

**ỦY BAN NHÂN DÂN THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐH SÀI GÒN**

**KHOA CÔNG NGHỆ THÔNG TIN**

-----🙞🙜🕮🙞🙜-----

**TIỂU LUẬN HỌC PHẦN SEMINAR CHUYÊN ĐỀ**

**THUẬT TOÁN TỐI ƯU HÓA ĐÀN KIẾN**

**GIẢI BÀI TOÁN NGƯỜI BÁN HÀNG**

Sinh viên: Phùng Hà Thảo

GVHD: TS. Phan Tấn Quốc

**Thành phố Hồ Chí Minh, tháng 5 năm 2021**

**DANH MỤC**

[DANH MỤC THUẬT NGỮ VIẾT TẮT 5](#_Toc71383219)

[DANH MỤC BẢNG 6](#_Toc71383220)

[DANH MỤC HÌNH VẼ 7](#_Toc71383221)

[MỞ ĐẦU 9](#_Toc71383222)

[1. Tính cấp thiết của đề tài 9](#_Toc71383223)

[2. Mục tiêu của đề tài 10](#_Toc71383224)

[3. Bố cục của đề tài 10](#_Toc71383225)

[Chương 1: Tối ưu tổ hợp 11](#_Toc71383226)

[1. Bài toán tối ưu tổ hợp tổng quát 11](#_Toc71383227)

[2. Một số bài toán tối ưu tổ hợp trong thực tế 12](#_Toc71383228)

[3. Các cách tiếp cận bài toán 12](#_Toc71383229)

[Chương 2: Bài toán người bán hàng (TSP) 13](#_Toc71383230)

[1. Phát biểu bài toán 13](#_Toc71383231)

[2. Lịch sử nghiên cứu bài toán 13](#_Toc71383232)

[3. Bài toán TSP và đồ thị cấu trúc 14](#_Toc71383233)

[4. Các thuật toán ACO cho bài toán TSP 15](#_Toc71383234)

[4.1. Hệ kiến (AS) 16](#_Toc71383235)

[4.2. Hệ đàn kiến (ACS) 18](#_Toc71383236)

[4.3. Hệ kiến Max-Min 21](#_Toc71383237)

[5. Một số phương pháp khác giải bài toán 23](#_Toc71383238)

[5.1. Giải thuật chính xác 23](#_Toc71383239)

[5.2. Giải thuật gần đúng heuristic 24](#_Toc71383240)

[5.2.1. Giải thuật tham lam 24](#_Toc71383241)

[5.2.2. Thuật toán tìm kiếm cục bộ 25](#_Toc71383242)

[5.2.3. Thuật toán nhánh cận 26](#_Toc71383243)

[5.2.4. Thuật toán di truyền 26](#_Toc71383244)

[5.2.5. Thuật toán bầy ong 28](#_Toc71383245)

[6. Ứng dụng của bài toán 29](#_Toc71383246)

[7. Kết luận chương 30](#_Toc71383247)

[Chương 3: Thuật toán tối ưu hóa đàn kiến 31](#_Toc71383248)

[1. Ý tưởng của bài toán 31](#_Toc71383249)

[2. Quá trình phát triển 32](#_Toc71383250)

[3. Thuật toán ACO cho bài toán TƯTH 33](#_Toc71383251)

[3.1. Đồ thị cấu trúc 33](#_Toc71383252)

[3.2. Mô tả thuật toán ACO tổng quát 34](#_Toc71383253)

[4. Tìm kiếm cục bộ 35](#_Toc71383254)

[5. Một số vấn đề khác khi áp dụng ACO 36](#_Toc71383255)

[6. Nhận xét thuật toán 36](#_Toc71383256)

[7. Các yếu tố quyết định hiệu quả của thuật toán 37](#_Toc71383257)

[8. Ứng dụng của thuật toán 37](#_Toc71383258)

[9. Các điểm hạn chế của ACO 38](#_Toc71383259)

[11. Kết luận chương 38](#_Toc71383260)

[Chương 4: Thuật toán tối ưu hóa đàn kiến giải bài toán người bán hàng 39](#_Toc71383261)

[1. Cơ sở lý thuyết 39](#_Toc71383262)

[2. Phân tích yêu cầu 39](#_Toc71383263)

[3. Thuật toán ACO giải bài toán TSP 39](#_Toc71383264)

[3.1. Khởi tạo lời giải ban đầu 39](#_Toc71383265)

[3.2. Quy tắc chuyển trạng thái 39](#_Toc71383266)

[3.3. Quy tắc cập nhật mùi 40](#_Toc71383267)

[3.4. Khởi trị và khởi tạo lại vệt mùi 41](#_Toc71383268)

[3.5. Thủ tục tìm kiếm cục bộ 41](#_Toc71383269)

[4. Kết luận chương 42](#_Toc71383270)

[Chương 5: Thực nghiệm và đánh giá 43](#_Toc71383271)

[1. Dữ liệu thực nghiệm 43](#_Toc71383272)

[2. Môi trường thực nghiệm 44](#_Toc71383273)

[3. Tham số thực nghiệm 44](#_Toc71383274)

[4. Kết quả thực nghiệm 44](#_Toc71383275)

[5. Kết luận 46](#_Toc71383276)

[KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 48](#_Toc71383277)

[TÀI LIỆU THAM KHẢO 49](#_Toc71383278)

# **DANH MỤC THUẬT NGỮ VIẾT TẮT**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Từ viết tắt** | **Từ đầy đủ** | **Ý nghĩa** |
| 1 | ACo | Ant Colony Optimization | Tối ưu hóa đàn kiến |
| 2 | AS | Ant System | Hệ kiến |
| 3 | ACS | Ant Colony System | Hệ đàn kiến |
| 4 | MMAS | Max – Min Ant System | Hệ kiến Max – Min |
| 5 | GA | Genetic Algorithm | Giải thuật di truyền |
| 6 | TƯTH | Tối ưu hóa tổ hợp | Tối ưu hóa tổ hợp |
| 7 | TSP | Travelling Salesman Problem | Bài toán Người bán hàng |
| 8 | Elitist AS |  |  |
| 9 | Ant - Q |  |  |
| 10 | Rank - based AS |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# **DANH MỤC BẢNG**

# **DANH MỤC HÌNH VẼ**

**Xác nhận của người hướng dẫn**

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Thành phố Hồ Chí Minh, ngày 05 tháng 03 năm 2021

**Người hướng dẫn khoa học**

(Kí và ghi rõ họ tên)

# **MỞ ĐẦU**

1. **Tính cấp thiết của đề tài**

Bài toán Người bán hàng (Travelling Salesman Problem - TSP) là một trong những bài toán kinh điển và khó trong tin học. Bài toán có phát biểu rất đơn giản nhưng rất khó giải trong trường hợp tổng quát với không gian tìm kiếm rộng lớn, khó bởi các thuật toán hiệu quả nhất đã được biết đến có thời gian giải quyết bài toán này tăng dần theo cấp số nhân của n, hay độ phức tạp thuật toán tăng theo hàm số mũ.

TSP có một vài ứng dụng thậm chí trong dạng nguyên thủy của nó như lập kế hoạch, logictic và sản xuất các microchip. Thay đổi đi chút ít nó xuất hiện như một bài toán trong rất nhiều lĩnh vực như việc phân tích gen trong sinh học. Trong những ứng dụng này, khái niệm thành phố có thể thay đổi thành khcash hàng, các điểm hàn trên bảng mạch, các mảnh DNA trong gen và các khái niệm khoảng cách có thể biểu diễn bởi thời gian du lịch hay giá thành, hay giống như sự so sánh giữa các mảnh DNA với nhau. Trong nhiều ứng dụng, các hạn chế truyền thông như giới hạn tài nguyên hay giới hạn thời gian thậm chí còn làm cho bài toán khó hơn.

Trong thực tế khi xây dựng hệ thống thông tin, ta thường gặp các bài toán tối ưu tổ hợp (TƯTH). Trong đó phải tìm các giá trị cho các biến rời rạc để làm cực trị hàm mục tiêu nào đó. Đa số các bài toán này thuộc lớp NP-Khó. Trừ các bài toán cỡ nhỏ có thể tìm được kết quả bằng phương pháp vét cạn, còn lại thì thường không thể tìm được lời giải tối ưu.

Đối với các bài toán cỡ lớn không có phương pháp giải đúng, đến nay người ta vẫn dùng các cách tiếp cận sau:

* Tìm kiếm heuristic để tìm lời giải tốt nhất
* Tìm kiếm cục bộ để tìm lời giải tối ưu địa phương
* Tìm lời giải gần đúng nhờ các thuật toán mô phỏng tự nhiên như: mô phỏng luyện kim, giả thuật di truyền, tối ưu bầy đàn,...

Hai cách tiếp cận đầu cho lời giải nhanh nhưng không thể cải thiện lời giải tìm được, nên cách tiếp cận thứ 3 được sử dụng rộng rãi trong các bài toán cỡ lớn.

Trong phương pháp mô tả tự nhiên, tối ưu đàn kiến (ACO) là cách tiếp cận metaheuristic tương đối mới, được giới thiệu bởi Dorigo năm 1991 đang được nghiên cứu và ứng dụng rộng rãi cho các bài toán tối ưu tổ hợp khó.

1. **Mục tiêu của đề tài**

Áp dụng thuật toán tối ưu hóa đàn kiến để tìm đường đi ngắn nhất trong bài toán Người bán hàng.

Xây dựng giải thuật mô phỏng bài toán người du lịch bằng thuật toán tối ưu hóa đàn kiến.

Thực nghiệm thuật toán tối ưu hóa đàn kiến với ít nhất 30 bộ test để tìm ra được kết quả tốt nhất.

1. **Bố cục của đề tài**

**Chương 1:** Tiểu luận giới thiệu một phát biểu bài toán tối ưu tổ hợp dạng tổng quát.

**Chương 2:** Tìm hiểu về những nét chính của thuật toán tối ưu hóa đàn kiến.

**Chương 3:** Tìm hiểu bài toán người bán hàng.

**Chương 4:** Áp dụng thuật toán tối ưu hóa đàn kiến để giải bài toán người bán hàng.

**Chương 5:** Thực nghiệm và đánh giá lời giải.

Qua đây, tôi xin trân trọng cảm ơn tiến sĩ Phan Tấn Quốc đã hướng dẫn tôi trong quá trình thực hiện bài tiểu luận.

**Chương 1:** Tối ưu tổ hợp

1. **Bài toán tối ưu tổ hợp tổng quát**

Trong đời sống thực tế ta thường phải giải quyết nhiều bài toán TƯTH quan trọng. Chẳng hạn như tìm đường đi ngắn nhất nối hai điểm trên một đồ thị đã cho, lập kế hoạc phân phối nguồn hàng tới nơi tiêu thụ với chi phí cực tiểu, lập kết hoạch phân phối nguồn hàng tới nơi tiêu thụ với chi phí cực tiểu, lập thời khóa biểu cho trường học, định tuyến các gói dữ liệu trên internet, lập lịch cho các hệ thống sản xuất, đối sánh hệ chuỗi gen sinh học phân tử,...

Mỗi bài toán tối ưu tổ hợp có thể phát biểu như sau một bộ ba (𝑆, 𝑓, Ω)

* S là tập hữu hạn trạng thái (lời giải tiềm năng hay phương án)
* 𝑓 là hàm mục tiêu xác định trên S
* Ω là tập các ràng buộc.

Mỗi phương án s ∈ S thỏa mãn ràng buộc Ω gọi là phương án (hay lời giải) chấp nhận được. Mục tiêu của ta là tìm ra phương án chấp nhận được s\* tối ưu hóa toàn cục hàm mục tiêu 𝑓. Chẳng hạn với bài toán cực tiểu thì 𝑓(s\*) < 𝑓(s) với mọi phương án chấp nhận được s. Đối với mỗi bài toán, tồn tại một tập hữu hạn gồm n thành phần C = {c1 , …, cn } sao cho mỗi phương án s trong S đều biểu diễn được nhờ các liên kết của các thành phần trong nó. Cụ thể hơn, các tập S, C và Ω có các đặc tính sau.

1) Ký hiệu X là tập các vectơ C trên độ dài không quá h:X = {<u0,...,un> ui ∈ C, ∀i ≤ k ≤ h}, khi đó mỗi phương án s trong S được xác định nhờ ít nhất một vectơ trong X như ở điểm 2.

2) Tồn tại tập con X\* của X và ánh xạ φ từ X\* lên S sao cho φ­-1(s) không rỗng với mọi s ∈ S. Trong đó tập X\* có thể xây dựng được từ tập con C0 nào đó của C nhờ m rộng tuần tự dưới đây.

3) Từ C0 mở rộng được thành X\* theo thủ tục tuần tự:

i) x0 = < u0 > là mở rộng được với mọi u0 ∈ C0

ii) Giả sử xk = < u0,...,uk > là m rộng được và chưa thuộc X\*. Từ tập ràng buộc Ω, xác định tập con J(xk) của C, sao cho với mọi uk+1 ∈ J(xk) thì xk+1 = < u0,...,uk,uk+1 > là m rộng được.

iii) Với mọi u0 ∈ C0, thủ tục m rộng nêu trên xây dựng được mọi phần tử của X\*.

Như vậy, mỗi bài toán TƯTH được xem là một bài toán cực trị hàm biến, trong đó mỗi biến nhận giá trị trong tập hữu hạn kể cả giá trị rỗng. Một cách nhìn khác, nó là bài toán tìm kiếm v ctơ độ dài không quá trên đồ thị đầy có các đỉnh có nhãn trong tập .

1. **Một số bài toán tối ưu tổ hợp trong thực tế**

Trong đời sống và trong các hệ thông tin, ta thường gặp nhiều bài toán tối ưu tổ hợp quan trọng. Chẳng hạn như: tìm đường đi ngắn nhất nối hai điểm trên một đồ thị đã cho, lập kế hoạch phân phối nguồn hàng tới nơi tiêu thụ với chi phí cực tiểu, lập thời khóa biểu cho giáo viên và học sinh thuận lợi nhất, định tuyến cho các gói dữ liệu trong Internet hay các bài toán trong lĩnh vực tin sinh học

1. **Các cách tiếp cận bài toán**

Với các bài toán TƯTH NP-khó có cỡ nhỏ, người ta có thể tìm lời giải tối ưu nhờ tìm kiếm vét cạn.

Tuy nhiên, với các bài toán cỡ lớn thì đến nay chưa thể có thuật toán tìm lời giải đúng với thời gian đa thức nên chỉ có thể tìm lời giải gần đúng hay đủ tốt.

Theo cách tiếp cận truyền thống hay là tiếp cận cứng, các thuật toán gần đúng phải được chứng minh tính hội tụ hoặc ước lượng được tỷ lệ tối ưu. Với việc đòi hỏi khắt khe về toán học như vậylàm hạn chế số lượng các thuật toán công bố, khôngđáp ứng được nhu cầu ngày càng phong phú và đa dạng trong nghiên cứu và ứng dụng. Để khắc phục tình trạng này, người ta dùng tiếp cận đủ tốt để xây dựng các thuật toán tối ưu mềm.

**Chương 2:** Bài toán người bán hàng (TSP)

1. **Phát biểu bài toán**

Bài toán người bán hàng (Traveling Salesman Problem - TSP) là bài toán có nhiều ứng dụng trong thực tế, được phát biểu như sau: một người giới thiệu sản phẩm muốn tìm một thành phố mà khách hàng cần giới thiệu sản phẩm và sau đó trở về thành phố xuất phát với điều kiện các thành phố của khách hàng chỉ được đi qua đúng một lần. Bài toán TSP thuộc loại NP-Khó và được xem là bài toán chuẩn để đánh giá hiệu quả của các thuật toán giải quyết bài toán tối ưu tổ hợp mới.

1. **Lịch sử nghiên cứu bài toán**

Nguồn gốc của bài toán người bán hàng vẫn chưa được biết rõ. Một cuốn sổ tay dành cho người bán hàng xuất bản năm 1832 có đề cập đến bài toán này và có ví dụ cho chu trình trong nước Đức và Thụy Sĩ, nhưng không chứa bất kì nội dung toán học nào.

Bài toán người bán hàng được định nghĩa trong thế kỉ 19 bởi nhà toán học Ireland [William Rowan Hamilton](https://vi.wikipedia.org/w/index.php?title=W._R._Hamilton&action=edit&redlink=1) và nhà toán học Anh [Thomas Kirkman](https://vi.wikipedia.org/w/index.php?title=Thomas_Kirkman&action=edit&redlink=1). [Trò chơi Icosa](https://vi.wikipedia.org/w/index.php?title=Tr%C3%B2_ch%C6%A1i_Icosa&action=edit&redlink=1) của Hamilton là một trò chơi giải trí dựa trên việc tìm kiếm [chu trình Hamilton](https://vi.wikipedia.org/wiki/%C4%90%C6%B0%E1%BB%9Dng_%C4%91i_Hamilton). Trường hợp tổng quát của TSP có thể được nghiên cứu lần đầu tiên bởi các nhà toán học ở Vienna và Harvard trong những năm 1930, đặc biệt là [Karl Menger](https://vi.wikipedia.org/w/index.php?title=Karl_Menger&action=edit&redlink=1), người đã định nghĩa bài toán, xem xét thuật toán hiển nhiên nhất cho bài toán, và phát hiện ra thuật toán láng giềng gần nhất là không tối ưu.

[Hassler Whitney](https://vi.wikipedia.org/w/index.php?title=Hassler_Whitney&action=edit&redlink=1) ở [đại học Princeton](https://vi.wikipedia.org/wiki/%C4%90%E1%BA%A1i_h%E1%BB%8Dc_Princeton) đưa ra tên *bài toán người bán hàng* ngay sau đó.

Trong những năm 1950 và 1960, bài toán trở nên phổ biến trong giới nghiên cứu khoa học ở châu Âu và Mỹ. [George Dantzig](https://vi.wikipedia.org/wiki/George_Dantzig), [Delbert Ray Fulkerson](https://vi.wikipedia.org/wiki/Delbert_Ray_Fulkerson) và Selmer M. Johnson ở công ty RAND tại Santa Monica đã có đóng góp quan trọng cho bài toán này, biểu diễn bài toán dưới dạng [quy hoạch nguyên](https://vi.wikipedia.org/w/index.php?title=Quy_ho%E1%BA%A1ch_nguy%C3%AAn&action=edit&redlink=1) và đưa ra phương pháp [mặt phẳng cắt](https://vi.wikipedia.org/w/index.php?title=M%E1%BA%B7t_ph%E1%BA%B3ng_c%E1%BA%AFt&action=edit&redlink=1) để tìm ra lời giải. Với phương pháp mới này, họ đã giải được tối ưu một trường hợp có 49 thành phố bằng cách xây dựng một chu trình và chứng minh rằng không có chu trình nào ngắn hơn. Trong những thập niên tiếp theo, bài toán được nghiên cứu bởi nhiều nhà nghiên cứu trong các lĩnh vực toán học, khoa học máy tính, hóa học, vật lý, và các ngành khác.

Năm 1972, [Richard M. Karp](https://vi.wikipedia.org/w/index.php?title=Richard_M._Karp&action=edit&redlink=1) chứng minh rằng bài toán [chu trình Hamilton](https://vi.wikipedia.org/wiki/%C4%90%C6%B0%E1%BB%9Dng_%C4%91i_Hamilton) là [NP-đầy đủ](https://vi.wikipedia.org/wiki/NP-%C4%91%E1%BA%A7y_%C4%91%E1%BB%A7), kéo theo bài toán TSP cũng là [NP-đầy đủ](https://vi.wikipedia.org/wiki/NP-%C4%91%E1%BA%A7y_%C4%91%E1%BB%A7). Đây là một lý giải toán học cho sự khó khăn trong việc tìm kiếm chu trình ngắn nhất.

Một bước tiến lớn được thực hiện cuối thập niên 1970 và 1980 khi Grötschel, Padberg, Rinaldi và cộng sự đã giải được những trường hợp lên tới 2392 thành phố, sử dụng phương pháp mặt phẳng cắt và [nhánh cận](https://vi.wikipedia.org/w/index.php?title=Nh%C3%A1nh_c%E1%BA%ADn&action=edit&redlink=1).

Trong thập niên 1990, Applegate, Bixby, Chvátal, và Cook phát triển một chương trình mang tên *Concorde* giải được nhiều trường hợp có độ lớn kỉ lục hiện nay. Gerhard Reinelt xuất bản một bộ dữ liệu các trường hợp có độ khó khác nhau mang tên TSPLIB năm 1991, và nó đã được sử dụng bởi nhiều nhóm nghiên cứu để so sánh kết quả. Năm 2005, Cook và cộng sự đã giải được một trường hợp có 33810 thành phố, xuất phát từ một bài toán thiết kế vi mạch. Đây là trường hợp lớn nhất đã được giải trong TSPLIB. Trong nhiều trường hợp khác với hàng triệu thành phố, người ta đã tìm được lời giải với sai số không quá 1% so với lời giải tối ưu.

1. **Bài toán TSP và đồ thị cấu trúc**

Về phương diện toán học, bài toán TSP là một bài toán tìm chu trình Hamilton có độ dài ngắn nhất trên đồ thị đầy đủ có trọng số G = (V, E), trong đó V là tập đỉnh tương ứng với tập các thành phố, E là tập cạnh nối các thành phố với trọng số là độ dài tương ứng. Chú ý rằng nếu đồ thị không phải đầy đủ ta luôn có thể thêm các cạnh còn thiếu để nhận được một đồ thị mới G’ đầy đủ và trọng số các cạnh này đủ lớn để hành trình tối ưu trên G’ cung là hành trình tối ưu trên G. Ta ký hiệu độ dài mỗi cạnh (i, j) ∈ A là dij tương ứng với khoảng cách giữa thành phố i và thành phố j (với mọi i, j ∈ N). Trong trường hợp tổng quát là bài toán TSP được xét trên đồ thị có hướng và khoảng cách giữa cặp đỉnh i, j có thể phụ thuộc vào hướng của cạnh, khi có ít nhất một cạnh mà dij ≠ dji thì ta nói bài toán không đối xứng và kí hiệu là ATSP, ngược lại thì là bài toàn đối xứng (luôn có dij = dji ∀i, j).

Nhắc lại chu trình Hamilton là một đường đi đóng thăm tất cả các đỉnh, mỗi đỉnh đúng một lần. Mục tiêu của bài toán TSP là tìm một chu trình Hamilton trên đồ thị có độ dài ngắn nhất. Do đó, lời giải tối ưu của bài toán TSP là một hoán vị π của tập n đỉnh { 1, 2,..., n } (n = |N|) sao cho hàm độ dài f(π) là nhỏ nhất, trong đó f(π) bằng:

 (3.2)

Thông tin heuristic trên cạnh (i, j) theo truyền thống thay cho hij sẽ kí hiệu là nij và là nghịch đảo của độ dài cạnh ni, j = ∀i, j. Trong tất cả thuật toán ACO giải bài toán TSP, vết mùi được đỉnh trên các cạnh và do đó τij dùng để chỉ thông tin học tăng cường cho mở rộng trực tiếp lời giải thừ i đến j.

Khi đó đồ thị cấu trúc của bài toán là đồ thị đầy đủ G = (V, E, H, τ). Nếu xk = <u0,..., uk> là đường đi mở rộng được, tức các đỉnh ui đều khác nhau và k < n thì J(xk) = Nuk là các đỉnh mà đương xk chưa đi đến. Các thuật toán ACO cho bài toán TSP đã có đều thực hiệu trên đồ thị cấu trúc này. Các kết quả thực ngiệm đã chỉ ra hiệu quả nổi trội của phương pháp này so với các tiếp cận khác.

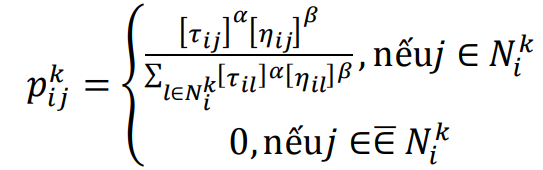
1. **Các thuật toán ACO cho bài toán TSP**
   1. **Hệ kiến (AS)**

Ban đầu có ba phiên bản của AS mà Dorigo đề xuất là ant-density, ant-quantity và ant-cycle. Ở phiên bản ant-density và ant-quantity, các kiến cập nật vết mùi trực tiếp lên cạnh vừa đi, còn trong phiên bản ant-cycle vết mùi được cập nhật khi tất cả các kiến đã xây dựng xong hành trình và lượng mùi được cập nhật của mỗi kiến phụ thuộc vào độ dài hành trình mà kiến tìm được. Hai thuật toán ant-dentity và ant-quantity không hiệu quả so với ant-cycle nên khi nói tới thuật toán AS người ta chỉ quan tâm đến phiên bản ant-cycle.

Hai bước chính của thuật toán AS là xây dựng lời giải của kiến và cập nhật mùi. Trong AS, một lời giải tìm được bằng phương pháp heuristic nào đó (có thể dùng phương pháp ăn tham) được dùng để xác định vết mùi khởi tạo. Giá trị vết mùi khởi tạo cho tất cả các cạnh là: τij = τi, j = ∀(i, j), trong đó m là số kiến, Cnn là độ dài lời giải tìm được của thuật toán heuristic. Lý do cho việc lựa chọn này là nếu khởi tạo vết mùi τ0 quá thấp thì việc tìm kiếm có khuynh hướng nhanh chóng hội tụ quanh những hành trình đầu tiên tìm được, dẫn đến việc tìm kiếm hướng vào vùng này và chất lượng lời giải kém, còn nếu khởi tạo vết mùi quá cao thì có thể phải mất nhiều vòng lặp để bay hơi mùi trên các cạnh tồi và thêm mùi cho các cạnh tốt để hướng việc tìm kiếm vào vùng không gian có chất lượng tốt.

**Xây dựng lời giải**

Trong AS, m con kiến nhân tạo đồng thời xây dựng lời giải. Ban đầu các kiến được đặt ngẫu nhiên trên các thành phố. Tại mỗi bước, kiến sẽ lựa chọn theo xác suất, gọi là ngẫu nhiên theo tỉ lệ (random proportional) để chọn đỉnh đến tiếp theo. Cụ thể, kiến k đang ở đỉnh i sẽ lựa chọn đỉnh j theo xác suất:

 (3.3)

Trong đó ni, j = là giá trị heuristic như đã nói ở trên, α, β là hai tham số quyết định đến sự ảnh hưởng tượng quan giữa thông tin mùi và thông tin heuristic, là các đỉnh lân cận của i mà kiến k có thể đi đến (là tập các đỉnh mà kiến k chưa đến, xác suất cho lựa chọn các đỉnh không thuộc bằng 0). Theo quy tắc ngẫu nhiên này, xác suất lựa chọn cạnh (i, j) tăng theo giá trị thông tin mùi τij và thông tin heuristic nij. Vài trò của hai tham số α, β như sau:

* Nếu α = 0 thành phố gần nhát sẽ được lựa chọn nhiều hơn, khi đó thuậ toán tương đối với thuật toán lựa chọn ngẫu nhiên theo nghịch đảo độ dài cạnh mà không có học tăng cường.
* Nếu β = 0 chỉ có thông tin học tăng cường biểu thị qua vết mùi được sử dụng mà không có thông tin heuristic.
* Nếu α > 1 thuanat toán nhanh chóng bị tắt nghẽn (tất cả các kiến sẽ lựa chọn cùng một hành trình) và lời giải tìm được hội tụ về lời giải tối ưu cục bộ

Để cài đặt mỗi kiến k sẽ duy trì một bộ nhớ Mk chứa thông tin lần lượt các thành phố mà kiến đã đi qua. Thông tin trong bộ nhớ dùng để xác định các thành phố lân cận phù hợp . Hơn nữa, thông tin trong bộ nhớ Mk giúp cho kiến tính được đồ dài hành trình Tk và dùng để xác định các cạnh được cập nhật mùi.

Liên quan đến việc xây dựng lời giải, có hai cách để thực hiện: xây dựng lời giải song song và xây dựng tuần tự. Trong cách xây dựng song song, tại mỗi bước tất cả các kiến sẽ di chuyển sang đỉnh tiếp theo, trong khi cách xây dựng tuần tự thì lần lượt từng kiến xây dựng lời giải (một kiến xây dựng xong mới đến kiến tiếp theo). Chú ý rằng trong AS, cả hai cách xây dựng này là như nhau vì không ảnh hưởng gì đến thuật toán nhưng điều này không đúng với thuật toán ACS

**Cập nhật mùi**

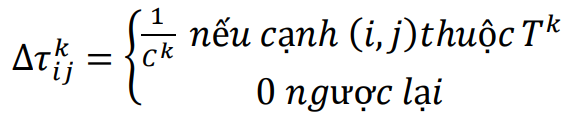
Sau khi tất cả các kiến xây dựng xong hành trình, vết mùi sẽ được cập nhật. Việc này sẽ thực hiện như sau: trước tiên tất cả các cạnh sẽ bị bay hơi theo một tỉ lệ không đổi, sau đó các cạnh có kiến đi qua sẽ được thêm một lượng mùi. Việc bay hơi mùi được thực hiện như sau:

 (3.4)

Trong đó 0 < ρ ≤ 1 là hệ số bay hơi. Tham số ρ được sử dụng để tránh sự tích tụ vết mùi quá nhiều trên một cạnh và giúp cho kiến quên đi các quyết định sai lầm. Trên thực tế, nếu một cạnh không được kiến lựa chọn vết mùi nhanh chóng bị giảm theo cấp số nhân. Sau khi bay hơi, tất cả các kiến sẽ để lại vệt mùi mà nó đi qua:

(3.5)

trong đó là lượng mùi do kiến k cập nhật trên cạnh mà kiến k đi qua. Giá trị này bằng:

(3.6)

Trong đó Ck là độ dài hành trình Tk do kiến k xây dựng, giá trị này được tính bằng tổng độ dài các cạnh thuộc hành trình. Theo công thức, các cạnh thuộc hành trình tốt hơn sẽ được cập nhật nhiều hơn. Nói chung, cạnh nào càng có nhiều kiến sử dụng và là cạnh thuộc hành trình ngắn sẽ càng được cập nhật mùi nhiều hơn và do đó sẽ được các kiến lựa chọn nhiều hơn trong các vòng lặp sau.

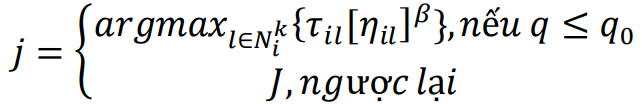
Hiệu quả của thuật toán AS so với các phương pháp metaheuristic khác có xu hướng giảm khi kích thước bài toán tăng vì vậy đã có nhiều nghiên cứu tập trung cải thiện thuật toán AS.

* 1. **Hệ đàn kiến (ACS)**

Thuật toán ACS (Dorigo & Gambardella) khác với AS ở ba điểm chính. Thứ nhất, đó là sự khai thát kinh nghiệm tìm kiếm mạnh hơn AS thông qua việc sử dụng quy tắc lựa chọn dựa trên thông tin tích lũy nhiều hơn. Thứ hai, việc bay hơi mùi và để lại mùi chỉ trên các cạnh thuộc vào lời giải tốt nhất đến lúc đó (best-so-far: G-best). Thứ ba, mỗi lần kiến đi qua cạnh (i, j) để di chuyển từ i sang j, vết mùi sẽ bị giảm trên cạnh (i, j) để tăng cường việc thăm dò đường mới. Sau đây chúng ta sẽ tìm hiểu chi tiết những thay đổi.

**Xây dựng lời giải**

Trong các thuật toán ACS, khi kiến k đang đứng ở đỉnh i lựa chọn di chuyển đến đỉnh j theo quy tắc:

(3.7)

Trong đó 𝑞 là một biến ngẫu nhiên phân bố đều trong [0, 1], 𝑞0 (0 ≤ 𝑞0 ≤ 1) là một tham số cho trước và 𝐽 là một biến ngẫu nhiên lựa chọn theo phân bố xác suất như trong (2.1) với α = 1. Hay nói cách khác, với xác suất 𝑞0 kiến lựa chọn khả năng tốt nhất có thể dựa trên kết hợp của thông tin học từ vết mùi và thông tin heuristic (trong trường hợp này kiến khai thác thông tin đã học) trong khi đó với xác suất (1 – 𝑞0) kiến thực hiện khám phá trên các cạnh. Điều chỉnh tham số 𝑞0 cho phép thay đổi mức độ khai thác và lựa chọn tập trung tìm kiếm quanh lời giải best-so-far hoặc khám phá các hành trình khác.

**Cập nhật mùi toàn cục**

Trong ACS chỉ có duy nhất một kiến tìm được lời giải tốt nhất (best-so-far) được phép để lại mùi sau mỗi lần lặp. Việc cập nhật mùi ACS được thực hiện như sau:

 (3.8)

Trong đó = , là độ dài lời giải tốt nhất, là tập các cạnh thuộc lời giải tốt nhất. Một điều quan trọng cần chú ý trong ACS là vết mùi được cập nhật bao gồm cả bay hơi và để lại mùi và chỉ cho các cạnh thuộc , không phải cho tất cả các cạnh như trong AS. Điều này rất quan trọng vì theo cách thức này thời gian cập nhật mùi cho mỗi lần lặp giảm từ O() còn O(n) trong đó n là số thành phố. Tham số ρ là tham số bay hơi. Không giống AS, trong (3.4) và (3.5) trong (3.8) vết mùi để lại giảm theo tham số ρ. Kết quả của việc cập nhật này là vết mùi được thay đổi bằng trung bình theo trọng số giữa vết mùi cũ và lượng mùi được để lại.

Trong thí nghiệm, người ta cũng sử dụng chọn lời giải tốt nhất trong bước lặp (iteration-best: i-best) để cập nhật mùi. Với các bộ dữ liệu TSP nhỏ thì việc sử dụng interation-best và best-so-far không nhiều, nhưng với dữ liệu lớn (số thành phố hơn 100) thì việc sử dụng best-so-far cho kết quả tốt hơn nhiều.

**Cập nhật mùi cục bộ**

Ngoài việc cập nhật mùi toàn cục thì ACS còn sử dụng cập nhật mùi cục bộ. Việc cập nhật mùi được thực hiện ngay lập tức khi cạnh (i, j) có kiến đi qua theo công thức:

 (3.9)

Trong đó ξ (0 < ξ < 1) và τ0 là hai tham số. Giá trị τ0 chính là giá trị khởi tạo mùi cho tất cả các cạnh. Theo kinh nghiệm giá trị tốt cho ξ bằng 0.1, giá trị τ0 là , trong đó n là số thành phố, là độ dài hành trình theo thuật toán heuristic ăn tham. Hiệu quả của thuật toán cập nhật mùi cục bộ là mỗi khi kiến sử dụng cạnh (i, j) thì vết mùi trên cạnh (i, j) bị giảm làm cho kiến ít lựa chọn lại cạnh này. Hay nói cách khác, việc cập nhật mùi cục bộ làm cho tăng cường khám phá các cạnh chưa được sử dụng. Trên thực tế, hiệu quả của cách cập nhật mùi này là thuật toán không bị tắc nghẽn (nghĩa là các kiến không bị hội tụ vào một con đường) như AS.

Một điều quan trọng cần chú ý, như đã nói ở trên, đối với AS thì việc các kiến xây dựng hành trình song song hay tuần tự là không ảnh hưởng gì, nhưng trong ACS thì lại có ảnh hưởng vì ACS có dùng cập nhật địa phương. Trong triển khai thực nghiệm, thuật toán ACS thường cho tất cả các kiến đồng thời xây dựng hành trình mặc dù không có kết quả thực nghiệm chứng tỏ kết quả nào tốt hơn.

Tồn tại một quan hệ thú vị giữa MMAS và ACS: cả hai thuật toán đề sử dụng giới hạn vệt mùi mặc dù trong ACS không rõ ràng như trong MMAS. Trên thực tế, trong ACS vết mùi không bao giờ nhở hơn τ0 bởi vì khi cập nhật theo hai công thức (3.9) và (3.8) thì vết mùi luôn lớn hơn hoặc bằng τ0 và việc khởi tạo mùi ban đầu là τ0. Hơn nữa vết mùi không bao giờ vượt quá . Do đó, vết mùi τ0­ ≤ τij ≤ .

Cuối cùng, ACS là thuật toán đầu tiên sử dụng danh sách ứng cử viên để hạn chế số lượng lựa chọn trong quá trình xây dựng lời giải. Danh sách ứng cử viên bao gồm các lựa chọn được đánh giá tốt nhất theo một số tiêu chí heuristic. Trong TSP, danh sách ứng cử viên cho mỗi thành phố i là thành phố j gần với i. Có rất nhiều cách để định nghĩa những thành phố trong danh sách ứng cử viên. Thuật toán ACO đầu tiên sắp xếp các thành phố lân cận của i theo tiêu chí tăng lần và thêm số cand cố định cho mỗi danh sách của i. Theo cách này, các danh sách ứng cử viên có thể được xây dựng trước khi bắt đầu tìm kiếm và sẽ được giữ cố định trong suốt quá trình tìm kiếm. Khi kiến đang ở đỉnh i kiến sẽ lựa chọn một trong số các ứng cử viện chưa được thăm, trong trường hợp tất cả các ứng cử viên trong thành phố điều được thăm thì chọn một thành phố chưa được thăm ngoài danh sách. Trong bài toán TSP, kết quả thực nghiệm cho thấy việc sử dụng danh sách ứng cử viên làm tăng chất lượng lời giải và làm giảm độ phức tạp.

* 1. **Hệ kiến Max-Min**

Thuật toán Max-Min Ant System – MMAS được Stutzle và Hoos đề xuất với bốn thay đổi so với AS.

Thứ nhất, để tăng cường khám phá lời giải tốt nhất tìm được: chỉ con kiến có lời giải tốt nhất tìm được trong lần lặp (i-best ant) hoặc cho đến lúc đó (G-best) được cập nhật mùi. Thật không may, điều này có thể dẫn đến tắc nghẽn, tất cả các kiến sẽ cùng đi một hành trình bởi vì lượng mùi trên các cạnh thuộc hành trình tốt được cập nhật quá nhiều, mặc dù hành trình này không phải là hành trình tối ưu.

Thứ hai, để khắc phục nhược điểm trong thay đổi thứ nhất, MMAS đưa ra miền giới hạn cho vệt mùi, vết mùi sẽ thuộc [τmin, τmax].

Thứ ba, vết mùi ban đầu được khởi tạo bằng τmax và hệ số bay hơi nhỏ nhằm tăng cường khám phá trong giai đoạn đầu.

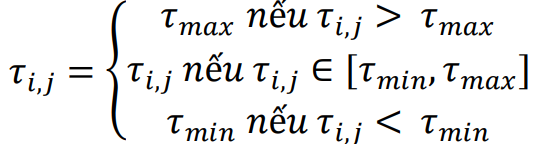
Điểm thay đổi cuối cùng là vết mùi sẽ được khởi tạo lại khi tắc nghẽn hoặc không tìm được lời giải tốt hơn trong một số bước.

**Cập nhật mùi**

Sau khi tất cả các kiến xây dựng lời giải, vết mùi được cập nhật bằng thủ tục bay hơi giống như AS (3.3), sau đó được thêm một lượng mùi như sau:

 (3.10)

Trong đó = . Kiến được lựa chọn để thêm mùi có thể là G-best (khi đó = ) hoặc iteration-best ant (khi đó = , trong đó là độ dài hành trình i-best ant). Sau đó vết mùi sẽ bị giới hạn trong đoạn [τmin, τmax] như sau:

 (3.11)

Nói chung, MMAS dùng cả i-best ant và G-best ant, thay phiên nhau. Rõ ràng, việc lựa chọn tần số tương đối cho hai cách cập nhật mùi ảnh hưởng đến hướng tìm kiếm. Nếu luôn cập nhật bằng G-best ant thì việc tìm kiếm sẽ sớm được hướng quanh , còn khi cập nhật bằng iteration-best ant thì số lượng cạnh được cập nhật mùi nhiều do đó việc tìm kiếm giảm tính định hướng hơn.

Kết quả thực nghiệm chỉ ra rằng với bộ dữ liệu TSP nhỏ đạt được kết quả tốt nhất khi chỉ sử dụng i-best để cập nhật. Trong khi với bộ dữ liệu lớn với hàng trăm thành phố, hiệu quả tốt khi tăng cường sự cập nhật bằng G-best. Điều này có thể thực hiện bằng cách tăng cường tần số sử dụng để cập nhật.

**Giới hạn vết mùi**

Trong MMAS, giới hạn trên τmaxvà giới hạn dưới τmin cho vết mùi trên tất cả các cạnh để tránh tắc nghẽn. Đạt biệt việc giới hạn vết mùi có ảnh hưởng đến giới hạn xác suất ρij trong đoạn [ρmin, ρmax] cho lựa chọn đỉnh j khi kiến đang ở đỉnh i, với 0 < ρmin ≤ ρij ≤ ρmax ≤1. Chỉ khi || = 1 thì ρmin = ρmax = 1

Dễ thấy, nếu chạy một trong thời gian dài thì vận trên của vết mùi là trong đó là độ dài hành trình tối ưu. Dựa trên kết quả đó, MMAS đặt τmax bằng , mỗi khi tìm được best-so-far tour mới τmax được cập nhật lại. Cận dưới τmin = trong đó α là một tham số. Kết quả thực nghiệm chỉ ra rằng: để tránh tắc nghẽn cận dưới τmin đóng vai trò quan trọng hơn τmax. Tuy nhiên, τmax lại hữu ích trong việc thiết đặt giá trị vết mùi khi khởi tạo lại.

**Khởi trị và khởi tạo lại vết mùi**

Khi bắt đầu thuật toán, vết mùi được đặt thiết kế bằng ước lượng cận trên của vết mùi τmax. Cách khởi tạo như vậy kết hợp với tham số bay hơi nhỏ làm chậm sự khác beiejt vết mùi của các cạnh, do đó giai đoạn đầu của MMAS mang tính khám phá.

Để tăng cường khả năng khám phá, MMAS khởi tạo lại vết mùi khi gặp tình trạng tắc nghẽn (kiểm tra sự tắt nghẽn đo được dựa trên sự thống kê vết mùi trên các cạnh) hoặc sau một số bước lặp mà không tìm được lời giải tốt hơn.

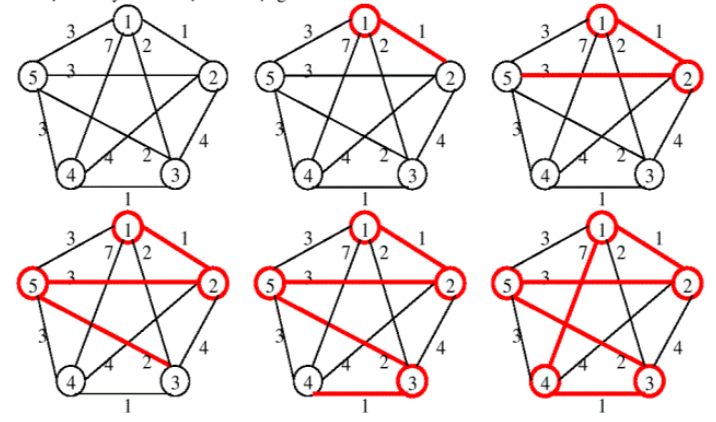
MMAS là thuật toán được nghiên cứu nhiều nhật trong các thuật toán ACO và nó có rất nhiều mở rộng. Một trong các cải tiến là khi khởi tạo lại vệt mùi, việc cập nhật mùi thường xuyên bằng lời giải tốt nhất tìm được lời giải mới nhất thay vì cố định. Một cải tiến khác sử dụng luật di truyền theo kiểu ACS

1. **Một số phương pháp khác giải bài toán**
   1. **Giải thuật chính xác**

Trong các giải thuật chính xác cho bài toán người bán hàng, phải kể đến thuật toán vét cạn. Thuật toán này tìm tất cả các chu trình Hamilton trong đồ thị, sau đó chọn một chu trình nhỏ nhất làm đáp án. Việc tìm chu trình Hamilton được thực hiện theo phương pháp duyệt chiều sâu và kết hợp quay lùi. Do quá trình duyệt có thể rất sâu nên ta không sử dụng đệ quy mà dùng stack để khử đệ quy. Sử dụng một biến min để lưu lại tổng trọng số của chu trình Hamilton nhỏ nhất.

* 1. **Giải thuật gần đúng heuristic**
     1. **Giải thuật tham lam**

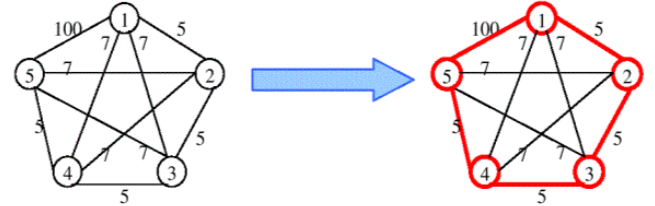
Giải thuật vét cạn ở trên cho ta một đáp án tối ưu, tuy nhiên độ phức tạp của nó quá lớn (n-1)! thuộc nhóm o(n!). Do đó, trong thực tiễn người ta chấp nhận các kết quả tốt bởi sự đơn giản, nhanh chóng và cài đặt dễ dàng. Một trong các giải thuật đó là giải thuật tham lam hay còn gọi là thuật toán láng giềng gần nhất. Trong thuật toán này, tại mỗi bước, ta chọn con đường (cạnh) đỉnh nối thành phố (đỉnh) gần nhất với hi vòng rằng n con đường ngắn nhất sẽ cho ra đường đi ngắn nhất. Ví dụ sau minh họa cho giải thuật tham lam với n=5.



Hình 2.1: Trường hợp thuật toán tham lam

Đáp án của giải thuật tham lam thường là tốt vì đáp án tối ưu là 1→3→4→5→2→1 với chi phí là 10 so với kết quả của thuật toán tham lam là 14.

Giải thuật tham làm có độ phức tạp nhỏ hơn nhiều so với giải thuật vét cạn, độ phức tạp của giải thuật này là n(n-1)/2 thuộc nhóm O(). Tuy nhiên, như đã nói ở trên, giải thuật này không phải luôn luôn đúng, trong một số trường hợp, lời giải của giải thuật tham lam rất tệ như trong ví dụ sau:



Hình 2.2: Trường hợp thuật toán tham lam

* + 1. **Thuật toán tìm kiếm cục bộ**

Thuật toán tìm kiếm cục bộ là giải pháp metaheuristic cho việc giải các bài toán tối ưu khó trong máy tính. Thuật toán này có thể được áp dụng cho các bài toán tìm kiếm lời giải gần đúng tối ưu trong một loạt các lời giải ứng viên. Phương pháp tìm kiếm sẽ duyệt qua các lời giải trong không gian tìm kiếm cho đến khi tìm ra lời giải tối ưu hơn hoắc vượt quá thời gian tìm kiếm cho phép. Thuật toán tìm kiếm cục bộ sẽ bắt đầu từ một ứng viên lời giải (chưa tối ưu), kiểm tra và cải thiện dần bằng cách chỉ quan tâm tới giải pháp hiện thời rồi xem xét chuyển sang ứng viên lời giải láng giềng, nên mỗi cách lựa chọn lời giải láng giềng trong danh sách láng giềng để thành bước duyệt kế tiếp có thể trở thành một thuật toán.

Trong bài tiểu luận này tôi có áp dụng thuật toán tìm kiếm cục bộ trong thuật toán ACO giải bài toán TSP để nâng cao hiệu quả của thuật toán ACO được trình bày trong chương thực nghiệm.

* + 1. **Thuật toán nhánh cận**

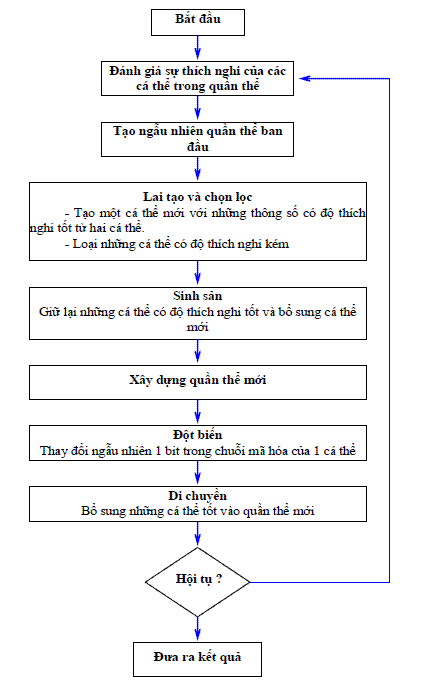
Thuật toán nhánh cận là phương pháp chủ yếu để giải các bài toán tối ưu tổ hợp. Tư tưởng cơ bản của thuật toán là trong quá trình tìm kiếm lời giải, ta sẽ phân hoạch tập các phương án thành hai hay nhiều nhánh con như cây tìm kiếm, sau đó đánh giá từng nhánh để quyết định xem nên tiếp tục ở nhánh nào và loại bỏ nhánh mà các phương án con của nó thể là phương án tối ưu.

* + 1. **Thuật toán di truyền**

Thuật toán di truyền (GA-Genetic Algorithm) là kỹ thuật phỏng theo quá trình thích nghi tiến hóa của các quần thể sinh học dựa trên học thuyết Darwin. GA là phương pháp tìm kiếm tối ưu ngẫu nhiên bằng mô phỏng theo sự tiến hóa của con người hay của sinh vật. Tư tưởng của thuật toán di truyền là mô phỏng các hiện tượng tư nhiên, là kế thừa và đấu tranh sinh tồn.

GA thuộc các phần tử tìm kiếm trực tiếp và ngẫu nhiên. Khác biệt quan trọng giữa tìm kiếm của GA và các phương pháp tìm kiếm khác là GA duy trì và xử lý một tập các lời giải, gọi là một quần thể (population). Trong GA, việc tìm kiếm giả thuyết thích hợp được bắt đầu với một quần thể, hay một tập hợp có chọn lọc babn đầu của các giả thuyết. Các cá thể của quần thể hiện tại khởi nguồn cho quần thể của thế hệ kế tiếp bằng cách hoạt động lai ghép và tiến hóa ngẫu nhiên - được lấy mẫu sau quá trình tiến hóa sinh học. Ở mỗi bước, các giải thuyết trong quần thể hiện tại được ước lượng liên hệ với đại lượng thích nghi, với các giả thuyết phù hợp nhất được chọn theo xác suất là các hạt giống cho việc sản sinh thế hệ kế tiếp, gọi là cá thể (individual). Cá thể nào phát triển hơn, thích ứng hơn với môi trường sẽ tồn tại và ngược lại sẽ bị đào thải. GA có thể dò tìm thế hệ mới có độ thích nghi tốt hơn. GA giải quyết các bài toán quy hoạch toán học thông qua các quá trình cơ bản: lai tạo (crossover), đột biến (mutation) và chọn lọc (selection) cho các cá thể trong quần thể. Dùng GA đòi hỏi phải xác định được: khởi tạo quần thể ban đầu, hàm đánh giá các lời giải theo mức độ thích nghi – hàm mục tiêu, các toán tử di truyền tạo hàm sinh sản.

Thuật toán di truyền đã và đang được ứng dụng để giải quyết các bài toán trong rất nhiều lĩnh vực của cuộc sống cũng như trong kĩ thuật.



Hình 2.3: Sơ đồ thuật toán di truyền

* + 1. **Thuật toán bầy ong**

Trong tự nhiên bầy ong mật tìm kiếm thức ăn theo quy trình sau: đầu tiên các ong do thám sẽ cử đi thăm dò các vùng thức ăn tiềm năng; các ong do thám sẽ di chuyển ngẫu nhiên từ bụi hoa này sang bụi hoa khác; sau đó chúng quay về tổ và thông tin cho cả bầy ong về hướng di chuyển, khoảng cách từ tổ đến vùng thức ăn và chất lượng của các vùng thức ăn; những thông tin này sẽ làm cho bầy ong bay tới nguồn thức ăn nhanh chóng và chính xác hơn. Một vấn đề tự nhiên nhưng là điểm trọng tâm cho ý tưởng thuật toán bầy ong là: Nơi nào có thức ăn dồi dào hơn nơi đó sẽ có nhiều ong được cử đến hơn.

Các thuật toán mô phỏng theo quá trình tìm kiếm thức ăn của loài ong mật (gọi chung là các thuật toán bầy ong) đã được biết đến với các đề xuất ban đầu của nhóm tác giả Desan Teodorovic (2010) và hiện được nhiều nhóm tác giả phát triển. Thuật toán bầy ong sử dụng hai chiến lược trọng tâm là tìm kiếm lân cận và tìm kiếm ngẫu nhiên. Thuật toán bầy ong được cho là có khả năng thoát khỏi tối ưu cục bộ và do đó có thể tìm được lời giải tối ưu toàn cục; do đó nó được đánh giá là một trong những thuật toán metaheuristic thích hợp cho việc giải các bài toán tối ư tổ hợp khó. Sơ đồ thuật toán bầy ong được mô tả như sau:

**Thuật toán bầy ong**

1. Khởi tạo quần thể ban đầu với n vùng; mỗi vùng chứa duy nhất một cá thể. Các cá thể này được chọn ngẫu nhiên hoặc bằng một thuật toán heuristic.
2. Đánh giá độ thích nghi cho mỗi cá thể của quần thể.
3. While (điều kiện dừng chưa thỏa)
4. Trong n vùng ban đầu, chọn ngẫu nhiên p vùng để thực hiện tìm kiếm lân cận (p < n) và trong p vùng này chọn tiếp ra e vùng có độ thích nghi cao nhất.
5. Mỗi vùng trong e vùng sẽ được tuyển thêm nep ong để tìm lân cận quanh nó.
6. Mỗi vùng trong p – e vùng được chọn sẽ được tuyển thêm nsp ong để tìm lân cận (trong đó nep > nsp).
7. Mỗi ong trong n – p vùng còn lại được thay thế bằng một ong ngẫu nhiên khác.
8. Đánh giá độ thích nghi cho tất cả các ong ở mỗi vùng.
9. Mỗi vùng chọn ra duy nhất một ong có độ thích nghi cao nhất.
10. **Ứng dụng của bài toán**

Ngoài việc là một "polytope" của một vấn đề tối ưu hóa tổ hợp khó khăn từ một phức tạp điểm lý thuyết của xem, có những trường hợp quan trọng của các vấn đề thực tế có thể được xây dựng như các vấn đề TSP và nhiều vấn đề khác là những khái quát của vấn đề này. Bên cạnh việc khoan mạch in bảng mô tả ở trên, vấn đề có cấu trúc TSP xảy ra trong phân tích cấu trúc của các tinh thể, (Bland và Shallcross, 1987), các đại tu động cơ tuốc bin khí (Pante, Lowe và Chandrasekaran, 1987), trong xử lý vật liệu trong một nhà kho (Ratliff và Rosenthal, 1981), trong việc cắt giảm các vấn đề chứng khoán, (Garfinkel, 1977), các phân nhóm của các mảng dữ liệu, (Lenstra và Rinooy Kạn, 1975), trình tự các công việc trên một máy tính duy nhất (và Gilmore Gomory, 1964) và phân công các tuyến đường cho máy bay của một hạm đội quy định (Boland, Jones, và Nemhauser, 1994). Biến thể có liên quan về vấn đề nhân viên bán hàng đi du lịch bao gồm các nguồn tài nguyên hạn chế đi du lịch vấn đề nhân viên bán hàng trong đó có các ứng dụng trong lập kế hoạch với thời hạn tổng hợp (Pekny và Miller, 1990). Nghiên cứu này cũng cho thấy giải thưởng thu thập đi vấn đề nhân viên bán hàng (Balas, 1989) và các vấn đề Orienteering (Golden, Levy và Vohra, 1987) là trường hợp đặc biệt của tài nguyên hạn chế TSP. Quan trọng nhất là vấn đề nhân viên bán hàng đi du lịch thường thể hiện như một bài toán con trong nhiều vấn đề tổ hợp phức tạp, là nổi tiếng và quan trọng nhất trong số đó là vấn đề định tuyến xe, có nghĩa là, vấn đề xác định cho một đội xe mà khách hàng sẽ được phục vụ bởi mỗi chiếc xe và theo thứ tự mỗi chiếc xe nên đến các khách hàng được giao.

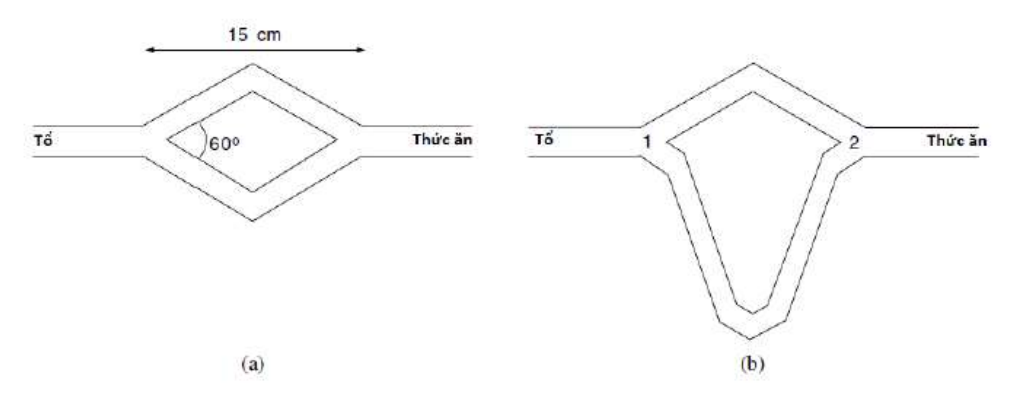
1. **Kết luận chương**

Chương này trình bày tổng quan các lý thuyết liên quan về bài toán người bán hàng, các phương pháp giải bài toán người bán hàng. Trong các phương pháp giải bài toán người bán hàng đã trình bày thuật toán tối ưu hóa đàn kiến được lựa chọn để trình bày trong tiểu luận này. Nội dung chi tiết của thuật toán tối ưu hóa đàn kiến sẽ được trình bày trong chương 3.

**Chương 3:** Thuật toán tối ưu hóa đàn kiến

1. **Ý tưởng của bài toán**

Năm 1989, nhà bác học người Đan Mạch Deneubourg và các cộng sự công bố kết quả nghiên cứu thí nghiệm trên đàn kiến Argentina (Một loài kiến hiếm trên thế giới), gọi là thí nghiệm “chiếc cầu đôi”

Cụ thể, họ đã đặt một chiếc cầu đôi gồn hai nhánh (Nhánh dài hơn có độ dài bằng hai lần nhánh ngắn hơn, như hình bên) nối tổ của đàn kiến với nguồn thức ăn, sau đó thả một đàn kiến và bắt đầu quan sát hoạt động của chúng trong một khoảng thời gian đủ lớn. Kết quả là ban đầu các con kiến đi theo cả hai nhánh của chiếc cầu với số lượng gần như ngang nhau, nhưng càng về cuối thời gian quan sát người ta nhận thấy các con kiến có su hướng chọn nhánh ngắn hơn để đi (80%-100% số lượng) 

*Hình 3.1: Thí nghiệm trên chiếc cầu đôi*

Kết quả được các nhà sinh học lý giải như sau: Do đặt tính tự nhiên và đặt tính hóa học, mỗi con kiến khi di chuyển sẽ luôn để lại một lượng hóa chất gọi là các vệt mùi (pheromone trail) trên đường đi và thường chúng sẽ đi theo con đường có mùi đậm đặt hơn. Các vệt mùi này là các hóa chất bay hơi theo thời gian, do vậy thì ban đầu mùi ở hai nhánh sấp sỉ như nhau khi phân bố ở nhánh dài hơn mật độ phân bố mùi ở nhánh này sẽ không dày bằng nhánh ngắn hơn, thêm nữa lượng mùi trên nhánh dài hơn sẽ bay hơi nhanh hơn trong cùng một khoảng thời gian.

Năm 1991, với cơ sở kết quả của thí nghiệm nổi tiếng trên, nhà khoa học người Bỉ Marco Dorigo đã xây dựng thuật toán đàn kiến (Ant algorithm, hay còn gọi là hệ kiến, Ant system) đầu tiên ứng dụng vào bài toán người du lịch, và công bố trong luận án tiến sĩ của ông. Trong bài báo này các tiến sĩ muốn giới thiệu về thuật toán cơ bản Ant-Cyle (Thuật toán nổi tiến và hiệu quả nhấ trong các thuật toán Hệ kiến) được công bố năm 1996 trên tạp chí lý thuyết của IEEE (Institute of Electrical and Electronics Engineers, là hiệp hội nghiên cứu công nghệ và khoa học hàng đầu thế giới).

Mô phỏng kiến tự nhiên, người ta dùng đa tác tử (multiagent) làm đàn kiến nhân tạo, trong đó mỗi con kiến có nhiều khả năng hơn kiến tự nhiên. Mỗi con kiến nhân tạo (về sau gọi là kiến) có bộ nhớ riêng, có khả năng ghi nhớ các đỉnh đã thăm trong hành trình và tính đường đi nó chọn. Ngoài ra các con kiến có thể trao đổi thông tin chúng có được với nhau, thực hiện tính toán cần thiết, cập nhật mùi.

Nhờ các con kiến nhân tạo này (về sau gọi đơn giản là kiến) Dorigo (1991) đã xây dựng hệ kiến (AS) giải bài toán người bán hàng, hiệu quả của nó so với các phương pháp mô phỏng tự nhiên khác như SA, GA đã được kiểm chứng bằng thực nghiệm và phát triển, ứng dụng phong phú với tên gọi chung là ACO

1. **Quá trình phát triển**

Thuật toán Ant System (AS) là thuật toán đầu tiên trong lớp thuật toán ACO được đề xuất bởi Dorigo trong luận án tiến sĩ của ông năm 1991. Thuật toán AS hướng tới bài toán tìm đường đi tối ưu trong đồ thị. Mặt dù thuật toán AS vẫn còn thua kém các thuật toán tốt nhất trong việc giải quyết bài toán trên, tuy nhiên ý tưởng của nó tỏ ra mới mẻ và có triển vọng. Về sau đã có nhiều cải tiến của thuật toán này do Dorigo đề xuất, cũng như rất nhiều thuật toán ACO khác điều dựa trên ý tưởng của thuật toán AS song đã khắc phục được một số nhược điểm của thuật toán này. Có thể kể tên hai cải tiến nổi trội nhất của thuật toán AS là thuật toán ACS và thuật toán MMAS.

|  |  |
| --- | --- |
| ACO algorithms | Tác giả |
| Ant System | Dorigo Maniezzo, & Colorni (1991) |
| Elitist AS | Dorigo (1992); Dorigo, Maniezzo & Colorni (1996) |
| Ant - Q | Gambardella & Dorigo (1995); Dorigo & Gambardella (1996) |
| Ant Colony System | Dorigo & Gambardella (1996) |
| Max - Min AS | Stutzle & Hoos (1996, 2000); Stutzle (1999) |
| Rank - based AS | Bullnheimer, Hartl, & Strauss (1997, 1999) |
| ANTS | Maniezzo (1999) |

*Bảng 3.1: Thuật toán ACO theo thứ tự thời gian xuất hiện*

1. **Thuật toán ACO cho bài toán TƯTH**
   1. **Đồ thị cấu trúc**

Xét bài toán TƯTH tổ hợp tổng quát được nêu trong chương 1 dưới dạng bài toán cực tiểu hóa (𝑆, 𝑓, Ω) trong đó S là tập hữu hạn trạng thái, 𝑓 là hàm mục tiêu xác định trên S, Ω là các ràng buột để xác định S qua qua các thành phần của tập hữu hạn C và các liên kết của tập này. Các tập S, C và Ω có các đặt tính đã nêu trong chương 1.

Như đã nói mỗi bài toán tối ưu tổ hợp là một bài toán tìm kiếm vecto độ dài không quá h trên đồ thị đầy đủ, các đỉnh có nhãn trên tập C. Để tìm lời giải chấp nhận được, ta xây dựng đồ thị đầy đủ với tập đỉnh V mà mỗi đỉnh của nó tương ứng với mỗi thành phần của C. Các lời giải chấp nhận được là các vecto xây dựng tuần tự theo thủ tục bước ngẫu nhiên.

Thông thường đối với bài toán thuộc loại NP-Khó, người ta có các phương pháp heuristic để tìm lời giải đủ tốt cho bài toán. Các thuật toán ACO kết hợp thông tin heuristic này với các phương pháp học tăng cường nhờ mô phỏng hành vi của đàn kiến để tìm lời giải tốt hơn.

Giả sử với mỗi cạnh nối các đỉnh i,j ∈ C có trọng số heuristic hi,j để định hướng chọn thành phần mở rộng là j khi thành phần cuối của xk là j theo thủ tục tuần tự (hi,j > 0 ∀(i,j)). Ký hiệu H là vecto các trọng số heuristic của cạnh tương ứng (trong bài toán TSP nó có thể là vecto thành phần là nghịch đảo độ dài của cạnh tương ứng), còn là vecto là vecto biểu thị các thông tin học tăng cường (về sau gọi là vệt mùi, ban đầu được khởi tạo bằng > 0) định hướng mở rộng xk với thành phần cuối là i nhờ thêm thành phần j theo thủ tục tuần tự. Trường hợp đặt biệt, hi,j và chỉ phụ thuộc vào j thì các thông tin này chỉ để ở các đỉnh tương ứng,. Không giảm tổng quát, ta sẽ xét cho trường hợp thông tin này ở các cạnh.

Khi đó ta gọi đồ thị G = (V, E, H, ) là đồ thị cấu trúc của bài toán tối ưu tổ hợp đang xét, trong đó V là tập đỉnh, H và là các thông tin đã nói ở trên còn E là tập cạnh của đồ thị sao cho từ các cạnh này có thể xây dựng được tập X\* nhờ mở rộng tập C0 theo thủ tục tuần tự. Nếu không có thông tin heuristic thì ta xem H có các thành phần như nhau và bằng 1

* 1. **Mô tả thuật toán ACO tổng quát**

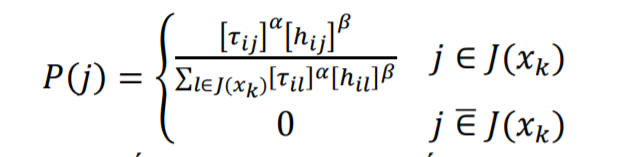
Với điều kiện kết thúc đã chọn (có thể là số bước lặp hoặc số lần chạy cho trước), người ta dùng đàn kiến m con thực hiện lặp xây dựng lời giải trên đồ thị cấu trúc G = (V, E, H, ) như sau. Trong mỗi lần lặp, mỗi con kiến chọn ngẫu nhiên 1 đỉnh u0 ∈ C0 làm thành phần khởi tạo x0 = { u0 } và thực hiện xây dựng lời giải theo thủ tục bước ngẫu nhiên để xây dựng lời giải. Dựa trên lời giải tìm được đàn kiến sẽ thực hiện cập nhật mùi theo cách học tăng cường.

**Lượt đồ chung phương pháp ACO**

|  |
| --- |
| **Produce** 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;  Cải tiến lời giải do kiến k xây dựng bằng tìm kiế cục bộ;  **end-for**  Cập nhật lời giải tốt nhất;  Cập nhật mùi;  **until;**  Đưa ra lời giải tốt nhất;  **End;** |

**Thủ tục bước ngẫu nhiên xây dựng lời giải**

Giả sử xk = <u0,.., uk> là mở rộng được, từ các ràng buộc Ω xác định được tập con J(xk) của C sao cho với mọi uk+1 ∈ J(xk) thì xk+1 = <u0,.., uk, uk+1> là mở rộng được hoặc xk ∈ X\*khi J(xk) rỗng. Đỉnh j = uk+1 để mở rộng được chọn với xác xuất P(j) như sau:

(3.1)

Quá trình mở rộng tiếp tục cho tới khi kiến r tìm được lời giải chập nhận được x(r) trong X\* và do đó s(r) = φ(x(r)) ∈ S.

Để tiện trình bày, về sau ta sẽ xem x(r) và s(r) như nhau và không phân biệt X\* với S.

**Cập nhật mùi**

Tùy theo chất lượng của lời giải tìm được mà vết mùi trên mỗi cạnh sẽ được điều chỉnh tăng hoặc giảm tùy theo đánh giá mức độ ưu tiên tìm kiếm về sau. Vì vậy quy tắc cập nhật mùi được dùng làm tên gọi thuật toán và thường có dạng:



Có nhiều quy tắc cập nhật mùi được đề xuất, trong đó điển hình là các quy tắc hệ kiến (AS), hệ đàn kiến (ACS), hệ kiến Max-Min (MMAS)

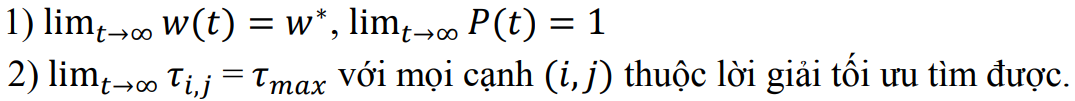
1. **Tìm kiếm cục bộ**

Nhiều tài liệu chỉ ra rằng với các phương pháp metaheuristic, một cách tiếp cận đầy hứa hẹn để thu được lời giải có chất lượng cao là kết hợp với thuật toán tìm kiếm cục bộ.

Mô hình ACO có thể bao gồm cả tìm kiếm cục bộ. Sau khi kiến xây dựng xong lời giải, có thể áp dụng tìm kiếm cục bộ để nhận được lời giải tối ưu địa phương. Việc cập nhật mùi được thực hiện trên các cạnh thuộc lời giải tối ưu địa phương. Việc kết hợp xây dựng lời giải với tìm kiếm cục bộ là một các tiếp cận đầy hứa hẹn. Trên thực tế, bởi vì cách xây dựng lời giải của ACO sử dụng lân cận khác với tìm kiếm cục bộ. Thực nghiệm cho thấy khải năng kết hợp tìm kiếm cục bộ cải tiến được lời giải là khá cao.

1. **Một số vấn đề khác khi áp dụng ACO**

Gutjahr khởi đàu cho nghiêm cứu đặc tính hội tụ của thuật toán MMAS không có thông tin heuristic. Ký hiệu P(t) là xác suất tìm thấy lời giải của thuật toán MMAS trong vòng t phép lặp, w(t) là lời giải tốt nhất ở bước lặp t. Nhờ sử dụng mô hình Markov không thuần nhất, Gutjahr đã chứng minh răng với xác suất bằng 1 ta có:



Mô hình này của Gutjahr không áp dụng được cho ACS. Trường hợp MMAS không có thông tin heuristic, Stutzle và Dorigo đã chứng minh rằng:





Các tác giả cũng suy luận rằng kết quả này cũng đúng cho ACS. Với giả thiết đã tìm được lời giải tối ưu sau hữu hạn bước, Stutzle và Dorigo suy ra rằng vết mùi trên các cạnh không thuộc lời giải này hội tụ về τmin hoặc τ0.

1. **Nhận xét thuật toán**

Nhờ kết hợp các thông tin heuristic, thông tin học tăng cường và mô phỏng hoạt động của đàn kiến, các thuật toán ACO có các ưu điểm như sau:

* Việc tìm kiếm ngẫu nhiên dựa trên các thông tin heuristic làm cho phép tìm kiếm linh hoạt và mềm dẻo trên miền rộng hơn phương pháp heuristic có sẳn, do đó cho ta lời giải tốt hơn và có thể tìm được lời giải tối ưu.
* Sự kết hợp học tăng cường thông qua thông tin về cường độ vết mùi cho phép ta từng bước thu hẹp không gian tìm kiếm mà vẫn không loại bỏ các lời giải tốt, do đó nâng cao chất lượng thuật toán.

Thuật toán ACO dễ song song hóa để giảm thời gian chạy trên máy song song do mỗi con kiến tìm lời giải một cách độc lập trong mỗi vòng lặp.

1. **Các yếu tố quyết định hiệu quả của thuật toán**

Có ba yếu tố quyết định hiệu quả của thuật toán

* 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 tuần tự không khó. Khó khăng chính đối với bài toán cỡ lớn thì thông tin tìm kiếm quá rộng, đòi hỏi ta sử dụng các ràng buộc Ω một cách hợp lý để giảm miền tìm kiếm cho mỗi con kiến.
* 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 đặt tính của học tăng cường và thuật toán vẫn được thực hiện.
* Chọn quy tắc cập nhật mùi: Quy tắc cập nhật mùi thể hiện chiến lượt 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 quy tắc phổ dụng và thường dùng để đặt tên cho thuật toán. Nó cũng đã được sử

1. **Ứng dụng của thuật toán**

Các thuật toán tối ưu hóa đàn kiến được áp dụng trong nhiều tổ hợp các vấn đề về tối ưu hóa, từ bật hai đến protein gấp hoặc phương tiện trực tuyến và rất nhiều phương pháp dẫn xuất đã được điều chỉnh để phù hợp với các bài toán động trong biến thực, bài toán ngẫu nhiên, đa mục tiêu và triển khai song song. Nó cũng đã được sử dụng để giải bài toán TSP. Thuật toán đàn kiến có thể được chạy liên tục và thích ứng với những thay đổi trong thời gian thực. Điều này được quan tâm trong định tuyến mạng và hệ thống giao thông đô thị.

1. **Các điểm hạn chế của ACO**

Khi một cạnh do ngẫu nhiên mà không được cập nhật mùi sau một số bước thì cường độ mùi của nó nhanh chóng bị giảm xuống và khó được các con kiến chọn sau đó mặc dù “chất lượng” của nó chưa chắc đã là xấu.

Nếu khởi tạo mùi như nhau và không dùng thông tin heuristic thì xác suất của mỗi cạnh được mỗi con kiến đã cho sử dụng trong lần lặp đầu là , xác suất này rất bé khi n lớn. Như vậy tùy theo từng loại bài toán mà tỉ lệ giữa τ0 và τ1 rất có ý nghĩa để cân bằng giữa tính khám phá và khai thát của thuật toán.

Các lượng mùi cập nhật của ACS và MMAS phụ thuộc vào giá trị hàm mục tiêu của lời giải mà các con kiến xây dựng được trong các bước lặp. Việc xác định các giá trị τ0, τ1 hay τmin, τmax cũng phụ thuộc vào tương quan với các giá trị chưa xác định trước này của từng bài toán thì thuật toán mới tốt được.

1. So sánh thuật toán ACO với một số thuật toán khác

## **Kết luận chương**

Thuật toán ACO ngày càng phong phú, không chỉ có 3 thuật toán ACO như đã trình bày trong tiểu luận này mà còn được phát triển thành nhiều phiên bản với nhiều tên gọi khác nhau, tuy nhiên vẫn mang những tư tưởng cốt lõi của thuật toán AS. Chương 4 sẽ trình bày cách thực thi thuật toán ACO cho bài toán người bán hàng.

**Chương 4:** Thuật toán tối ưu hóa đàn kiến giải bài toán người bán hàng

1. **Cơ sở lý thuyết**
2. **Phân tích yêu cầu**

Xây dựng chương trình minh họa thuật toán tối ưu hóa đàn kiến ACO cho bài toán người bán hàng với dữ liệu thực nghiệm được lấy trong thư viện TSPLib

Dự kiến kết quả của chương trình sẽ là:

* Xuất ra đường đi ngắn nhất xuất phát từ một đỉnh bất kì đi qua tất cả các thành phố mỗi thành phố 1 lần.
* Lưu kết quả chạy chương trình vào 1 folder
* So sánh kết quả chạy chương trình bằng thuật toán tối ưu hóa đàn kiến với kết quả của chương trình trong thư viện TSPLib

1. **Thuật toán ACO giải bài toán TSP**

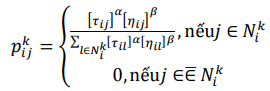
Giải thuật ACO là một lớp các giải thuật dựa trên ý tưởng mô phỏng cách tìm đường đi của các con kiến tự nhiên đến tổ thức ăn của chúng. Các giải thuật ACO cho bài toán TSP được mô tả trong mục 4 chương 3. Trong bài tiểu luận này tôi sẽ áp dụng giải thuật Max-Min ant system, là lớp giải thuật ACO để giải bài toán TSP. Sự khác biệt của giữa thuật toán MMAS so với hai thuật toán AS và ACS là cách thức cập nhật vết mùi trên đường đi của các kiến.

* 1. **Khởi tạo lời giải ban đầu**

Khởi tạo số kiến m, ma trận khoảng cách các thành phố, điểm xuất phát, khởi tạo mùi.

* 1. **Quy tắc chuyển trạng thái**

Quy tắc chuyển trạng thái của mỗi con kiến di chuyển từ thành phố i đến thành phố j dựa trên công thức:

 (4.1)

Trong đó:

: xác suất kiến k chọn đỉnh j từ đỉnh i hiện tại.

τij : thông tin vết mùi trên đoạn (i, j).

ηij = : giá trị heuristic.

dij : cạnh.

α, β : hai tham số quyết định sự ảnh hưởng tương quan giữa thông tin mùi và thông tin heuristic

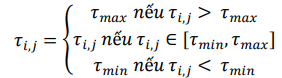
: các đỉnh lân cận đỉnh i mà kiến k có thể đi đến.

* 1. **Quy tắc cập nhật mùi**

Sau khi mỗi con kiến điều xây dựng xong lời giải ở mỗi bước lặp, vệt mùi được thay đổi theo công thức:

τij (1 - ρ)τij + (4.2)

Sau đó vệt mùi sẽ được giới hạn trong đoạn [τmin, τmax] như sau:

 (4.3)

Trong đó:

ρ : hệ số bay hơi.

= : nồng độ vệt mùi do kiến để lại.

τmax = : giới hạn trên, được cập nhật lại mỗi khi tìm dc best-so-far tour.

τmin = : giới hạn dưới, α là một tham số.

τ0 = τmax : vết mùi ban đầu.

: độ dài lời giải tốt nhất.

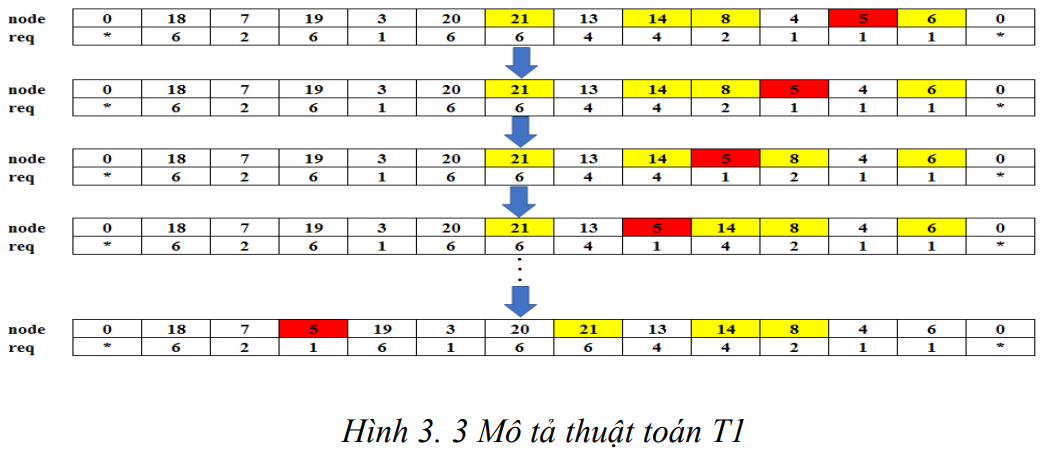
* 1. **Khởi trị và khởi tạo lại vệt mùi**

Để tăng cường khả năng khám phá , MMAS khởi tạo lại vệt mùi khi gặp tình trạng tắt nghẽn (kiểm tra tình trạng tắc nghẽn đo được dựa trên sự thống kê vết mùi trên các cạnh) hoặc sau một số bước lặp mà không tìm được lời giải tốt hơn.

* 1. **Thủ tục tìm kiếm cục bộ**

Để tăng cường chất lượng cho lời giải tối ưu ta có thủ tục tìm kiếm cục bộ. Tôi đề xuất hai thuật toán heuristic là T1 và T2 được áp dụng cho bài toán TSP. Trong đó:

T1: được xây dựng dựa trên thuật toán tham lam nhằm giảm khoảng cách di chuyển cho tuyến đường đã định tuyến. T1 được thực hiện bằng cách đổi ngẫu nhiên vị trí của một số nút đón trong cùng một tuyến đường nhằm mục đính tạo ra tuyến đường mới tốt hơn (nếu có)



*Hình 4.1 Mô tả thuật toán T1*

T2: được xây dựng tương tự như T1 nhưng thay vì đổi vị trí các nút đón trong một tuyến đường, giải thuật T2 sẽ thay đổi thuộc router i cho router j bất kỳ trong lời giải tại vị trí tốt nhất sao cho tổng chi phí là tối thiểu. Trong hình 4.1, chúng ta có thể thấy phương án thứ 2 đã được xây dựng bằng cách đổi vị trí request 7 (bao gồm các nút 23, 22, 24) từ router 2 sang router 3. Và phương án 3 được xây dựng bằng các đổi vị trí request 5 (bao gồm các nút 15, 16, 17) từ router 2 sang router 3.



*Hình 4.2 Mô tả thuật toán T2*

1. **Kết luận chương**

Trong bài này tôi đã giới thiệu thuật toán MMAS giải bài toán người bán hàng

**Chương 5:** Thực nghiệm và đánh giá

1. **Dữ liệu thực nghiệm**

Việc lựa chon dự liệu để thực nghiệm rất quan trọng giúp đánh giá được hiệu quả của thuật toán. Bộ dữ liệu sử dụng được lấy từ TSPLib gồm 26 bộ với các định dạng là ma trận chi phí và khoảng cánh Euclide.

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Tên | Số thành phố | Chi phí thấp nhât theo web |
| 1 | five | 5 | 19 |
| 2 | p01 | 15 | 291 |
| 3 | gr17 | 17 | 2085 |
| 4 | fri26 | 26 | 937 |
| 5 | dantzig42 | 42 | 699 |
| 6 | att48 | 48 | 33523 |
| 7 | st70 | 70 | 675 |
| 8 | pr76 | 76 | 108159 |
| 9 | lin105 | 105 | 14379 |
| 10 | xqf131 | 131 | 564 |
| 11 | xqg237 | 237 | 1019 |
| 12 | pma343 | 343 | 1368 |
| 13 | pak379 | 379 | 1332 |
| 14 | bcl380 | 380 | 1621 |
| 15 | pbl395 | 395 | 1281 |
| 16 | pbk411 | 411 | 1343 |
| 17 | pbn423 | 423 | 1365 |
| 18 | pbm436 | 436 | 1443 |
| 19 | xql662 | 662 | 2513 |
| 20 | rbx711 | 711 | 3115 |
| 21 | rbu737 | 737 | 3314 |
| 22 | dkg813 | 813 | 3199 |
| 23 | lim963 | 963 | 2789 |
| 24 | pbd984 | 984 | 2797 |
| 25 | xit1083 | 1083 | 3558 |
| 26 | rby1599 | 1599 | 5533 |
| 27 | dcb2086 | 2086 | 6600 |

*Bảng 5.1: Bộ dữ liệu chạy thử nghiệm*

1. **Môi trường thực nghiệm**

Thuật toán được thực nghiệm bằng phần mềm do chính tôi viết trên một máy tính Core i5-8350U 1.7Hz với 16GB RAM. Để đanh giá hiệu quả của giải thuật ACO cho bài toán TSP, tôi đã sử dụng các đồ thị liên kết đầy đủ (complete graph). Ma trận kề của đồ thị được xây dựng dựa trên khoảng cách Euclide của tọa độ các đỉnh.

1. **Tham số thực nghiệm**

Để lựa chọn các tham số tối ưu cho thuật toán ACO, tôi đã sử dụng các bộ test có số thành phố từ 5 đến 1083 thành phố. Cụ thể các giá trị tham số được chọn chia làm nhiều bộ để đánh giá được hiệu quả của thuật toán:

Bộ số 1:

alpha = 0.1 :

beta = 0.3 :

ro = 0.05 :

noAnts = 25 : số lượng kiến

Bộ số 2:

alpha = 0.1 :

beta = 0.3 :

ro = 0.05 :

noAnts = 50% số đỉnh: số lượng kiến

1. **Kết quả thực nghiệm**

Việc thực nghiệm được thực hiện đối với 27 bộ dữ liệu đã cho ở trên mỗi bộ thực hiện trong 30 lần sau đó chọn ra kết quả tốt nhất trong 30 lần chạy lưu lại chi phí tốt nhất tìm được và đường đi tìm được của từng bộ dữ liệu vào trong một folder.

**Bộ số 1:** alpha = 1.0, beta = 3.0, ro = 0.5, noAnts = 25

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| STT | Tên | Số thành phố | Chi phí tìm được khi chạy MMAS | Chi phí thấp nhât theo TSPLib | Tỉ lệ % chính xác |
| 1 | five | 5 | 19 | 19 | 100% |
| 2 | p01 | 15 | 291 | 291 | 100% |
| 3 | gr17 | 17 | 2085 | 2085 | 100% |
| 4 | fri26 | 26 | 937 | 937 | 100% |
| 5 | dantzig42 | 42 | 798 | 699 | 100% |
| 6 | att48 | 48 | 38074 | 33523 | 100% |
| 7 | st70 | 70 | 830 | 675 | 100% |
| 8 | pr76 | 76 | 153462 | 108159 | 87,59% |
| 9 | lin105 | 105 | 20356 | 14379 | 88,05% |
| 10 | xqf131 | 131 | 704 | 564 | 81,33% |
| 11 | xqg237 | 237 | 1343 | 1019 | 70,48% |
| 12 | pma343 | 343 | 1799 | 1368 | 70,64% |
| 13 | pak379 | 379 | 1835 | 1332 | 80,11% |
| 14 | bcl380 | 380 | 1925 | 1621 | 75,87% |
| 15 | pbl395 | 395 | 1618 | 1281 | 76,04% |
| 16 | pbk411 | 411 | 1784 | 1343 | 72,59% |
| 17 | pbn423 | 423 | 1768 | 1365 | 84,21% |
| 18 | pbm436 | 436 | 1739 | 1443 | 79,17% |
| 19 | xql662 | 662 | 3124 | 2513 | 75,28% |
| 20 | rbx711 | 711 | 3890 | 3115 | 77,21% |
| 21 | rbu737 | 737 | 4211 | 3314 | 82,98% |
| 22 | dkg813 | 813 | 3942 | 3199 | 80,44% |
| 23 | lim963 | 963 | 3503 | 2789 | 80,08% |
| 24 | pbd984 | 984 | 3471 | 2797 | 78,7% |
| 25 | xit1083 | 1083 | 4563 | 3558 | 80,44% |
| 26 | rby1599 | 1599 | 6818 | 5533 | 80,44% |
| 27 | dcb2086 | 2086 | 8112 | 6600 | 80,44% |

*Bảng 5.2: Kết quản thực nghiệm bộ số 1*

Tỉ lệ chính xác thông qua thực nghiệm trung bình đạt 84,52%

**Bộ số 2:** alpha = 0.1, beta = 0.3, ro = 0.05, noAnts = 50% số thành phố

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| STT | Tên | Số thành phố | Chi phí tìm được khi chạy MMAS | Chi phí thấp nhât theo web | Tỉ lệ % chính xác |
| 1 | five | 5 | 19 | 19 | 100 |
| 2 | p01 | 15 | 291 | 291 | 100 |
| 3 | gr17 | 17 | 2085 | 2085 | 100 |
| 4 | fri26 | 26 | 937 | 937 | 100 |
| 5 | dantzig42 | 42 | 798 | 699 | 87.594 |
| 6 | att48 | 48 | 38074 | 33523 | 88.047 |
| 7 | st70 | 70 |  | 675 |  |
| 8 | pr76 | 76 |  | 108159 |  |
| 9 | lin105 | 105 |  | 14379 |  |
| 10 | xqf131 | 131 | 704 | 564 | 80.1136 |
| 11 | xqg237 | 237 | 1343 | 1019 | 75.8749 |
| 12 | pma343 | 343 | 1799 | 1368 | 76.0422 |
| 13 | pak379 | 379 | 1835 | 1332 | 72.5886 |
| 14 | bcl380 | 380 | 1925 | 1621 | 84.2078 |
| 15 | pbl395 | 395 | 1618 | 1281 | 79.1718 |
| 16 | pbk411 | 411 | 1784 | 1343 | 75.2803 |
| 17 | pbn423 | 423 | 1768 | 1365 | 77.2059 |
| 18 | pbm436 | 436 | 1739 | 1443 | 82.9787 |
| 19 | xql662 | 662 | 3124 | 2513 | 80.4417 |
| 20 | rbx711 | 711 | 3890 | 3115 | 80.0771 |
| 21 | rbu737 | 737 | 4211 | 3314 | 78.6986 |
| 22 | dkg813 | 813 | 3942 | 3199 | 81.1517 |
| 23 | lim963 | 963 | 3503 | 2789 | 79.6175 |
| 24 | pbd984 | 984 | 3471 | 2797 | 80.582 |
| 25 | xit1083 | 1083 | 4563 | 3558 | 77.975 |
| 26 | rby1599 | 1599 | 6818 | 5533 | 81.1528 |
| 27 | dcb2086 | 2086 | 8112 | 6600 | 81.3609 |

*Bảng 5.3:Kết quả thực nghiệm bộ số 2*

Tỉ lệ chính xác thông qua thực nghiệm trung bình đạt

1. **Đánh giá**

Vì mỗi bộ test được thực hiện 30 lần nên việc sử dụng nhiều kiến hay ít kiến không cho thấy nhiều sự khác biệt trong kết quả. Đối với những bộ test từ 70 thành phố trở xuống hiệu quả của thuật toán đạt 100% từ đó cho thấy thuật toán hiệu quả cao đối với những bộ dữ liệu TSP nhỏ. Từ 76 thành phố trở lên hiệu quả của thuật toán đạt từ 70% đến 83%. Tỉ lệ chính xác trung bình thông qua thực nghiệm đạt

# **KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN**

Khi dùng phương pháp ACO, quy tắt cập nhật mùi đóng vai trò quan trọng quyết định hiệu quả của thuật toán được dùng. Tiểu luận đã cài đặt thuật toán TSP áp dụng quy tắc cập nhật mùi của hệ kiến MMAS. Bài toán TSP thực nghiệm với dữ liệu chuẩn cho thấy thuật toán ACO có hiệu quả và dễ sử dụng.

# **TÀI LIỆU THAM KHẢO**

[1] [Marco Dorigo](https://www.google.com/search?sa=X&rlz=1C1CHBF_enVN919VN919&biw=1920&bih=947&sxsrf=ALeKk00gmloOXtx_C2QasRIUab_UljQEdQ:1620465319964&q=marco+dorigo&stick=H4sIAAAAAAAAAOPgE-LRT9c3rEwpt0zKMC9T4tTP1TcwSbMwTdeSyU620k_Kz8_WLy_KLClJzYsvzy_KtkosLcnIL1rEypObWJScr5CSX5SZnr-DlREAk_wZ2EsAAAA&ved=2ahUKEwj8lZHe37nwAhXY7WEKHantCIoQmxMoATAbegQIGxAD), [Thomas G. Stützle](https://www.google.com/search?sa=X&rlz=1C1CHBF_enVN919VN919&biw=1920&bih=947&sxsrf=ALeKk00gmloOXtx_C2QasRIUab_UljQEdQ:1620465319964&q=thomas+stutzle&stick=H4sIAAAAAAAAAOPgE-LRT9c3rEwpt0zKMC9T4gXxDNOyTFMsikostWSyk630k_Lzs_XLizJLSlLz4svzi7KtEktLMvKLFrHyAancxGKF4pLSkqqc1B2sjAAAc-CUUQAAAA&ved=2ahUKEwj8lZHe37nwAhXY7WEKHantCIoQmxMoAjAbegQIGxAE) 2004. Ant Colony Optimization.

[1] ĐỖ Đức Đông 2012. Phương pháp tối ưu đàn kiến và ứng dụng. Luận án tiến sĩ công nghệ thông tin

[2] Vũ Đức Quang 2016. Áp dụng thuật toán tối ưu hóa đàn kiến để giải bài toán vị trí cơ sở. Luận văn thạc sĩ ngành công nghệ thông tin.

[3] Trần Ngọc Hà 2017. Bài toán tối ưu tổ hợp và tính toán mềm. Luận án tiến sĩ công nghệ thông tin.

[3] Đặng Quý Linh 2013. Nghiên cứu áp dụng giải thuật đàn kiến để giải quyết bài toán người du lịch. Luận văn thạc sĩ kỹ thuật.