hàng (TSP), heuristic là nghịch đảo của khoảng cách giữa các thành phố, giúp các cạnh ngắn hơn được ưu tiên.

Bước 3: Cập nhật pheromone

Bay hơi pheromone:

$$\tau_{ij} \leftarrow (1 - \rho) \cdot \tau_{ij}$$

- Trong đó ρ ($0 < \rho < 1$) là hệ số bay hơi, giúp giảm dần lượng pheromone, loại bỏ các thông tin kém quan trọng.

Tăng cường pheromone:

- Pheromone trên các cạnh thuộc giải pháp tốt được tăng cường:

$$\tau_{ij} \leftarrow \tau_{ij} + \Delta\tau_{ij}$$

Trong đó:

$$\Delta\tau_{ij} = \sum_{k=1}^{m} \frac{Q}{L_k},$$

- với Q là hằng số và Lk là chi phí của giải pháp kiến k

Ý nghĩa:

- Bay hơi giúp tránh hiện tượng hội tụ sớm, trong khi tăng cường pheromone khuyến khích các giải pháp tốt.

Bước 4: Kiểm tra điều kiện dừng

Thuật toán dừng khi:

- Đạt số vòng lặp tối đa.
- Không có cải thiện đáng kể trong nhiều vòng lặp liên tiếp.

3.3. Các tham số chính trong thuật toán ACO

Pheromone (τ)

- Đại diện cho thông tin tích lũy từ các giải pháp tốt trong các vòng lặp trước.
- Điều chỉnh thông qua cơ chế bay hơi và tăng cường.

Hệ số trọng số (α,β)

- α*alpha*α: Quy định mức độ ảnh hưởng của pheromone.
- β*beta*β: Quy định mức độ ảnh hưởng của thông tin heuristic.

Hệ số bay hơi (ρ)

- Quy định tốc độ giảm pheromone qua thời gian.

Kích thước đàn kiến (m)

- Số lượng kiến nhân tạo trong mỗi vòng lặp.

3.4. Các biến thể của ACO

Ant System (AS):

- Phiên bản gốc của ACO, tất cả các kiến đều đóng góp vào việc cập nhật pheromone.

MAX-MIN Ant System (MMAS):

$$\tau_{ij} \in [\tau_{\min}, \tau_{\max}]$$

- Cải tiến bằng cách giới hạn lượng pheromone trong khoảng *[τmin,τmax][*
- Chỉ các giải pháp tốt nhất (tốt nhất iteration hoặc tốt nhất so-far) được phép tăng cường pheromone.
- Điều này kiểm soát lượng pheromone, tránh hiện tượng hội tụ vào một giải pháp duy nhất.

Ant Colony System (ACS):

- Thêm cơ chế cập nhật pheromone cục bộ:

$$\tau_{ij} = (1 - \phi) \cdot \tau_{ij} + \phi \cdot \tau_0$$

Với ϕ là hệ số cập nhật cục bộ, giúp duy trì tính ngẫu nhiên và tránh hiện tượng hội tụ sớm.

Hybrid ACO:

- Kết hợp ACO với các phương pháp khác như tìm kiếm cục bộ (local search) hoặc thuật toán di truyền để tăng hiệu quả tìm kiếm.

CHƯƠNG 4: ỨNG DỤNG THUẬT TOÁN ACO GIẢI BÀI TOÁN TSP

4.1. Bài toán TSP

4.1.1. 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.

4.1.2. Tư tưởng giải quyết bài toán TSP bằng thuật toán ACO.

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.

4.2. Thuật toán ACO giải bài toán TSP

4.2.1. Các bước của thuật toán ACO giải bài toán TSP

Dưới đây là cách triển khai **Ant System (AS)**, phiên bản gốc của thuật toán **ACO**, để giải bài toán TSP. Đây là thuật toán, trong đó mọi con kiến đều đóng góp pheromone cho các lộ trình đã đi qua.

Khởi tạo các thông số

- Số lượng kiến (m): Tổng số kiến tham gia giải quyết bài toán.
- Pheromone ban đầu (Tij): Mỗi cạnh (i,j)trên đồ thị được khởi tạo với một giá trị pheromone ban đầu (T0).

Hệ số ảnh hưởng:

- α: Quyết định mức độ ảnh hưởng của pheromone.
- β: Quyết định mức độ ảnh hưởng của độ hấp
- Tốc độ bốc hơi pheromone (ρ): Quy định mức độ giảm của pheromone sau mỗi vòng lặp.
- Tổng lượng pheromone (Q): Xác định lượng pheromone được bổ sung trên các cạnh.

- Số vòng lặp (Max Iterations): Giới hạn số vòng lặp của thuật toán.

Lặp qua các vòng lặp (Iterations)

- Mỗi con kiến xây dựng một lộ trình
- Xuất phát từ một thành phố ngẫu nhiên, mỗi con kiến di chuyển qua các thành phố chưa thăm.

Xác suất để chọn thành phố tiếp theo được tính bằng công thức:

$$P_{ij} = \begin{cases} \frac{\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum_{k \in \mathrm{L}} \tau_{ik}^{\alpha} \cdot \eta_{ik}^{\beta}}, & \text{nếu } j \notin \mathrm{L} \text{ (thành phố đã thăm)}, \\ 0, & \text{ngược lại.} \end{cases}$$

Trong đó:

- τ_{ij}: Pheromone trên cạnh (i, j).
- $\eta_{ij} = \frac{1}{d_{ij}}$: Độ hấp dẫn (nghịch đảo khoảng cách giữa i và j).
- L: Tập hợp các thành phố đã thăm.

Đánh giá lộ trình

- Tính tổng độ dài lộ trình (Lk) cho mỗi con kiến k

$$L_k = \sum_{i=1}^{n} d_{path[i],path[i+1]}$$

- Với path là lộ trình mà con kiến đã đi qua.

Cập nhật pheromone

- Bốc hơi pheromone

Lượng pheromone trên tất cả các cạnh giảm dần theo công thức:

$$\tau_{ij} \leftarrow (1 - \rho) \cdot \tau_{ij}$$

- Bổ sung pheromone

Các cạnh thuộc lộ trình của kiến kkk được bổ sung pheromone dựa trên độ dài lộ trình (Lk):

$$\tau_{ij} \leftarrow \tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}$$

Trong đó:

$$\Delta\tau_{ij}^{k} = \begin{cases} \frac{Q}{L_k}, & \text{nếu } (i,j) \text{ thuộc lộ trình của kiến } k, \\ 0, & \text{ngược lại.} \end{cases}$$

Lặp lại quá trình

- Quá trình xây dựng lộ trình và cập nhật pheromone được lặp lại qua nhiều vòng lặp, cho đến khi đạt một trong các điều kiện dừng:
- Số vòng lặp đạt giới hạn.
- Không có cải thiện đáng kể về lộ trình tốt nhất trong nhiều vòng lặp.

Kết quả

- Lộ trình tốt nhất được tìm thấy sau khi thuật toán kết thúc, cùng với tổng độ dài ngắn nhất.

CHƯƠNG 5: XÂY DỰNG CHƯƠNG TRÌNH ÁP DỤNG THUẬT TOÁN ACO GIẢI BÀI TOÁN TSP

5.1. Tính ma trận khoảng cách

Hàm này nhận vào danh sách các tọa độ của các thành phố và tính toán khoảng cách Euclidean giữa mỗi cặp thành phố. Ma trận khoảng cách có kích thước n x n (với n là số lượng thành phố), trong đó mỗi phần tử [i][j] lưu trữ khoảng cách giữa thành phố i và thành phố j.

```
def calculate_distance_matrix(coords):
    n = len(coords)
    return np.array([[np.linalg.norm(np.array(coords[i]) - np.array(coords[j])) if i != j
```

Đây là cách tính khoảng cách giữa các thành phố theo công thức Euclidean:

$$d(i,j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

với ((xi,yi) là tọa độ của thành phố i.

5.2. Thuật toán Ant

Khởi tạo pheromone: Ban đầu, giá trị pheromone giữa mọi cặp thành phố đều được khởi tạo bằng 1.

Khởi tạo các thông số: Các thông số quan trọng trong thuật toán ACO bao gồm:

- n_ants: Số lượng kiến (ants).
- n_iterations: Số vòng lặp (iterations).
- alpha: Hệ số ảnh hưởng của pheromone.
- beta: Hệ số ảnh hưởng của heuristic (tính nghịch đảo của khoảng cách).
- rho: Tỷ lệ bay hơi pheromone (pheromone evaporation rate).
- Q: Một hằng số dùng để điều chỉnh lượng pheromone mới.

Trong mỗi vòng lặp:

- Mỗi con kiến bắt đầu từ một thành phố ngẫu nhiên và chọn thành phố tiếp theo dựa trên tỷ lệ pheromone và heuristic (được tính từ khoảng cách).

Sau khi tất cả các con kiến hoàn thành chuyến đi của mình, thuật toán sẽ cập nhật pheromone trên các con đường đã đi qua, sao cho những con đường ngắn hơn (tức là có chiều dài nhỏ hơn) sẽ nhận được nhiều pheromone hơn.

Pheromone sẽ bay hơi một phần để tránh việc kiến tập trung quá nhiều vào một số con đường.

```
def ant_system(coords, n_ants=10, n_iterations=100, alpha=1, beta=2, rho=0.1, Q=100):
    n = len(coords)
    dist_matrix = calculate_distance_matrix(coords)
    pheromone = np.ones((n, n))
    best_path, best_length = None, float('inf')

    for _ in range(n_iterations):
        paths, lengths = [], []
        for _ in range(n_ants):
            path, visited = [random.randint(0, n - 1)], set()
            visited.add(path[0])

            for _ in range(n - 1):
                current = path[-1]
                probs = [(pheromone[current][j]**alpha) * ((1 / dist_matrix[current][j])**
                next_city = random.choices(range(n), weights=probs)[0]
                path.append(next_city)
                visited.add(next_city)
```

```
            path.append(path[0])
            length = sum(dist_matrix[path[i]][path[i + 1]] for i in range(n))
            paths.append(path)
            lengths.append(length)

        min_length = min(lengths)
        if min_length < best_length:
            best_length = min_length
            best_path = paths[lengths.index(min_length)]

        pheromone *= (1 - rho)
        for path, length in zip(paths, lengths):
            for i in range(n):
                pheromone[path[i]][path[i + 1]] += Q / length

    return best_path, best_length
```

5.3. Chạy thuật toán:

Tọa độ các thành phố được lưu trữ trong danh sách coords. Các thành phố này được lựa chọn ngẫu nhiên.

Hàm ant_system được gọi với các tham số như số lượng kiến, số vòng lặp, hệ số pheromone, hệ số heuristic, và tỷ lệ bay hơi pheromone.

Sau khi chạy thuật toán, kết quả lộ trình tốt nhất và chiều dài ngắn nhất của nó sẽ được in ra.

```
coords = [(0, 0), (1, 2), (2, 4), (3, 1), (4, 3)]

# Chạy thuật toán
best_path, best_length = ant_system(coords)
print(f"Lộ trình tốt nhất: {best_path}")
print(f"Chiều dài ngắn nhất: {best_length}")
```

Các tham số có thể điều chỉnh:

- n_ants: Số lượng kiến (ants) tham gia tìm kiếm. Số lượng lớn sẽ giúp thuật toán tìm được giải pháp tốt hơn, nhưng cũng tốn thời gian hơn.
- n_iterations: Số vòng lặp. Chạy nhiều vòng giúp cải thiện độ chính xác của kết quả.
- alpha và beta: Điều chỉnh ảnh hưởng của pheromone và khoảng cách. Tăng alpha sẽ làm cho pheromone ảnh hưởng mạnh hơn đến việc chọn đường đi, trong khi tăng beta sẽ làm cho khoảng cách trở nên quan trọng hơn.
- rho: Tỷ lệ bay hơi pheromone, giúp tránh việc tập trung quá nhiều vào một vài con đường.
- Q: Hằng số điều chỉnh lượng pheromone mới được thêm vào.

CHƯƠNG 6: ỨNG DỤNG CỦA ACO

6.1. Trong các bài toán NP-Hard

Trong ACO (Ant Colony Optimization), các ứng dụng của thuật toán thường liên quan đến việc giải quyết các bài toán NP-Hard. Bài toán NP-Hard là những bài toán rất khó giải bằng cách sử dụng các thuật toán chính xác trong thời gian hợp lý khi kích thước bài toán tăng lên. ACO được phát triển để tìm các giải pháp gần tối ưu cho những bài toán này trong một khoảng thời gian có thể chấp nhận được.

Trong trường hợp của ACO, loại nghiên cứu này ban đầu bao gồm kiểm tra các thuật toán trên TSP. Sau đó, các vấn đề khó khác cũng được xem xét. Cho đến nay, ACO đã được thử nghiệm trên có lẽ hơn một trăm vấn đề N P-hard khác nhau. Nhiều vấn đề được giải quyết có thể được coi là thuộc một trong các loại sau: routing problems, assignment problem, scheduling problems, subset problems.Ngoài ra, ACO đã được ứng dụng thành công cho các vấn đề khác đang nổi lên trong các lĩnh vực như học máy và tin sinh học.

6.2. Ứng dụng cho mạng viễn thông

Ant Colony Optimization (ACO) đã được ứng dụng rộng rãi trong các mạng viễn thông để giải quyết nhiều vấn đề tối ưu hóa phức tạp. Trong các hệ thống mạng, ACO có thể hỗ trợ tối ưu hóa việc phân bổ tài nguyên, lập lịch, tìm đường đi tối ưu, và giảm thiểu độ trễ, giúp cải thiện hiệu suất tổng thể của mạng. Dưới đây là một số ứng dụng của ACO trong các mạng viễn thông.

Các thuật toán ACO đã được chứng minh là một cách tiếp cận rất hiệu quả cho các vấn đề định tuyến trong mạng viễn thông, nơi các thuộc tính của hệ thống, chẳng hạn như chi phí sử dụng liên kết hoặc tính khả dụng của các nút, thay đổi theo thời gian. Các thuật toán ACO lần đầu tiên được áp dụng cho các vấn đề định tuyến trong mạng chuyển mạch (chẳng hạn như mạng điện thoại) và sau đó trong các mạng chuyển mạch gói (chẳng hạn như mạng cục bộ hoặc Internet) .Theo bằng chứng khái niệm do Schoonderwoerd và cộng sự cung cấp, các thuật toán định tuyến lấy cảm hứng từ kiến cho mạng viễn thông đã được cải thiện đến mức hiện đại trong mạng có dây. Một ví dụ nổi tiếng là AntNet ,AntNet đã được thử nghiệm rộng rãi, trong mô phỏng, trên các

mạng khác nhau và theo các mô hình lưu lượng khác nhau, chứng tỏ khả năng thích ứng cao và mạnh mẽ. So sánh với các thuật toán định tuyến hiện đại đã chỉ ra rằng, trong hầu hết các tình huống được xem xét, AntNet vượt trội so với các đối thủ cạnh tranh. Các thuật toán dựa trên ant đã tạo ra một số thuật toán định tuyến khác, nâng cao hiệu suất trong nhiều tình huống mạng có dây. Gần đây hơn, một thuật toán ACO được thiết kế cho lớp mạng ad hoc di động đầy thách thức đã được chứng minh là cạnh tranh với các thuật toán định tuyến hiện đại đồng thời cung cấp khả năng mở rộng tốt hơn.

6.3. Ứng dụng cho các vấn đề công nghiệp

Ant Colony Optimization (ACO) đã được ứng dụng thành công trong nhiều vấn đề công nghiệp để tối ưu hóa các quy trình sản xuất, chuỗi cung ứng, và các vấn đề tối ưu hóa khác. Trong ngành công nghiệp, ACO có thể giúp cải thiện hiệu quả, giảm chi phí, và tăng cường khả năng cạnh tranh thông qua việc giải quyết các bài toán phức tạp mà các thuật toán chính xác khó có thể thực hiện được trong thời gian hợp lý.

Thành công về các vấn đề học thuật đã thu hút sự chú ý của một số công ty đã bắt đầu sử dụng thuật toán ACO cho các ứng dụng trong thế giới thực. Trong số những thuật toán đầu tiên khai thác các thuật toán dựa trên metaheuristic ACO là EuroBios (www.eurobios.com). Họ đã áp dụng ACO cho một số vấn đề lập kế hoạch khác nhau như vấn đề cửa hàng dòng chảy hai giai đoạn liên tục với các hồ chứa hữu hạn. Các vấn đề được mô hình hóa bao gồm các ràng buộc trong thế giới thực khác nhau như thời gian thiết lập, hạn chế công suất, khả năng tương thích tài nguyên và lịch bảo trì. Một công ty khác đã và vẫn đóng một vai trò rất quan trọng trong việc thúc đẩy ứng dụng ACO trong thế giới thực là AntOptima (www.antoptima .com). Các nhà nghiên cứu của AntOptima đã phát triển một bộ công cụ để giải quyết các vấn đề định tuyến xe có các thuật toán tối ưu hóa dựa trên ACO. Các sản phẩm đặc biệt thành công dựa trên các công cụ này là DYVOIL, để quản lý và tối ưu hóa việc phân phối dầu sưởi ấm với một đội xe tải không đồng nhất, được Pina Petroli ở Thụy Sĩ sử dụng lần đầu tiên và AntRoute, để định tuyến hàng trăm phương tiện của các công ty như Migros Thụy Sĩ, hay Barilla, nhà sản xuất mì ống chính của Ý. Một ứng dụng định tuyến phương tiện khác đã được phát triển bởi BiosGroup cho công ty Air Liquide của Pháp. Các ứng dụng thú vị khác trong thế giới thực là của Gravel, Price và Gagne', những người đã áp dụng

ACO cho một vấn đề lập kế hoạch công nghiệp trong một trung tâm đúc nhôm, và của Bautista và Pereira, những người đã áp dụng thành công ACO để giải quyết vấn đề cân bằng dây chuyền lắp ráp với nhiều mục tiêu và ràng buộc giữa các nhiệm vụ

6.4. Các chủ đề nóng hiện nay trong ACO

Trong lĩnh vực Ant Colony Optimization (ACO), có một số chủ đề đang được nghiên cứu và phát triển mạnh mẽ để giải quyết các vấn đề phức tạp và mở rộng khả năng áp dụng của thuật toán trong nhiều lĩnh vực khác nhau như :

- Các vấn đề tối ưu hóa động: Các vấn đề tối ưu hóa động liên quan đến các bài toán mà trong đó môi trường, thông số hoặc các điều kiện thay đổi theo thời gian. Ví dụ, trong các mạng viễn thông, lưu lượng mạng có thể thay đổi, hay trong các bài toán sản xuất, các yêu cầu về sản phẩm có thể thay đổi.
- Vấn đề tối ưu hóa ngẫu nhiên : Các bài toán tối ưu hóa ngẫu nhiên liên quan đến việc tìm kiếm giải pháp tối ưu trong các điều kiện có sự biến động ngẫu nhiên, như các bài toán tối ưu hóa trong các môi trường không chắc chắn hoặc các hệ thống với các yếu tố ngẫu nhiên không thể dự đoán trước.
- Tối ưu hóa đa mục tiêu : Các bài toán tối ưu hóa đa mục tiêu yêu cầu tối ưu hóa đồng thời nhiều mục tiêu khác nhau, và thường không có một giải pháp duy nhất mà thay vào đó là một tập hợp các giải pháp tốt nhất, gọi là "Front Pareto" (tập hợp các giải pháp không thể cải thiện một mục tiêu mà không làm giảm chất lượng của mục tiêu khác).
- Triển khai song song : Để cải thiện tốc độ và hiệu quả của ACO, việc triển khai song song (parallelization) giúp tận dụng tối đa tài nguyên tính toán và tăng cường khả năng xử lý các bài toán lớn và phức tạp
- Tối ưu hóa liên tục : Tối ưu hóa liên tục liên quan đến các bài toán trong đó không gian giải pháp là một không gian liên tục, thay vì không gian rời rạc. Các bài toán như tối ưu hóa các tham số trong các mô hình học máy hoặc tối ưu hóa trong các hệ thống vật lý có thể là các bài toán tối ưu hóa liên tục.

KẾT LUẬN

Ant Colony Optimization (ACO) là một kỹ thuật tối ưu hóa hiệu quả, lấy cảm hứng từ hành vi tự tổ chức của loài kiến, sử dụng cơ chế pheromone để dẫn dắt quá trình tìm kiếm giải pháp. Với khả năng giải quyết các bài toán tối ưu tổ hợp phức tạp và tính linh hoạt trong ứng dụng, ACO đã chứng minh vai trò quan trọng trong các lĩnh vực như định tuyến mạng, lập lịch và học máy. Thành công của ACO đến từ việc kết hợp cơ chế tự nhiên của loài kiến với các cải tiến như bay hơi pheromone, cập nhật dựa trên chất lượng giải pháp, và tích hợp các phương pháp tìm kiếm cục bộ, giúp tăng cường hiệu quả và tránh hội tụ cục bộ.

Trong tương lai, ACO hứa hẹn sẽ tiếp tục mở rộng phạm vi ứng dụng sang các lĩnh vực mới như phân tích dữ liệu lớn, y sinh học và tối ưu hóa năng lượng. Đồng thời, việc nghiên cứu lý thuyết sâu hơn về hội tụ, giảm chi phí tính toán và kết hợp với các phương pháp hiện đại như học sâu hoặc học tăng cường sẽ nâng cao hơn nữa hiệu quả của ACO. Những hướng phát triển này không chỉ giúp ACO duy trì vị thế hàng đầu trong các phương pháp metaheuristic mà còn mở ra những khả năng ứng dụng rộng lớn trong các hệ thống thực tế và động lực học phức tạp.

TÀI LIỆU KHAM KHẢO

Santosa, Budi, Metoda Metaheuristik, Konsep dan Implementasi, Guna Widya, Surabaya, Indonesia, 2011.

M. Dorigo and L.M. Gambardella. Ant colony system: A cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation, 1(1):53 66, 1997.

M. Dorigo, V. Maniezzo, and A. Colorni. The ant system optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and CyberneticsPart B, 26(1):29–41, 1996.

Wallace, R. J. (1996). Analysis of heuristic methods for partial constraint satisfaction problems. In E.

Freuder (Ed.), Principles and Practice of Constraint Programming—CP'96, vol. 1118 of Lecture Notes in

Computer Science (pp. 482–496). Berlin, Springer-Verlag.

Wallace, R. J., & Freuder, E. C. (1996). Heuristic methods for over-constrained constraint satisfaction

problems. In M. Jampel, E. Freuder, & M. Maher (Eds.), OCS'95: Workshop on Over-Constrained Sys

tems at CP'95, vol. 1106 of Lecture Notes in Computer Science (pp. 207–216). Berlin, Springer-Verlag.