ĐẠI HỌC QUỐC GIA HÀ NỘI

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

VŨ ĐỨC QUANG

ÁP DỤNG THUẬT TOÁN TỐI ƯU HÓA ĐÀN KIẾN ĐỂ GIẢI QUYẾT BÀI TOÁN VỊ TRÍ CƠ SỞ

LUẬN VĂN THẠC SĨ NGÀNH CÔNG NGHỆ THÔNG TIN

Hà Nội, năm 2016

ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

VŨ ĐỨC QUANG

ÁP DỤNG THUẬT TOÁN TỐI ƯU HÓA ĐÀN KIẾN ĐỂ GIẢI QUYẾT BÀI TOÁN VỊ TRÍ CƠ SỞ

Ngành : Công nghệ thông tin
Chuyên ngành : Hệ thống thông tin
Mã số : 60480104

LUẬN VĂN THẠC SĨ NGÀNH CÔNG NGHỆ THÔNG TIN

Người hướng dẫn khoa học: PGS. TS Hoàng Xuân Huấn

Hà Nội, năm 2016

LỜI CAM ĐOAN

Tôi xin cam đoan luận văn này của tự bản thân tôi tìm hiểu, nghiên cứu dưới sự hướng dẫn của PGS.TS Hoàng Xuân Huấn. Các chương trình thực nghiệm do chính bản thân tôi lập trình, các kết quả là hoàn toàn trung thực. Các tài liệu tham khảo được trích dẫn và chú thích đầy đủ.

TÁC GIẢ LUẬN VĂN

Vũ Đức Quang

LỜI CẢM ƠN

Em xin bày tỏ lời cảm ơn chân thành tới tập thể các thầy cô giáo trường Đại học công nghệ - Đại học Quốc gia Hà Nội và Viện công nghệ thông tin - Viện Hàn lâm Khoa học và Công nghệ Việt Nam đã dạy dỗ chúng em trong suốt quá trình học tập chương trình cao học tại trường.

Đặc biệt em xin bày tỏ lòng biết ơn sâu sắc tới thầy giáo PGS.TS Hoàng Xuân Huấn, Trường Đại học Công nghệ - Đại học Quốc gia Hà Nội đã quan tâm, định hướng và đưa ra những góp ý, gợi ý, chỉnh sửa quý báu cho em trong quá trình làm luận văn tốt nghiệp.

Cuối cùng, em xin chân thành cảm ơn các bạn bè đồng nghiệp, gia đình và người thân đã quan tâm, giúp đỡ và chia sẻ với em trong suốt quá trình làm luận văn tốt nghiệp.

Em xin chân thành cảm ơn!

Hà Nội, tháng 11 năm 2016

Học viên

Vũ Đức Quang

MỤC LỤC

Trang

DANH SÁCH KÍ HIỆU, TỪ VIẾT TẮT

Viết tắt	**Viết đầy đủ**
ACO	*Ant Colony Optimization* (Tối ưu hóa đàn kiến)
ACS	*Ant Colony System* (Hệ kiến ACS)
aiNet	*Artificial Immune Network* (Thuật toán mạng miễn dịch)
AS	*Ant System* (Hệ kiến AS)
CFLP	*Capacitated Facility Location Problem* (Bài toán vị trí cơ sở có hạn chế khả năng)
CSLP	*Construction Site Layout Problem* (Bài toán bố trí vị trí xây dựng)
GA	*Genetic Algorithm* (Giải thuật di truyền)
IEM	Iterative Exact Method
MEM	Modified Exact Method
MLAS	*Multi-level Ant System* (Hệ kiến đa mức MLAS)
MMAS	*Max-Min Ant System* (Hệ kiến MMAS)
PSO	*Particle Swarm Optimization* (Tối ưu hóa bầy đàn)
r|p-centroid	*r|p-trung tâm*
SMMAS	*Smooth-Max Min Ant System* (Hệ kiến MMAS trơn)
SRFL	*Single Row Facility Layout* (Bài toán bố trí cơ sở theo hàng)
STS	Stochastic Tabu Search
TƯTH	*Tối ưu tổ hợp*
UPLP	*Uncapacitated Facility Location Problem* (Bài toán vị trí cơ sở không hạn chế khả năng)
VNS	Variable Neighborhood Search

DANH SÁCH BẢNG

DANH SÁCH HÌNH VẼ

MỞ ĐẦU

Trong cuộc sống, việc đạt lợi nhuận cao hay thấp trong kinh doanh buôn bán, cung cấp dịch vụ phụ thuộc rất nhiều yếu tố. Trong đó, có một yếu tốt quan trọng đầu tiên, đóng góp một phần rất lớn đó là xác định được địa điểm đặt dịch vụ thuật lợi – nơi cung cấp dịch vụ cho khách hàng. Có rất nhiều tiêu chí đặt ra khi chọn vị trí đặt cơ sở như: thuận tiện về giao thông, là nơi tập trung đông dân cư, … để làm sao thu được lợi nhuận cao nhất. Đặc biệt, đối với các trường hợp khẩn cấp như cứu thương, cứu hỏa thì yêu cầu về khoảng cách nhỏ nhất là vô cùng quan trọng, có thể nói là quan trọng nhất trong các yếu tố. Bài toán đặt ra là: đặt các trạm dịch vụ ở đâu để thời gian di chuyển bệnh nhân từ nơi xa bệnh viên nhất (hoặc ngược lại, từ các trạm dịch vụ đến nơi bệnh nhân xa nhất) là nhỏ nhất có thể. Còn với dịch vụ phổ biến như trạm xăng, thùng phiếu, bốt điện thoại, … thì yêu cầu lại là chi phí từ các khách hàng (hay người có nhu cầu) đến địa điểm phục vụ gần khách hàng nhất là nhỏ nhất.

Bài toán này thuộc dạng NP-khó, có rất nhiều các thuật giải khác nhau được đưa ra để có thể tìm lời giải tối ưu cho bài toán này như: thuật toán di truyền, thuật toán tham lam, thuật toán tối ưu hóa bầy đàn, tìm kiếm tabu… Tuy nhiên các giải thuật trên đều tốn chi phí về thời gian và/hoặc không gian lớn.

Tối ưu hóa đàn kiến (*Ant Colony Optimization - ACO*) là cách tiếp cận metaheuristic tương đối mới, do Dorigo giới thiệu vào năm 1991 và liên tục được phát triển cho đến nay. Thành công đầu tiên của các thuật toán ACO là giải quyết bài toán Người chào hàng nổi tiếng với số đỉnh lên tới hơn 2000 với kết quả thu được là tốt, hiệu quả của nó được chứng minh bằng thực nghiệm.

Đầu tiên, luận văn đã hệ thống hóa các kiến thức cơ sở về lý thuyết độ phức tạp thuật toán, lớp các bài toán P, NP, NP-khó và NP-đầy đủ. Sau đó, luận văn trình bày các bài toán điển hình trong lớp các bài toán vị trí cơ sở cùng các nghiên cứu đã được công bố gần đây. Tiếp theo, tác giả đề xuất thuật toán dựa trên giải thuật tối ưu đàn kiến giải một số bài toán vị trí cơ sở hiện nay và so sánh kết quả thu được với một số công trình đã được công bố gần đầy nhằm rút ra được các ưu nhược điểm của thuật toán. Kết quả này đã được tác giả công bố trong 2 công trình nghiên cứu khoa học.

Nội dung chính của luận văn được chia thành 4 chương như sau:

Chương 1: Tìm hiểu tổng quan về các kiến thức cơ sở về độ phức tạp thuật toán, lớp các bài toán P, NP và NP-khó và các bài toán thuộc lớp bài toán vị trí cơ sở cũng như các công bố gần đây.

Chương 2: Trình bày chi tiết về thuật toán tối ưu hóa đàn kiến.

Chương 3: Trình bày về cài đặt chương trình, thử nghiệm và so sánh kết quả với một số công trình đã công bố gần đây.

Kết luận

Tài liệu tham khảo

CHƯƠNG 1
MỘT SỐ KIẾN THỨC TỔNG QUAN VÀ BÀI TOÁN VỊ TRÍ CƠ SỞ

Trong cuộc sống, việc đạt lợi nhuận cao hay thấp trong kinh doanh buôn bán, cung cấp dịch vụ phụ thuộc rất nhiều yếu tố. Trong đó, có một yếu tố quan trọng đầu tiên, đóng góp một phần rất lớn đó là xác định được địa điểm đặt dịch vụ thuận lợi – nơi cung cấp dịch vụ. Có rất nhiều tiêu chí đặt ra khi chọn địa điểm: thuận tiện về giao thông, là nơi tập trung đông dân cư…để làm sao thu được lợi nhuận cao nhất. Đặc biệt, đối với các trường hợp khẩn cấp như cứu thương, cứu hỏa thì yêu cầu về khoảng cách nhỏ nhất là vô cùng quan trọng, có thể nói là quan trọng nhất trong các yếu tố.

Yêu cầu của bài toán vị trí cơ sở là tìm phương án đặt các trạm dịch vụ ở đâu để thời gian di chuyển bệnh nhân từ nơi xa bệnh viện nhất (hoặc ngược lại, từ các trạm dịch vụ đến nơi bệnh nhân xa nhất) là nhỏ nhất có thể. Còn với các dịch vụ phổ biến như trạm xăng, thùng phiếu, bốt điện thoại,… thì yêu cầu lại là tổng chi phí từ khách hàng (hay người có nhu cầu) đến địa điểm phục vụ gần khách hàng nhất là nhỏ nhất.

1.1. Độ phức tạp tính toán của bài toán

Gọi $T_A(X)$ là thời gian tính của thuật toán A đối với đầu vào X. Khi đó thời gian tính trong tình huống tồi nhất của thuật toán A đối với dữ liệu đầu vào kích thước n được định nghĩa như là:

$$T_A(n) = \max_{|X|=n} T_A(X)$$

Độ phức tạp trong tình huống tồi nhất của thuật toán P là thời gian tính trong tình huống tồi nhất của thuật toán *nhanh nhất* để giải nó:

$$T_p(n) = \min_{A\in\Delta} T_A(n) = \min_{A\in\Delta}(\max_{|X|=n} T_A(X))$$

Trong đó Δ là tập tất cả các thuật toán giải bài toán P.

Việc đánh giá đúng độ phức tạp của bài toán là một vấn đề hết sức phức tạp. Vì vậy chúng ta quan tâm đến việc đưa ra các cận trên và cận dưới cho nó.

Nếu ta có thuật toán A với thời gian tính trong tình huống tồi nhất là $T_A(n) = O(f(n))$ thì:

$$T_P(n) \leq T_A(n) \leq O(f(n))$$

Tức là ta có cận trên cho độ phức tạp của bài toán P. Thuật toán nhanh hơn sẽ cho cận trên tốt hơn.

Chúng ta còn quan tâm đến việc đánh giá cận dưới độ phức tạp của bài toán, nghĩa là quan tâm đến việc nó khó đến mức độ nào.

Để chỉ ra rằng:

$$T_P(n) = \Omega(f(n))$$

Ta cần phải chỉ ra rằng:

i. Có thuật toán với thời gian tính $\Omega(f(n))$ để giải bài toán P.
ii. Mọi thuật toán giải bài toán P đều đòi hỏi thời gian tính trong tình huống tồi nhất là $\Omega(f(n))$.

Yêu cầu ii. có thể thay thế bởi:

ii'. cận dưới cho độ phức tạp tính toán của bài toán P là $\Omega(f(n))$.

1.2. NP- đầy đủ

1.2.1. Bài toán quyết định

Bài toán quyết định là bài toán mà đầu ra chỉ có thể là 'yes' hoặc 'no' (Đúng/sai, 0/1, chấp nhận/từ chối, accept/reject). Đối với một bài toán quyết định, có những bộ dữ liệu vào của nó có câu trả lời (đầu ra) là 'yes' và cũng có những bộ dữ liệu vào có câu trả lời là 'no'. Những bộ dữ liệu vào có câu trả lời 'yes' ('no') sẽ được gọi là bộ dữ liệu vào 'yes' ('no').

Ví dụ 1:

- ✓ **Bài toán về tính nguyên tố:** "Hỏi số nguyên n có là số nguyên tố hay không?" N=23 là bộ dữ liệu vào 'yes', còn n=24 là bộ dữ liệu vào 'no' của bài toán.
- ✓ **Bài toán tổng con:** "Cho tập I gồm n số nguyên dương $x_1, x_2, \ldots, x_n$ và số nguyên dương T. Hỏi có thể tìm được tập con S của I với tổng các số trong S là bằng T?"
- ✓ **Bài toán người du lịch dạng quyết định (Dec – TSP):** "Tồn tại hay chăng hành trình của người du lịch với tổng chi phí không vượt quá số K cho trước?"

1.2.2. Bằng chứng ngắn gọn để kiểm tra

Rất nhiều các bài toán quyết định có một đặc điểm chung, đó là để xác nhận câu trả lời 'yes' đối với bộ dữ liệu vào 'yes' của chúng, ta có thể đưa ra

bằng chứng ngắn gọn dễ kiểm tra xác nhận câu trả lời 'yes' cho bộ dữ liệu vào 'yes' đó.

Ví dụ 2:

- ✓ Đối với bài toán kiểm tra tích hợp số: "Có phải số n là hợp số?", để xác nhận câu trả lời 'yes' cho đầu vào n, ta có thể đưa ra một ước số b (1<b<n) của n. Để kiểm tra xem b đúng là ước số của n ta có thể thực hiện phép chia n cho b sau thời gian đa thức.Trong ví dụ này b là bằng chứng ngắn gọn (vì b<n) và dễ kiểm tra: có thuật toán thời gian tính đa thức để kiểm tra b đúng là ước số của n.
- ✓ Đối với bài toán tổng con, bằng chứng xác nhận câu trả lời 'yes' đối với bộ dữ liệu $(x_1,...,x_n)$ là vecto $c = (c_1,...,c_n)$, trong đó $c_i = 1$ nếu x_i được chọn vào tập S và $c_i = 0$ nếu trái lại. Việc kiểm tra xem tập S gồm các số được chọn có thỏa mãn yêu cầu đặt ra hay không, rõ ràng, có thể thực hiện sau thời gian đa thức.
- ✓ Đối với bài toán người du lịch dạng quyết định, bằng chứng xác nhận câu trả lời 'yes' cho ma trận chi phí $C = \{c_{ij}: i,j=1,...,n\}$ của bài toán là dãy các thành phố trên hành trình. Việc kiểm tra xem dãy các thành phố đã cho có phải là hành trình với chi phí không vượt quá K có thể thực hiện xong sau thời gian đa thức.

Ta gọi **bằng chứng ngắn gọn dễ kiểm tra xác nhận câu trả lời 'yes'** cho bộ dữ liệu vào 'yes' của bài toán là một bằng chứng có độ dài bị chặn bởi một đa thức bậc cố định của độ dài dữ liệu đầu vào của bài toán, và việc kiểm tra nó là bằng chứng xác nhận câu trả lời 'yes' đối với đầu vào đã cho của bài toán có thể thực hiện xong sau thời gian đa thức.

Như vừa chỉ ra ở trên, các bài toán trong ví dụ 2 đều có bằng chứng ngắn gọn dễ kiểm tra để xác nhận câu trả lời 'yes' của bộ dữ liệu vào 'yes'.

Hoàn toàn tương tự, có thể đưa ra khái niệm **bằng chứng ngắn gọn dễ kiểm tra để xác nhận câu trả lời 'no'.**

Đối với một số bài toán việc đưa ra bằng chứng ngắn gọn xác định câu trả lời 'no' là dễ hơn so với việc đưa ra bằng chứng ngắn gọn xác định câu trả lời 'yes'.

Ví dụ 3:

Đối với bài toán kiểm tra tính nguyên tố, để đưa ra bằng chứng ngắn gọn dễ kiểm tra xác nhận câu trả lời 'no' cho đầu vào n của nó, ta có thể đưa ra một ước số b của n.

Có những bài toán mà việc đưa ra bằng chứng ngắn gọn dễ kiểm tra xác nhận câu trả lời 'yes' cũng như 'no' đều là không dễ dàng.

Ví dụ 4:

Cho đơn đồ thị vô hướng G = (V,E). Hỏi có đường đi đơn dài nhất nối hai đỉnh s và t của đồ thị G có tồn tại duy nhất?

1.2.3. Lớp bài toán P, NP và co-NP

Trước hết, ta nêu khái niệm về lớp các bài toán dễ giải – đó là các bài toán có thể giải được nhờ các thuật toán thời gian tính đa thức.

Định nghĩa: *Ta gọi **P** là lớp các bài toán có thể giải được sau thời gian đa thức.*

Ví dụ 5:

Bài toán về tính liên thông của đồ thị có thể giải được nhờ thuật toán với thời gian tính là $O(n^2)$, vì vậy, nó là bài toán thuộc lớp P. Bài toán cây khung nhỏ nhất giải được nhờ thuật toán Prim với thời gian $O(n^2)$, cũng thuộc vào lớp **P**.

Định nghĩa: *Ta gọi **NP** là lớp các bài toán quyết định mà để xác nhận câu trả lời 'yes' của nó ta có thể đưa ra bằng chứng ngắn gọn dễ kiểm tra.*

Ví dụ 6:

Các bài toán trình bày trong ví dụ 2 đều thuộc lớp **NP.**

Định nghĩa: *Ta gọi **co-NP** là lớp các bài toán quyết định mà để xác nhận câu trả lời 'no' của nó ta có thể đưa ra bằng chứng ngắn gọn dễ kiểm tra.*

Ví dụ 7:

Các bài toán trình bày trong ví dụ 3 đều thuộc lớp **co-NP.**

Bài toán trong ví dụ 4 còn chưa biết có thuộc vào lớp nào trong hai lớp **NP** và **co-NP** hay không.

Rõ ràng, nếu một bài toán thuộc lớp P, thì ta có thể tìm được lời giải của nó sau thời gian đa thức, và vì thế ta cũng có thể xác nhận được câu trả lời 'yes' của nó (bằng việc giải nó) sau thời gian đa thức. Vì vậy:

$$\mathbf{P} \subseteq \mathbf{NP}$$

Tương tự như vậy ta có:

$$\mathbf{P} \subseteq \mathbf{co\text{-}NP}$$

Một trong những vấn đề trung tâm của lý thuyết tính toán, đó là chứng minh hoặc bác bỏ đẳng thức:

$$\mathbf{P = NP}$$

Cho đến hiện nay vấn đề này vẫn là vấn đề mở.

1.2.4. Lớp bài toán NP-khó và NP-đầy đủ

Ta sẽ đưa ra định nghĩa về những bài toán khó nhất trong lớp **NP**: bài toán **NP**-đầy đủ (NP-complete).

Định nghĩa:

Một bài toán quyết định A được gọi là NP-đầy đủ nếu như:

i. *A là bài toán trong **NP**;*
ii. *Mọi bài toán trong **NP** đều có thể qui dẫn về A.*

Như vậy, có thể nói khái niệm về *"bài toán khó nhất"* trong lớp NP được xây dựng trên cơ sở phép qui dẫn. Nếu tất cả các bài toán trong NP có thể qui dẫn về một bài toán A thì A khó không kém bất cứ bài toán nào trong số chúng. Điều đáng ngạc nhiên là sự tồn tại của những bài toán có tính chất như vậy.

Khó khăn nhất là việc tìm ra được một bài toán như vậy. Bởi vì hễ chúng ta đã có một bài toán **NP**-đầy đủ thì để ta có thể dễ dàng chứng minh nhiều bài toán khác là **NP**-đầy đủ nhờ sử dụng kết quả sau đây.

Bổ đề*: Giả sử bài toán A là* **NP***-đầy đủ, bài toán B là thuộc* **NP***, và bài toán A qui dẫn về B. Khi đó bài toán B cũng là* **NP***-đầy đủ.*

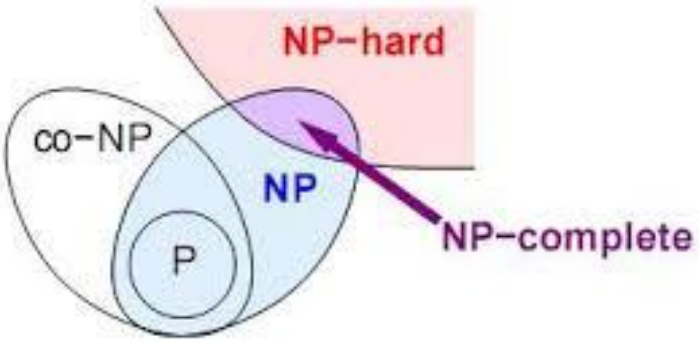
Định nghĩa: *Một bài toán A được gọi là* **NP***-khó (NP-hard) nếu như sự tồn tại thuật toán đa thức để giải nó kéo theo sự tồn tại thuật toán đa thức để giải một bài toán trong* **NP**.

Một cách không hình thức, có thể nói rằng nếu ta có thể giải được một cách hiệu quả một bài toán NP-khó cụ thể, thì ta cũng có thể giải hiệu quả bất kỳ bài toán nào trong NP bằng cách sử dụng thuật toán giải bài toán NP-khó như là một chương trình con.

Từ định nghĩa **NP**-khó suy ra rằng mỗi bài toán **NP**-đầy đủ đều là **NP**-khó. Tuy nhiên, như đã nêu ở trên, một bài toán là **NP**-khó không nhất thiết phải là **NP**-đầy đủ.

Từ Bổ đề suy ra rằng để chứng minh một bài toán A nào đó là NP-khó, ta chỉ cần chỉ ra phép qui dẫn một bài toán đã biết là NP-khó về nó.

Ta có bức tranh tạm thời đầy đủ hơn về phân lớp các bài toán trên hình 1.3:



Hình 1.1. Phân lớp các bài toán

Từ phần trình bày trên, ta thấy rằng có rất nhiều bài toán ứng dụng quan trọng thuộc vào lớp NP-khó, và vì thế khó hi vọng xây dựng được thuật toán đúng hiệu quả để giải chúng. Một trong những hướng phát triển thuật toán giải các bài toán như vậy là xây dựng các thuật toán gần đúng.

1.3. Bài toán vị trí cơ sở không hạn chế khả năng

Bài toán vị trí cơ sở không hạn chế khả năng (Uncapacitated Facility Location Problem - UFLP) có thể được gọi với nhiều tên khác nhau, chẳng hạn như: Simple Plant Location Problem, the location of bank accounts problem, warehouse location problem, the standardization and unification problem, the problem of a nonrecoverable tools optimal system…

Bài toán UFLP được phát biểu như sau: Xét một tập $I = \{1, 2, 3, \dots, N\}$ các cơ sở tiềm năng cung cấp sản phẩm hoặc dịch vụ. Một cơ sở i $\in I$ có chi phí xây dựng là C_i ($C_i > 0$). Mỗi cơ sở mở có thể cung cấp một số lượng không giới hạn hàng hóa cho mỗi khách hàng. Và một tập $J = \{1, 2, \dots, M\}$ là tập các khách hàng sử dụng dịch vụ. Giá trị g_{ij} (với $i \in I$ và $j \in J$) là chi phí vận chuyển từ cơ sở i đến khách hàng j. Mục tiêu là xác định một tập hợp con S của tập hợp các địa điểm cơ sở tiềm năng I ($S \subseteq I$, $S \neq \varnothing$) để cung cấp cho tất cả các khách hàng sao cho tổng chi phí xây dựng và chi phí vận chuyển là nhỏ nhất.

$$F(S) = \sum_{i \in S} C_i + \sum_{j \in J} \min_{i \in S} g_{ij} \rightarrow \min_{S \subseteq I} \quad (1.1)$$

Bài toán UFLP còn có thể được mô tả dưới dạng một bài toán quy hoạch tuyến tính nguyên như sau:

$$Minimize \sum_{i \in I} c_i y_i + \sum_{i \in I} \sum_{j \in J} g_{ij} x_{ij} \quad (1.2)$$

sao cho:

$$x_{ij} \leqslant y_i \quad (i \in I, j \in J) \quad (1.3)$$

$$\sum_{i \in I} x_{ij} = 1 \ (j \in J) \quad (1.4)$$

$$x_{ij} \in \{0,1\} \ (i \in I, j \in J) \quad (1.5)$$

$$y_i \in \{0,1\} \ (i \in I) \quad (1.6)$$

Bài toán UFLP là một trong những bài toán được nghiên cứu rộng rãi nhất trong lớp các bài toán tối ưu hóa tổ hợp. Bài toán được đề xuất lần đầu tiên bởi Erlenkotter năm 1978 dưới dạng một bài toán quy hoạch tuyến tính. Neebe và Khumawala năm 1981, Karkazis và Boffey năm 1981 đề xuất bài toán với giả định mỗi cơ sở chỉ giao dịch được một sản phẩm. Năm 1987 Klincewicz và Luss là người đầu tiên nghiên cứu một mô hình vị trí cơ sở mà không bị hạn chế về số lượng sản phẩm tại mỗi cơ sở.

Tất cả các phương pháp tiếp cận quan trọng có liên quan đến bài toán UFLP có thể được chia thành 2 loại chính là: Thuật toán chính xác và phương pháp dựa trên metaheuristics. Các thuật toán chính xác để giải quyết bài toán UFLP chẳng hạn như nhánh cận, quy hoạch tuyến tính (linear programing), thuật toán nới lỏng Lagrăng (Lagrangean relaxation). Cách tiếp cận đối ngẫu (dual approach (DUALLOC)) và phương pháp đối ngẫu nguyên thủy (primaldual approaches). Bài toán UFLP được chứng minh là NP khó nên các thuật toán chính xác trên có thể không thực sự hiệu quả khi giải quyết các trường hợp số lượng cơ sở lớn. Vì vậy, đã có rất nhiều các nghiên cứu giải bài toán UFLP dựa trên phương pháp heuristics hay metaheuristics.

1.4. Bài toán vị trí cơ sở có hạn chế khả năng

Bài toán vị trí cơ sở có hạn chế khả năng (Capacitated Facility Location Problem – CFLP) là khái quát hóa bài toán UFLP khi mà những cơ sở bị giới

hạn về số lượng sản phẩm. Mặc dù mô hình toán học của hai bài toán này không khác nhau nhiều nhưng các phương pháp giải bài toán CFLP thì thường khó hơn.

Trong bài toán CFLP, mỗi khách hàng có nhu cầu nhất định để đáp ứng và các cơ sở có hạn chế về công suất phục vụ hay hạn chế về sản phẩm cung cấp, tức là tổng nhu cầu của khách hàng được phân công một cơ sở không thể vượt quá khả năng của cơ sở đó. Cả hai bài toán UFLP và CFLP được coi là NP-khó (Garey & Johnson, 1990; Kariv & Hakimi, 1979). Các bài toán vị trí cơ sở có thể được nghiên cứu trên không gian rời rạc hoặc liên tục. Khi cơ sở có thể được đặt ở bất cứ nơi nào trong khu vực, bài toán được coi là là liên tục. Khi cơ sở có thể được đặt chỉ tại các địa điểm cụ thể, bài toán được coi là rời rạc. Mục tiêu của bài toán CFLP là tìm ra p vị trí đặt cơ sở sao cho tổng chi phí xây dựng và chi phí vận chuyển giữa các khách hàng và cơ sở là nhỏ nhất. Chính vì vậy, bài toán CFLP còn có tên gọi khác là bài toán p-median.

Bài toán CFLP được mô tả chi tiết như sau:

- $I = \{1, 2 \dots n\}$ các khách hàng
- $J = \{1, 2 \dots m\}$ các cơ sở tiềm năng
- F_j chi phí xây dựng cơ sở j ($j \in J$)
- H_i nhu cầu của mỗi khách hàng i ($i \in I$)
- S_j khả năng cung cấp của cơ sở j ($j \in J$)
- C_{ij} chi phí di chuyển (vận chuyển) từ khách hàng i đến cơ sở j.
- a là số lượng ít nhất các cơ sở có thể được chọn để mở
- b là số lượng nhiều nhất các cơ sở có thể được chọn để mở

$$x_{ij} = \begin{cases} 1 \text{ nếu khách hàng } i \text{ chọn cơ sở } j, \text{và} \\ 0 \text{ nếu ngược lại;} \end{cases}$$

$$y_j = \begin{cases} 1 \text{ nếu cơ sở } j \text{ được mở} \\ 0 \text{ nếu ngược lại} \end{cases}$$

Hàm mục tiêu có thể được biểu diễn như sau:

$$Min \sum_{j \in J} f_j y_j + \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} \qquad (1.7)$$

sao cho

$$\sum_{j \in J} x_{ij} = 1, \ \forall i \in I, \qquad (1.8)$$

$$\sum_{i\in I} h_i x_{ij} \le s_j y_j, \ \forall j \in J, \tag{1.9}$$

$$a \le \sum_{i\in I} y_j \le b \quad \forall j \in J, \tag{1.10}$$

$$x_{ij} \in \{0,1\}, \ \forall i \in I, \forall j \in J, \tag{1.11}$$

$$y_j \in \{0,1\}, \ \forall j \in J \tag{1.12}$$

Hàm mục tiêu (1.7) là để tối thiểu hóa tổng chi phí sao cho các điều kiện từ (1.8) đến (1.12) đều thỏa mãn. Trong đó, hạn chế (1.8) đảm bảo rằng mỗi khách hàng chỉ được cung cấp bởi một cơ sở. Hạn chế (1.9) đảm bảo rằng tổng nhu cầu của khách hàng được phân công đến một cơ sở không vượt quá khả năng đáp ứng của cơ sở đó. Hạn chế (1.10) đảm bảo rằng số lượng các cơ sở mở là trong khoảng a và b, đối với bài toán p-median thì số lượng cơ sở được mở ra chính xác bằng số p. Hạn chế (1.11) và (1.12) là các điều kiện nhị phân.

Trong trường hợp $h_i = 1, \forall i \in I$ và $s_j = n, \forall j \in J$ thì bài toán CFLP sẽ trở thành bài toán UFLP.

Có rất nhiều phương pháp đã được đề xuất để giải quyết bài toán bao gồm thuật toán dựa trên đối ngẫu được Erlenkotter [13] công bố. Ý tưởng chính là sử dụng tiếp cận đối ngẫu quy hoạch tuyến tính tìm một cận cho hàm mục tiêu. Ngoài ra, các thuật toán đã được áp dụng để giải quyết bài toán UFLP cũng đã được các tác giả triển khai cho bài toán CFLP.

1.5. Bài toán vị trí cơ sở cạnh tranh

Với bài toán UFLP hay CFLP, hàm mục tiêu được đưa ra nhằm tối đa hóa lợi nhuận của một người hoặc giảm thiểu tổng chi phí của khách hàng với cơ sở. Nhưng bài toán vị trí cơ sở cạnh tranh (Competitive Facilities Location Problem) hay còn được gọi là bài toán r|p-centroid (r|p trung tâm) xét tình huống phức tạp hơn khi hai người chơi *Trước* và *Sau* là đối thủ của nhau lần lượt chọn vị trí đặt cơ sở. Trong khi đó, mỗi khách hàng dựa trên sở thích riêng của họ, lựa chọn cơ sở tốt nhất trong số tất cả các cơ sở được mở làm nhà cung cấp cho mình do đó mang lại nhuận cho cả hai bên.

Bài toán $(r|p)$-trung tâm lần đầu tiên được Hakimi [16] nghiên cứu dưới dạng bài toán rời rạc, có thể phát biểu như sau: Cho một tập I hữu hạn các địa điểm có thể chọn để đặt các cơ sở dịch vụ và một tập J hữu hạn của các vị trí của khách hàng, ma trận (d_{ij}) là khoảng cách từ khách hàng $j \in J$ tới cơ sở $i \in I$, các

giá trị w_j xác định lợi nhuận của một cơ sở thu được trong việc phục vụ khách hàng j. Hai công ty / người chơi *Trước* và *Sau* sẽ mở các cơ sở kinh doanh tại các điểm của tập I. Đầu tiên, người chơi *Trước* mở p cơ sở. Biết được quyết định của *Trước*, *Sau* sẽ chọn để mở ra r cơ sở. Mỗi khách hàng sẽ chọn ra cơ sở gần họ nhất trong số $p + r$ cơ sở của cả hai người chơi đã mở ra như là nhà cung cấp cho mình. Kết quả là tập khách hàng sẽ được chia thành hai phần: tập khách hàng lựa chọn *Trước* và tập khách hàng lựa chọn *Sau*. Bài toán đặt ra là tìm ra p vị trí đặt cơ sở cho *Trước* để đạt tối đa nhất lợi nhuận dưới sự phản ứng mạnh mẽ nhất có thể của *Sau*

Hiện nay, các nghiên cứu cơ bản của bài toán này có thể được phát triển thêm nhiều biến thể [10], [25], [29], dưới dạng các bài toán trên đồ thị [25], [29] và trong không gian Euclide [10]. Tuy nhiên, dạng bài toán rời rạc vẫn được nhiều người quan tâm nhất và người ta đã chỉ ra rằng bài toán của *Trước* thuộc loại $\sum_2^P - khó$ [29] còn khi đã biết các cơ sở của *Trước* thì bài toán của *Sau* thuộc loại NP-khó [8] [37]

Bài toán dưới dạng đồ thị. Bài toán $(r|p)$-trung tâm rời rạc được phát biểu như sau: (xem [10] [37])

Xét một đồ thị hai phía đầy đủ có trọng số $G = (I, J, E)$, trong đó tập đỉnh $I = \{1, \ldots, m\}$ biểu diễn tập hợp các địa điểm cơ sở tiềm năng mà hai người chơi *Trước* và *Sau* có thể lựa chọn để mở cơ sở, tập đỉnh $J = \{1, \ldots, n\}$ biểu diễn tập khách hàng, $E = I \times J$ là tập các cạnh có độ đo khoảng cách tương ứng $d_{ij} \forall (i, j) \in E$, mỗi $j \in J$ có trọng số w_j $(w_j > 0)$ ứng với doanh thu mà cơ sở nhận được nếu khách hàng này chọn cơ sở làm nhà cung cấp. Biết rằng mỗi khách hàng sẽ chọn cơ sở phục vụ gần nó nhất, trong trường hợp khoảng cách tới *Trước* bằng khoảng cách tới *Sau* thì khách hàng sẽ chọn *Trước*.

Ta cần tìm p vị trí trong tập I cho *Trước* sao cho tối đa hóa doanh thu của *Trước* với lưu ý rằng *Sau* sẽ chọn r cơ sở từ các địa điểm còn lại cũng nhằm tối đa hóa doanh thu của họ khi đã biết vị trí dịch vụ của *Trước*.

Gọi (X, Y) là lời giải cho bài toán $(r|p)$-trung tâm, trong đó $X \subseteq I$, $|X| = p$ là tập các cơ sở được *Trước* chọn, và $Y \subseteq \{I \setminus X\}$, $|Y| = r$ là tập các cơ sở được *Sau* lựa chọn. Với mỗi tập $V \subseteq I$ và $\forall j \in J$, ký hiệu $D(j, V) = min\{d_{ji} | \, i \in V\}$ cho khoảng cách tối thiểu từ khách hàng j đến tất cả các cơ sở trong tập V. Khi đó tập khách hàng sẽ được chia thành hai phần: Tập khách hàng lựa chọn *Trước* $U^T = \{ j \in J | D(j, Y) \geq D(j, X)\}$ và tập khách hàng lựa

chọn Sau $U^S = \{J \backslash U^T\}$. Doanh thu của *Trước* sẽ là $p^T = \sum_{j \in U^T} w_j = \sum_{j \in J} w_j - p^s$ còn doanh thu của Sau sẽ là $p^S = \sum_{j \in U^s} w_j = \sum_{j \in J} w_j - p^T$.

Yêu cầu bài toán là tìm ra tập cơ sở X cho *Trước* sao cho lợi nhuận của họ nhận được là nhiều nhất cho dù Sau có lựa chọn cơ sở Y nào đi nữa. Bài toán tìm tập cơ sở Y tối ưu cho Sau khi biết trước X được gọi là bài toán $(r|Xp)$-trung vị ($(r|Xp) - medianoid$) và nó đã được Hakimi chứng minh là NP-khó [16]. Noltemeier cùng các cộng sự [25] đã chứng minh bài toán tập cơ sở X cho *Trước* có độ phức tạp là $\sum_2^P - khó$ ngay cả khi ma trận (d_{ij}) là ma trận khoảng cách Euclide trên mặt phẳng.

Bài toán dưới dạng quy hoạch hai mức. Bài toán $(r|p)$-trung tâm có thể phát biểu dưới dạng bài toán tìm minimax trong bài toán quy hoạch hai mức. Ký hiệu:

$$x_i = \begin{cases} 1 \text{ nếu } Trước \text{ mở cơ sở } i \\ 0 \text{ nếu ngược lại} \end{cases} \quad (1.13)$$

$$y_i = \begin{cases} 1 \text{ nếu } Sau \text{ mở cơ sở } i \\ 0 \text{ nếu ngược lại} \end{cases} \quad (1.14)$$

$$z_j = \begin{cases} 1 \text{ nếu khách hàng } j \text{ được } Trước \text{ phục vụ} \\ 0 \text{ nếu khách hàng } j \text{ được } Sau \text{ phục vụ} \end{cases} \quad (1.15)$$

Khi đó $X = \{ i \in I \mid x_i = 1\}$, $Y = \{ i \in I \mid y_i = 1\}$. Với mỗi khách hàng j, chúng ta định nghĩa tập cơ sở $I_j(X)$ cho phép Sau "thu hút" khách hàng j.

$$I_j(X) = \left\{ i \in I \mid d_{ij} < \min_{l \in I} \{ d_{lj} \mid x_l = 1 \} \right\} \quad (1.16)$$

Với các ký hiệu như trên, bài toán $(r|p)$-trung tâm được biểu diễn như sau:

$$\begin{aligned} &\max_x \sum_{j \in J} w_j z_j^*(X), \\ &\sum_{i \in I} x_i = p, \\ &x_i \in \{0,1\}, i \in I, \end{aligned} \quad (1.17)$$

Với $z_j^*(X), y_i^*(X)$ là phương án tối ưu thì bài toán Sau sẽ là:

$$\begin{aligned} &\max_{y,z} \sum_{j\in J} w_j(1-z_j), \\ &\sum_{i\in I} y_i = r, \\ &1-z_j \le \sum_{i\in I_j(X)} y_i,\ i\in I, \\ &y_i, z_j \in \{0,1\},\ i\in I, j\in J. \end{aligned} \tag{1.18}$$

Lưu ý rằng, hàm mục tiêu $W^*(X) = \sum_{j\in J} w_j z_j^*(X)$ là tổng lợi nhuận của *Trước* khi nó mở đúng p cơ sở, giá trị này phụ thuộc vào lời giải tối ưu của *Sau*.

Đã có nhiều thuật toán đề xuất cho bài toán này [8] [37] [9]. Đặc biệt, Davydov cùng các cộng sự [9] theo tiếp cận metaheuristics đã đề xuất hai thuật toán VNS (Variable Neighborhood Search) và STS (Stochastic Tabu Search) giải gần đúng nhanh bài toán của *Trước*, trong đó họ dùng phần mềm CPLEX (một phần mềm của IBM cung cấp nhằm giải các bài toán quy hoạch tuyến tính) để tìm lời giải tối ưu cho *Sau* mỗi khi biết các cơ sở của *Trước*; Alekseeva cùng các cộng sự [5] phát triển thuật toán IM giải đúng bài toán *Trước*, trong đó cũng sử dụng phần mềm CPLEX cho toán *Sau*. Kết quả thực nghiệm cho thấy ưu điểm của các thuật toán này so với các thuật toán đã biết trước đó.

1.6. Bài toán bố trí vị trí xây dựng

Bố trí vị trí xây dựng (Construction Site Layout Problem - CSLP) là một nhiệm vụ quan trọng cần được xem xét cẩn thận trong công tác quy hoạch xây dựng. Mục tiêu của bài toán CSLP là sắp xếp các cơ sở như: văn phòng, nhà kho, phòng trưng bày, … trong không gian của một dự án xây dựng một cách hợp lý. Thông thường nhiệm vụ này được thực hiện bởi các nhà quản lý xây dựng. Tuy nhiên, quyết định này thường được đưa ra dựa trên trực giác, thí nghiệm và kinh nghiệm. Việc bố trí hợp lý các cơ sở sẽ góp phần làm giảm thiểu chi phí xây dựng, thời gian vận chuyển, xử lý vật liệu và giảm thiểu việc di chuyển nguyên liệu hay trang thiết bị, đặc biệt đối với các dự án lớn.

Có rất nhiều hàm mục tiêu cho bài toán đã được công bố, tuy nhiên hai hàm mục tiêu sau đây được các nghiên cứu rộng rãi và phổ biến nhất.

1.6.1. Hàm mục tiêu thứ nhất

Hàm mục tiêu thứ nhất được chia nhỏ ra thành ba trường hợp ứng với ba loại điều kiện khác nhau trong thực tế, ta kí hiệu các trường hợp này lần lượt là TH1, TH2 và TH3.

Trong TH1, bài toán được giả định rằng các vị trí là có sẵn và mỗi vị trí được phép đặt duy nhất một cơ sở [14]. Các cơ sở được lựa chọn được liệt kê trong bảng 1.1.

Bảng 1.1. Ký hiệu các cơ sở

Cơ sở	Kí hiệu
Site office	SO
Falsework shop	FS
Labor residence	LR
Storeroom 1	S1
Storeroom 2	S2
Carpentry workshop	CW
Reinforcement steel workshop	RW
Side gate	SG
Electrical, water and other utilities control room	UR
Concrete batch workshop	BW
Warehouse	W

Các cơ sở trong dự án thường có sự kết nối khá chặt chẽ với nhau, ví dụ như: một nhân viên thường di chuyển giữa *Site office* và *Concrete batch workshop* nhưng lại rất hiếm khi di chuyển từ *Site office* tới *Storeroom*. Do đó các địa điểm để đặt các cơ sở sẽ được lựa chọn cẩn thận để giảm thiểu tổng chi phí vận chuyển và đó cũng là mục tiêu bài toán. Tổng khoảng cách được định nghĩa như trong hàm (1.19) và (1.20)

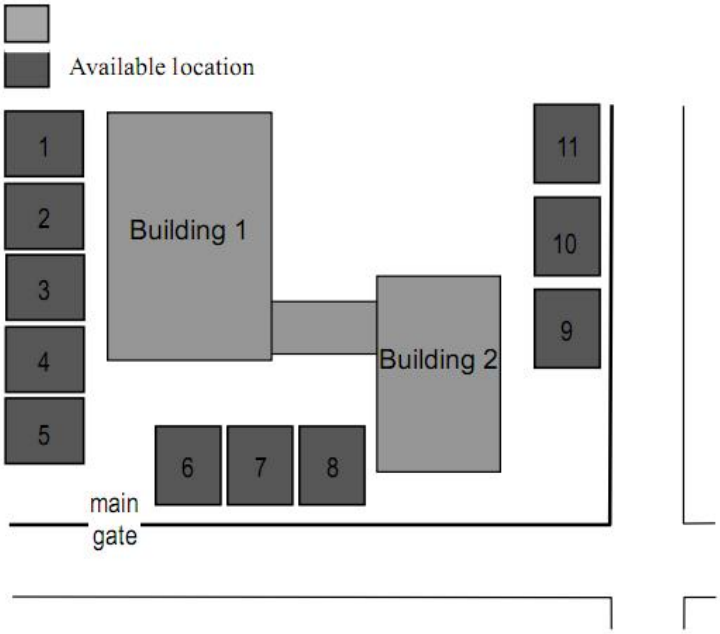
$$MinF = \sum_{i=1}^{n}\sum_{x=1}^{n}\sum_{j=1}^{n}\delta_{xi} f_{ij} d_{ij} \tag{1.19}$$

sao cho:

$$\sum_{x=1}^{n}\delta_{xi} = 1 \ \{i = 1,2,...,n\} \tag{1.20}$$

Với n là số lượng cơ sở, δ_{xi} là ma trận hoán vị. Hệ số f_{ij} là tần suất di chuyển được thực hiện bởi nhân viên giữa cơ sở i và cơ sở j. Từ đó, có thể thấy f_{ij} bằng f_{ji}. Tần số này biểu thị bằng số lượng di chuyển trong một khoảng thời gian nhất định, thông thường là một ngày. Hệ số d_{ij} là khoảng cách giữa vị trí i

và vị trí j. Do đó, hàm mục tiêu F là tổng khoảng cách di chuyển được thực hiện bởi nhân viên. Hình 1.2. dưới là ví dụ về một dự án với n = 11.



Hình 1.2. Các vị trí biểu diễn một dự án xây dựng

Tần suất di chuyển (trong một ngày) giữa các cơ sở được cho trong bảng 1.2.

Bảng 1.2. Tần suất di chuyển giữa các cơ sở

	SO	**FS**	**LR**	**S1**	**S2**	**CW**	**RW**	**SG**	**UR**	**BW**	**W**
SO	0	5	2	2	1	1	4	1	2	9	1
FS	5	0	2	5	1	2	7	8	2	3	8
LR	2	2	0	7	4	4	9	4	5	6	5
S1	2	5	7	0	8	7	8	1	8	5	1
S2	1	1	4	8	0	3	4	1	3	3	6
CW	1	2	4	7	3	0	5	8	4	7	5
RW	4	7	9	8	4	5	0	7	6	3	2
SG	1	8	4	1	1	8	7	0	9	4	8
UR	2	2	5	8	3	4	6	9	0	5	3
BW	9	3	6	5	3	7	3	4	5	0	5
W	1	8	5	1	6	5	2	8	3	5	0

Nhìn bảng trên ta có thể thấy tần số di chuyển giữa từ một cơ sở đến cơ sở khác và ngược lại là như nhau, hay nói cách khác ma trận tần số f_{ij} là ma trận

đối xứng qua đường chéo chính. Ví dụ, tần số di chuyển từ SO đến BW và ngược lại đều bằng 9.

Khoảng cách giữa các vị trí đo bằng đơn vị mét (m) và được cung cấp ở bảng 1.3 dưới đây.

Bảng 1.3. Khoảng cách giữa các cơ sở (đơn vị m)

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
1	0	15	25	33	40	42	47	55	35	30	20
2	15	0	10	18	25	27	32	42	50	45	35
3	25	10	0	8	15	17	22	32	52	55	45
4	33	18	8	0	7	9	14	24	44	49	53
5	40	25	15	7	0	2	7	17	37	42	52
6	42	27	17	9	2	0	5	15	35	40	50
7	47	32	22	14	7	5	0	10	30	35	40
8	55	42	32	24	17	15	10	0	20	25	35
9	35	50	52	44	37	35	30	20	0	5	15
10	30	45	55	49	42	40	35	25	5	0	10
11	20	35	45	53	52	50	40	35	15	10	0

TH2 là trường hợp mở rộng từ TH1 với *Site gate* và *Main gate* phải luôn được đặt tương ứng ở vị trí thứ 1 và vị trí thứ 10. Điều này được xác định dựa trên đời sống thực tế. Trong xây dựng, M*ain gate* và *Side gate* thường được xác định đầu tiên và nằm ở những vị trí quan trọng thuận lợi cho việc ra vào. Do đó, *Main gate* và *Side gate* được định vị trên những vị trí xác định trước.

TH3 được giả định rằng các cơ sở như: *Site office, Labor residence, Concrete batch shop* không thể được đặt ở những vị trí thứ 7 và 8 [20]. Trường hợp này được sử dụng để minh họa các hạn chế, trong đó các cơ sở có kích thước lớn sẽ không được phép đặt ở những vị trí nhỏ.

1.6.2. Hàm mục tiêu thứ hai

Hàm mục tiêu thứ hai được đưa ra với hai loại trường hợp ứng với các bộ dữ liệu khác nhau, chúng ta sẽ kí hiệu các trường hợp đó lần lượt là TH4 và TH5.

Mục tiêu bố trí cơ sở ở phần này là chi phí khoảng cách giữa các đối tượng liền kề. Bài toán được mô phỏng bởi Yeh [38] và Mawdesley [23] cùng các cộng sự. Bài toán CSLP còn có thể được xây dựng như sau:

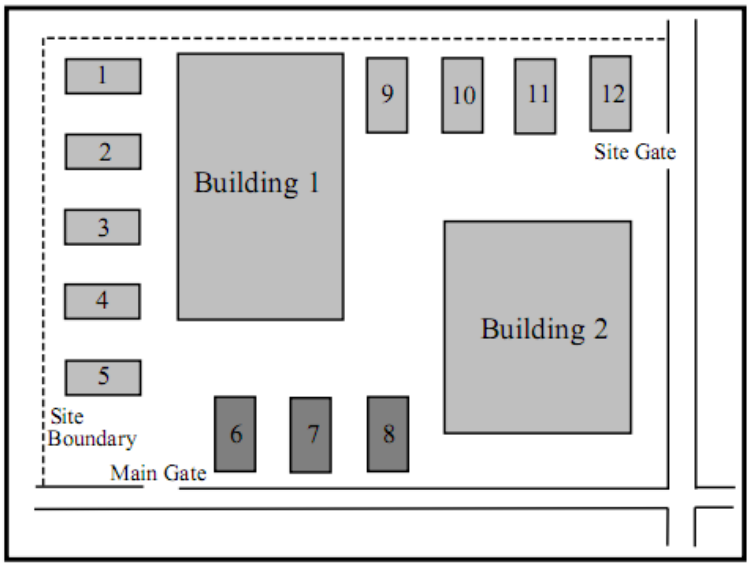
$$Min\,F = \sum_{x=1}^{n}\sum_{i=1}^{n}\delta_{xi}C_{xi} + \sum_{x=1}^{n}\sum_{i=1}^{n}\sum_{y=1}^{n}\sum_{j=1}^{n}\delta_{xi}\delta_{yj}A_{ij}D_{xy} \qquad (1.21)$$

sau cho:

$$\delta_{yj} = 0 if\ \delta_{xi} = 1 and\ y \neq x \qquad (1.22)$$

$$\delta_{xj} = 1 if\ \delta_{xi} = 1 and\ i \neq j \qquad (1.23)$$

Với hàm mục tiêu F được tính bằng tổng các tích của khoảng cách và chi phí. δ_{xi} là giá trị của ma trận hoán vị (=1 nếu cơ sở x được đặt vào vị trí i), C_{xi} là chi phí xây dựng cơ sở x ở vị trí i. $A_{ij} = 1$ nếu vị trí i là hàng xóm của vị trí j, D_{xy} là chi phí tương tác giữa cơ sở x với cơ sở y. Hình 1.3. là một ví dụ với n = 12.



Hình 1.3. Ví dụ về một dự án xây dựng

Trong TH4, có hai tòa nhà được đặt cố định trên một khuôn viên và 12 vị trí có sẵn để đặt các cơ sở tiện ích như: Reinforcing steel shop 1 (R1), Reinforcing steel shop 2 (R2), Carpentry shop 1 (C1), Carpentry shop (C2), Falsework shop 1 (F1), Falsework shop2 (F2), Concrete batch plant 1 (B1), Concrete batch plant 2 (B2), Job office (JO), Labor residence (LR), Electricity equipment and water-supply shop (E), Warehouse (W). Ma trận chi phí xây dựng C, ma trận chỉ số các vị trí láng giềng A và ma trận chi phí tương tác D được cho tương ứng ở các bảng 1.4, 1.5 và 1.6.

Bảng 1.4. Ma trận chi phí xây dựng (C)

	1	2	3	4	5	6	7	8	9	10	11	12
R1	35	35	30	30	35	15	10	15	6	6	7	10
R2	35	30	9	9	13	30	30	35	15	18	12	7
C1	18	15	15	15	15	8	14	10	8	10	15	15
C2	13	7	12	18	18	15	15	15	15	8	8	12
F1	18	15	15	20	15	8	10	8	8	7	15	15
F2	14	8	10	17	12	15	15	15	15	8	7	9
B1	32	35	15	15	15	10	9	13	7	10	15	15
B2	31	30	9	8	15	18	15	16	15	15	15	15
JO	39	35	13	8	8	15	18	15	8	18	9	18
LR	18	8	8	8	15	10	15	15	13	15	15	15
E	7	10	8	19	15	10	10	8	15	10	6	15
W	9	10	6	7	7	7	15	15	18	15	15	12

Bảng 1.5. Ma trận láng giềng (A) trong TH4

	1	2	3	4	5	6	7	8	9	10	11	12
1	0	1	0	0	0	0	0	0	0	0	0	0
2	1	0	1	0	0	0	0	0	0	0	0	0
3	0	1	0	1	0	0	0	0	0	0	0	0
4	0	0	1	0	1	0	0	0	0	0	0	0
5	0	0	0	1	0	1	0	0	0	0	0	0
6	0	0	0	0	1	0	1	0	0	0	0	0
7	0	0	0	0	0	1	0	1	0	0	0	0
8	0	0	0	0	0	0	1	0	1	0	0	0
9	0	0	0	0	0	0	0	1	0	1	0	0
10	0	0	0	0	0	0	0	0	1	0	1	0
11	0	0	0	0	0	0	0	0	0	1	0	1
12	0	0	0	0	0	0	0	0	0	0	1	0

Bảng 1.6. Ma trận chi phí tương tác giữa các cơ sở (D) trong TH4

	1	2	3	4	5	6	7	8	9	10	11	12
1	0	0	0	0	0	0	0	0	100	100	0	0
2	0	0	0	0	0	0	0	00	100	100	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0

6	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0
9	100	100	0	0	0	0	0	0	0	0	0	0
10	100	100	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0

TH5 cũng tương tự như TH4 chỉ khác nhau về ma trận vị trí láng giềng A và ma trận tương tác D. Các ma trận này được cho tương ứng ở bảng 1.7, 1.8 như sau:

Bảng 1.7. Ma trận láng giềng (A) trong TH5

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
1	0	1	0.5	0	0	0	0	0	0	0	0.5	0
2	1	0	1	0.5	0	0	0	0	0	0	0	0.5
3	0.5	1	0	1	0.5	0	0	0	0	0	0	0
4	0	0.5	1	0	1	0.5	0	0	0	0	0	0
5	0	0	0.5	1	0	1	0.5	0	0	0	0	0
6	0	0	0	0.5	1	0	1	0.5	0	0	0	0
7	0	0	0	0	0.5	1	0	1	0.5	0	0	0
8	0	0	0	0	0	0.5	1	0	1	0.5	0	0
9	0	0	0	0	0	0	0.5	1	0	1	0.5	0
10	0	0	0	0	0	0	0	0.5	1	0	1	0.5
11	0.5	0	0	0	0	0	0	0	0.5	1	0	1
12	0	0.5	0	0	0	0	0	0	0	0.5	1	0

Bảng 1.8. Ma trận chi phí tương tác giữa các cơ sở (D) trong TH5

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
1	0	0	0	0	0	0	0	0	100	100	100	0
2	0	0	0	0	0	0	0	0	100	100	100	0
3	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0

7	0	0	0	0	0	0	0	0	100	100	100	0
8	0	0	0	0	0	0	0	0	100	100	100	0
9	100	100	0	0	0	0	100	100	0	-50	-50	-50
10	100	100	0	0	0	0	100	100	-50	0	0	0
11	100	100	0	0	0	0	100	100	-50	0	0	0
12	0	0	0	0	0	0	0	0	-50	0	100	0

Bài toán CSLP được phân loại là bài toán phân loại bậc hai và nó đã được chứng minh là NP-khó [24]. Có rất nhiều phương pháp khác nhau để giải quyết bài toán bao gồm cả việc áp dụng các thuật toán chính xác chẳng hạn như nhánh cận…hay các thuật toán gần đúng như: PSO, GA, và ACO…

Phương pháp GA đã được áp dụng rộng rãi cho việc giải quyết bài toán CSLP [20]. Mawdesley cùng các cộng sự đã đề xuất một giải thuật mà các chi phí di chuyển được mô phỏng sử dụng thuật toán GA tăng cường (augmented GA) [23]. Ka Chi Lam cùng các cộng sự [19] đã đề xuất một thuật toán kết hợp thuật toán Max-Min Ant System (MMAS) với GA.

Thuật toán PSO dựa trên quá trình tối ưu hóa ngẫu nhiên được lấy cảm hứng từ hành vi tự nhiên của các con vật như chim, ong, cá. Mỗi cá thể trong đàn có thể được sử dụng làm đại diện cho một lời giải của bài toán. Một bầy đàn gồm rất nhiều cá thể được khởi tạo ở các vị trí ngẫu nhiên sẽ bay về phía vị trí tối ưu theo một con đường mà luôn được cập nhật lặp đi lặp lại dựa trên vị trí hiện tại tốt nhất của các cá thể. Trong [39] Zhang và Wang đã đề xuất một phương pháp dựa trên thuật toán PSO để giải quyết các công trình xây dựng không đồng đều, bài toán vị trí cơ sở.

Thuật toán ACO là một thuật toán metaheuristic được lấy cảm hứng từ sinh học bắt chước hành vi của kiến trong quá trình tìm kiếm thức ăn. Thuật toán này đã được Dorigo cùng các cộng sự [12] đề xuất dựa trên quan sát từ đàn kiến thực. Trong thực tế tự nhiên, các con kiến bắt đầu tìm kiếm thức ăn một cách ngẫu nhiên. Mỗi con kiến sẽ chọn các hướng đi khác nhau và con kiến đi trên con đường ngắn nhất sẽ có xu hướng để lại vết mùi pheromone với nồng độ cao hơn các con đường khác, như vậy, các con kiến lân cận sẽ ngửi thấy vết mùi ở các đường đi có nồng độ cao và có xu hướng tham gia vào con đường ngắn nhất và tiếp tục làm tăng vết mùi lên. Các con kiến sẽ tiếp tục tham gia quá trình này cho đến khi phần lớn lượng vết mùi được hội tụ trên con đường ngắn nhất. Để áp dụng ACO, các bài toán tối ưu hóa được chuyển thành bài toán tìm con

đường tốt nhất trên một đồ thị có trọng số. Ning và Liu [24] đã sử dụng thuật toán MMAS như một cải tiến của thuật toán ACO để giải quyết bài toán CSLP. Gharaie [14] cùng các cộng sự đã áp dụng ACO để giải quyết bài toán CSLP tĩnh trong một dự án xây dựng.

Gulben Calis và Orhan Yuksel [6] trình bày một thuật toán lai giữa ACO và tìm kiếm địa phương (2-opt). Trong [7] Gulben Calis và Orhan Yuksel đề xuất một thuật toán kết hợp giữa ACO với Phân tích Parametric (PA) và tìm kiếm địa phương (2-opt) cho kết quả tốt hơn so với những nghiên cứu trước đây. Adrian [3] đã đề xuất cách thức lựa chọn tham số tối ưu cho 3 thuật toán GA, PSO và ACO. Qua thực nghiệm, ông kết luận rằng ACO được xem là nhanh nhất trong số ba thuật toán khi áp dụng vào bài toán CSLP.

1.7. Bài toán bố trí cơ sở theo hàng

Bài toán bố trí cơ sở theo hàng (Single row facility layout – SRFL) là một bài toán đặc biệt của bài toán vị trí cơ sở, giả sử các cơ sở được sắp xếp trên một hàng dài, mỗi cơ sở i có độ dài L_i ($L_i > 0$). Một ma trận kích thước $n \times n$ được ký hiệu là $C = (C_{ij})$ với C_{ij} là chi phí vận chuyển từ cơ sở i đến cơ sở j. Một phương án Π được gọi là một cách sắp xếp các cơ sở thành hàng theo thứ tự $\Pi = \{\pi_1, ..., \pi_n\}$ trong đó Π_i là vị trí i đặt cơ sở Π_i. Khi đó, chi phí được tính như sau:

$$Minimize \sum_{i}^{n-1} \sum_{j}^{n} c_{\pi_i \pi_j} d_{\pi_i \pi_j} \tag{1.24}$$

với

$$d_{\pi_i \pi_j} = l_{\pi_i} / 2 + \sum_{i<k<j} l_{\pi_k} + l_{\pi_j} / 2 \tag{1.25}$$

là khoảng cách giữa trung tâm cơ sở đến trong phương án sắp xếp. Mục tiêu của bài toán là tìm ra phương án sắp xếp sao cho hàm mục tiêu thu được là nhỏ nhất.

SRFL là một bài toán thuộc loại NP-khó và đã có nhiều phương pháp chính xác cũng như xấp xỉ được đề xuất giải. Các phương pháp chính xác được các tác giả công bố như: Thuật toán nhánh cận, thuật toán quy hoạch tuyến tính nguyên (integer programing), thuật toán quy hoạch động, phương pháp siêu phẳng cắt (cutting plane). Tuy nhiên, các phương pháp chính xác đều có nhược điểm chung là gặp khó khăn khi giải quyết một khối lượng lớn các tính toán và

bộ nhớ lưu trữ với số lượng cơ sở lớn. Chính vì vậy, các thuật toán heuristic và metaheuristic đã được đề xuất làm giảm chi phí tính toán và thời gian chạy của thuật toán mà vẫn cho những lời giải chấp nhận được. Solimanpur [34] đề xuất thuật toán ACO và PSO được Samarghandi đề xuất [31]. Feristah Ozcelik [27] đã đề xuất giải thuật lai giữa giải thuật di truyền và tìm kiếm địa phương. Sinem [33] đề xuất thuật toán đàn dơi giải bài toán nhanh cho lời giải tốt trong thời gian ngắn.

1.8. Kết luận chương

Trên đây chúng ta đã tìm hiểu các kiến thức tổng quan về độ phức tạp thuật toán, lớp các bài toán P, NP, NP-khó. Việc đánh giá một bài toán thuộc lớp nào là công đoạn đầu tiên và vô cùng quan trọng, nó góp phần giúp người lập trình định hướng và lựa chọn các thuật toán giải phù hợp cho bài toán.

Các bài toán điển hình trong lớp các bài toán vị trí cơ sở cùng các thuật toán đã được đề xuất giải các bài toán đó. Chi tiết việc đánh giá, so sánh hiệu năng của các thuật toán sẽ được trình bày trong chương 3.

CHƯƠNG 2
THUẬT TOÁN TỐI ƯU HÓA ĐÀN KIẾN

Tối ưu đàn kiến (ACO) là một phương pháp metaheuristic dựa trên ý tưởng mô phỏng cách tìm đường đi từ tổ tới nguồn thức ăn của các con kiến tự nhiên. Đến nay nó được cải tiến đa dạng và có nhiều ứng dụng [1]. Trước khi giới thiệu phương pháp ACO, cần giới thiệu phương thức trao đổi thông tin gián tiếp của các con kiến thực và mô hình kiến nhân tạo.

2.1. Từ kiến thực đến kiến nhân tạo

Khi tìm đường đi, đàn kiến trao đổi thông tin gián tiếp và hoạt động theo phương thức tự tổ chức. Mặc dù đơn giản nhưng phương thức này giúp cho đàn kiến có thể thực hiện được những công việc phức tạp vượt xa khả năng của từng con kiến, đặc biệt là khả năng tìm đường đi ngắn nhất từ tổ đến nguồn thức ăn mặc dù chúng không có khả năng đo độ dài đường đi. Trước hết ta xem các đàn kiến tìm đường đi như thể nào mà có thể giải quyết được các vấn đề tối ưu hóa.

2.1.1. Kiến thực

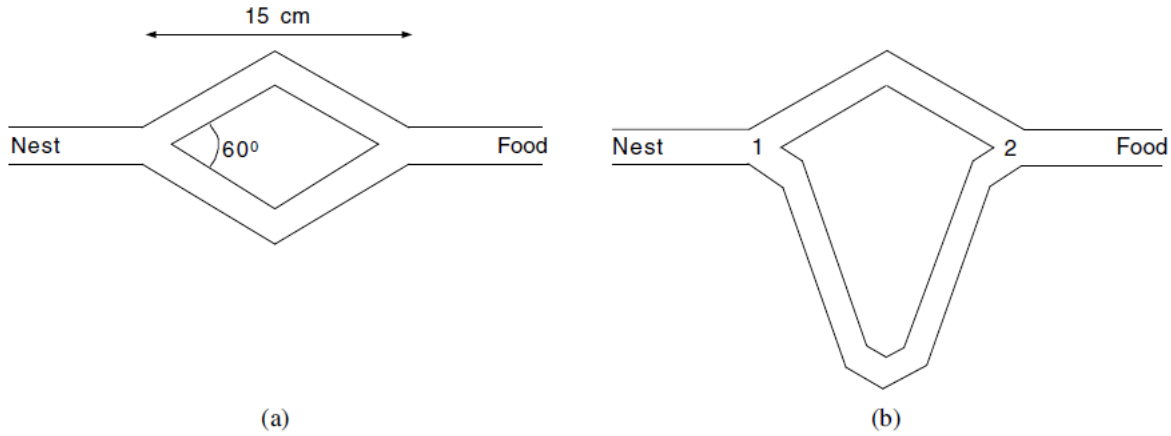
Trên đường, mỗi con kiến để lại một chất hóa học gọi là vết mùi (pheromon) dùng để đánh dấu đường đi. Bằng cách cảm nhận vết mùi, kiến có thể lần theo đường đi đến nguồn thức ăn được các con kiến khác khám phá theo phương thức chọn ngẫu nhiên có định hướng theo nồng độ vết mùi. Các con kiến chịu ảnh hưởng của các vết mùi của các con kiến khác là ý tưởng để thiết kế thuật toán ACO.

Thí nghiệm trên cây cầu đôi

Có nhiều thực nghiệm nghiên cứu về hành vi để lại vết mùi và đi theo vết mùi của loài kiến. Một thực nghiệm được thiết kế bởi Deneubourg và các đồng nghiệp (Deneubourg, Aron, Goss, & Pasteels, 1990; Goss et al., 1989) nhờ dùng một chiếc cầu đôi nối từ tổ của loài kiến tới nguồn thức ăn như minh họa trong hình 3.1. Họ đã thực nghiệm với tỉ lệ độ dài đường $r = \frac{l_l}{l_s}$ giữa hai nhánh của chiếc cầu đôi khác nhau, trong đó l_l là độ dài của nhánh dài còn l_s là độ dài của nhánh ngắn.

Trong thực nghiệm thứ nhất, chiếc cầu đôi có hai nhánh bằng nhau ($r = 1$, hình 3.1.a). Ban đầu, kiến lựa chọn đường đi một cách tự do đi từ tổ đến nguồn thức ăn, cả hai nhánh đều có kiến đi, nhưng sau một thời gian các con kiến này tập trung đi theo cùng một nhánh. Kết quả có thể được giải thích như

sau: Ban đầu không có vết mùi nào trên cả hai nhánh, do đó các kiến lựa chọn nhánh bất kỳ với xác suất như nhau. Một cách ngẫu nhiên, sẽ có một nhánh có số lượng kiến lựa chọn nhiều hơn nhánh kia. Do kiến để lại vết mùi trong quá trình di chuyển, nhánh có nhiều kiến lựa chọn sẽ có nồng độ mùi lớn hơn nồng độ mùi của nhánhcòn lại. Nồng độ mùi trên cạnh lớn hơn sẽ ngày càng lớn hơn vì ngày càng có nhiều kiến lựa chọn. Cuối cùng, hầu như tất cả các kiến sẽ tập trung trên cùng một nhánh.Thực nghiệm này cho thấylà sự tương tác địa phương giữa các con kiến nhờ thông tin gián tiếp qua vết mùi để lại mà có thể điều chỉnh hoạt động vĩ mô của kiến.



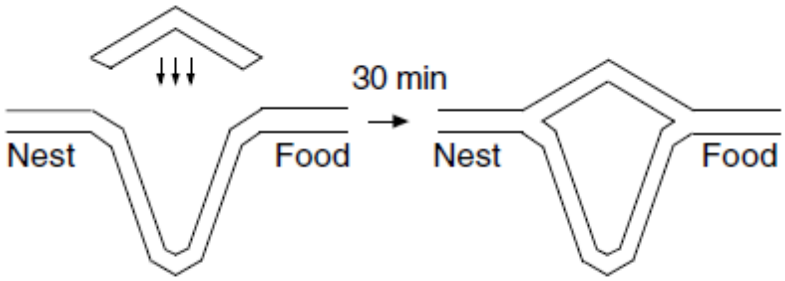
Hình 2.1. Thí nghiệm trên cây cầu đôi

(a) Hai nhánh có độ dài bằng nhau.

(b) Hai nhánh có độ dài khác nhau.

Trong thực nghiệm thứ hai (xem hình 2.1b), độ dài giữa nhánh dài hơn gấp đôi nhánh ngắn hơn (tỉ lệ $r = 2$). Trong trường hợp này, sau một thời gian tất cả các kiến chọn đi duy nhất theo đường ngắn hơn. Cũng như thực nghiệm thứ nhất, ban đầu kiến lựa chọn đường đi theo hai nhánh như nhau, ta có thể giả sử một nửa kiến đi theo nhánh ngắn và một nửa đi theo nhánh dài (mặc dù ngẫu nhiên nhưng vẫn có thể một nhánh sẽ có nhiều lựa chọn hơn nhánh khác). Tuy nhiên thực nghiệm này khác với thực nghiệm trên: bởi vì một nhánh ngắn hơn nhánh kia nên các kiến lựa chọn theo nhánh ngắn hơn sẽ nhanh chóng quay trở lại từ nguồn thức ăn về tổ và phải lựa chọn giữa nhánh ngắn và nhánh dài, nồng độ mùi trên nhánh ngắn đang cao hơn nồng độ mùi trên nhánh dài, do đó nó ưu tiên lựa chọn đi theo nhánh ngắn hơn. Thật thú vị, chúng ta có thể quan sát thấy rằng, với nhánh dài gấp đôi nhánh ngắn thì không phải tất cả các kiến cũng đều đi theo nhánh ngắn hơn và phải mất một thời gian nữa thì các kiến này sẽ đi theo nhánh ngắn hơn. Điều này được hiểu là một loại thăm dò, tìm đường mới.

Một điều thú vị nữa là điều gì sẽ xảy ra nếu sau khi hội tụ lại xuất hiện một đường mới từ tổ đến nguồn thức ăn. Việc này được thực nghiệm như sau: Ban đầu từ tổ đến nguồn thức ăn chỉ có một nhánh dài và sau 30 phút thì thêm một nhánh ngắn (xem hình 3.2). Trong trường hợp này, nhánh ngắn thường không được kiến chon mà chúng tập trung đi trên nhánh dài. Điều này có thể giải thích bởi nồng độ vết mùi trên cạnh dài cao và do sự bay hơi chậm của vết mùi nên đại đa số các con kiến lựa chọn các nhánh dài (vì nồng độ vết mùi cao) và hành vi này tiếp tục củng cố trên nhánh dài, ngay cả khi một nhánh ngắn xuất hiện. Việc bay hơi vết mùi có thể có lợi cho việc tìm đường mới, nghĩa là việc bay hơi có thể giúp kiến quên đi đường đi tối ưu cục bộ để tìm đường đi mới tốt hơn có thể được khám phá.



Hình 2.2. Thí nghiệm ban đầu chỉ một nhánh dài và sau 30 phút thêm nhánh ngắn

2.1.2. Kiến nhân tạo

Qua thực nghiệm cây cầu đôi ta nhận thấy rằng đàn kiến trong tự nhiên có thể sử dụng luật di chuyển theo xác suất dựa trên thông tin địa phương để tìm được đường đi ngắn nhất giữa hai địa điểm. Vết mùi của đàn kiến làm ta liên tưởng tới cách học tăng cường (reinforcement learning) cho bài toán chọn tác động tối ưu gợi mở một mô hình mô phỏng cho bài toán tìm đường đi ngắn nhất giữa hai nút (tương ứng là tổ và nguồn thức ăn) trên đồ thị, trong đó dùng các tác tử (agent) làm con kiến nhân tạo.

Tuy nhiên, trong các bài toán ứng dụng thì các đồ thị phức tạp hơn, từ mỗi đỉnh có nhiều cạnh nên nếu mô phỏng trung thực với kiến tự nhiên thì nhiều con kiến sẽ đi luẩn quẩn và hiệu quả sẽ rất kém. Vì vậy 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 sẽ gọi là kiến) có bộ nhớ riêng, có khả năng ghi nhớ các đỉnh đã thăm trong hành trình và tính được độ dài

đường đi nó chọn. Ngoài ra các con kiến có thể trao đổi thông tin có được với nhau, thực hiện tính toán cần thiết, cập nhật mùi…

Nhờ các con kiến nhân tạo này Dorigo [12] đã xây dựng hệ kiến (AS) giải bài toán người chào hàng, hiệu quả của nó so với các phương pháp mô phỏng tự nhiên khác như SA và GA đã được kiểm chứng bằng thực nghiệm và được phát triển và ứng dụng phong phú với tên gọi chung là phương pháp ACO.

2.2. Phương pháp ACO cho bài toán TƯTH tổng quát

2.2.1. Đồ thị cấu trúc

Xét bài toán tối ưu tổ hợp (TƯTH) tổng quát dưới dạng bài toán cực tiểu hoá (S, f, Ω), trong đó S là tập hữu hạn trạng thái, f là hàm mục tiêu xác định trên S còn Ω là các ràng buộc để xác định S qua các thành phần của tập hữu hạn C và các liên kết của tập này. Các tập S, C và Ω có các đặc tính sau.

1) Ký hiệu X là tập các xâu trong C độ dài không quá h: $X = \{< u_0, \ldots, u_k > u_i \in C \ \forall i \leq k \leq h\}$, khi đó mỗi phương án s trong S được xác định nhờ ít nhất một xâu trong X như ở điểm 2.

2) Tồn tại tập con X^* của X và ánh xạ φ từ X^* lên S sao cho $\varphi^{-1}(s)$ không rỗng với mọi $s \in S$. Trong đó tập X^* có thể xây dựng được từ tập con C_0 nào đó của C nhờ mở rộng tuần tự dưới đây.

3) Từ C_0 mở rộng được thành X^* theo thủ tục tuần tự:

   i) $x_0 = < u_0 >$ là mở rộng được với mọi $u_0 \in C_0$.

   ii) Nếu $x_k = < u_0, \ldots, u_k >$ là mở rộng được thì từ các ràng buộc Ω xác định được tập con $J(x_k)$ của C sao cho với mọi $u_{k+1} \in J(x_k)$ để $x_{k+1} = < u_0, \ldots, u_k, u_{k+1} >$ là mở rộng được hoặc $x_k \in X^*$ khi $J(x_k)$ là rỗng.

   iii) Với mọi $u_0 \in C_0$, thủ tục mở rộng nêu trên xây dựng được mọi phần tử của X^*.

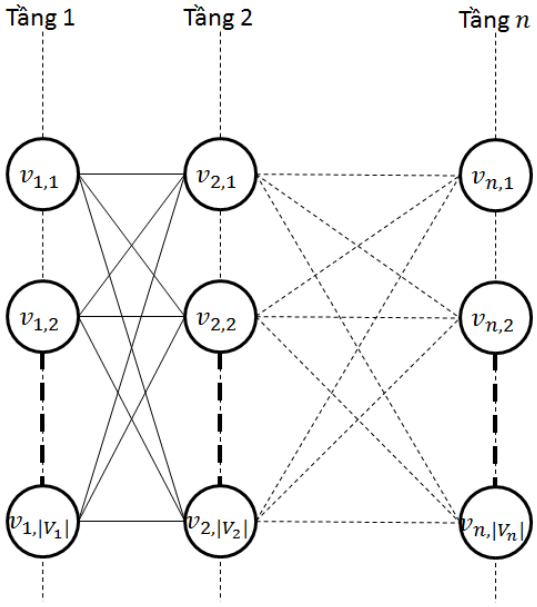
Mỗi bài toán TƯTH được xem như một bài toán tìm kiếm xâu độ dài không quá h trên đồ thị đầy có các đỉnh có nhãn trong tập C. Để tìm các lời giải chấp nhận được, ta xây dựng đồ thị đầy với tập đỉnh V mà mỗi đỉnh của nó tương ứng với mỗi thành phần của C. Các lời giải chấp nhận được sẽ là các xâu được tìm theo thủ tục tuần tự hay là bước ngẫu nhiên như mô tả chi tiết trong mục 3.2.2.

Thông thường, đối với các 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 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 $i, j \in C$ có trọng số heuristic $h_{i,j}$ để định hướng chọn thành phần mở rộng là j khi thành phần cuối của x_k là i theo thủ tục tuần tự nêu trên ($h_{i,j} > 0 \ \forall(i,j)$). Ký hiệu H là vectơ các trọng số heuristic của cạnh tương ứng (trong bài toán TSP nó có thể là vectơ mà thành phần là nghịch đảo độ dài của cạnh tương ứng), còn τlà vectơ biểu thị các thông tin học tăng cường $\tau_{i,j}$ (về sau gọi là vết mùi, ban đầu được khởi tạo bằng τ_0>0) định hướng mở rộng x_k với thành phần cuối là i nhờ thêm thành phần j theo thủ tục tuần tự. Trường hợp đặc biệt, $h_{i,j}$ và $\tau_{i,j}$ 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 các thong tin này ở các cạnh.

Khi đó ta gọi đồ thị $G = (V, E, H, \tau)$ là *đồ thị cấu trúc* của bài toán tối ưu tổ hợp đang xét, trong đó V là tập đỉnh, H và τlà các thông tin đã nói ở trên còn E là tập cạnh nào đó của đồ thị sao cho từ các cạnh này có thể xây dựng được tập X^* nhờ mở rộng tập C_0 theo thủ tục tuần tự. Nếu không có thông tin heuristics thì ta xem H có các thành phần như nhau và bằng 1.

Trường hợp tổng quát, G là đồ thị đầy, tuy nhiên tùy theo ràng buộc của bài toán mà các cạnh có thể giảm bớt để giảm miền tìm kiếm lời giải theo thủ tục tuần tự. Chẳng hạn, với bài toán tìm cực trị của hàm giải tích $f(x_1, \ldots, x_n)$ trong đó x_i thuộc tập giá trị hữu hạn V_i thì đồ thị cấu trúc có tập đỉnh $V=\bigcup_{i=1}^{n} V_i$, các đỉnh phân bố thành n tầng và tầng I chứa các đỉnh thuộc tập V_i còn tập cạnh E chỉ gồm các cạnh nối các đỉnh thuộc tầng i với các đỉnh thuộc tầng $i+1$ ($i = 1, 2, .., n-1$) như trong hình 3.3. Khi đó tập C_0 là tập V_1, mỗi lời giải tuần tự sẽ được mở rộng từ một đỉnh thuộc tập này.



Hình 2.3.Đồ thị cấu trúc tổng quát cho bài toán cực tri hàm $f(x_1,\ldots x_n)$

Với đồ thị cấu trúc đã xác định, thuật toán ACO tương ứng thực hiện theo lược đồ dưới đây.

2.2.2. Mô tả thuật toán ACO tổng quát.

Với điều kiện kết thúc đã chọn (có thể là số bước lặp hoặc/và thời gian chạy cho trước), người ta dùng đàn kiến m con thực hiện lặp xây dựng lời giải trên đồ thị cấu trúc $G = (V, E, H, \tau)$ như sau: Trong mỗi lần lặp, mỗi con kiến chọn ngẫu nhiên một đỉnh $u_0 \in C_0$ làm thành phần khởi tạo $x_0 = \{u_0\}$ 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.

Thủ tục bước ngẫu nhiên:

Giả sử $x_k =< u_0,\ldots,u_k >$ là mở rộng được, từ các ràng buộc Ω xác định được tập con $J(x_k)$ của C sao cho với mọi $u_{k+1} \in J(x_k)$ thì $x_{k+1} =< u_0,\ldots,u_k,u_{k+1} >$ là mở rộng được hoặc $x_k \in X^*$ khi $J(x_k)$ là rỗng. Đỉnh $j = u_{k+1}$ để mở rộng được chọn với xác suất $P(j)$ như sau:

$$P(j) = \begin{cases} \dfrac{[\tau_{ij}]^{\alpha}[h_{ij}]^{\beta}}{\sum_{l \in J(x_k)}[\tau_{il}]^{\alpha}[h_{il}]^{\beta}} & j \in J(x_k) \\ 0 & j \in \overline{J}(x_k) \end{cases} \qquad (2.1)$$

Quá trình mở rộng tiếp tục cho tới khi kiến r tìm được lời giải chấp nhận được $x(r)$ trong X^* và do đó $s(r) = \varphi(x(r)) \in S$. Khi đó ta nói cạnh (i, y) thuộc $x(r)$.

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. Lượng mùi cập nhật theo mỗi quy tắc cập nhật mùi khác nhau cho ta các thuật toán khác nhau. Vì vậy, quy tắc cập nhật mùi thường dùng làm tên gọi thuật toán. Đa số trong chúng đều có dạng:

$$\tau_{i,j} \leftarrow (1-\rho)\tau_{i,j} + \Delta(i,j), \forall\ (i,j) \qquad (2.2)$$

đối với các cạnh được cập nhật, trong đó ρ là hằng số thuộc khoảng (0,1) là tỷ lệ lượng mùi bị bay hơi.

Các bước thực hiện của các thuật toán ACO được đặc tả trong hình 2.4.

```
Procedure of ACO algorithms;
Begin
   Initialize;//  khởi tạo ma trận mùi, khởi tạo m con kiến
   Repeat
      Construct solutions;      // mỗi con kiến xây dựng lời giải
      Update trail;          // cập nhật mùi
   Until End condition;          //điều kiện kết thúc
End;
```

Hình 2.4. Đặc tả thuật toán ACO

Nhận xét chung về các thuật toán ACO

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

1) 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 sẵn có, do đó cho ta lời giải tốt hơn và có thể tìm được lời giải tối ưu.

2) 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.

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

1) *Xây dựng đồ thị cấu trúc thích hợp*. Trong mục 3.2.1 ta đã chỉ ra rằng việc xây dựng đồ thị cấu trúc để tìm được lời giải cho bài toán theo thủ tục tuần tự không khó. Khó khăn chính là với các bài toán cỡ lớn thì không gian tìm kiếm quá rộng, đòi hỏi ta sử dụng các ràng buộc Ω một cách hợp lý để giảm miền tìm kiếm cho mỗi con kiến.

2) *Chọn thông tin heuristic*. Thông tin heuristic tốt sẽ tăng hiệu quả thuật toán. Tuy nhiên, nhiều bài toán ta không có thông tin này thì có thể đánh giá chúng như nhau. Khi đó lúc ban đầu, thuật toán chỉ đơn thuần chạy theo phương thức tìm kiếm ngẫu nghiên, vết mùi thể hiện định hướng của học tăng cường và thuật toán vẫn thực hiện được.

3) *Chọn quy tắc cập nhật mùi*. Quy tắc cập nhật mùi thể hiện chiến lược học của thuật toán. Nếu đồ thị cấu trúc và thông tin heuristics luôn phụ thuộc vào từng bài toán cụ thể thì quy tắc cập nhật mùi là yếu tố phổ dụng và thường dùng để đặt tên cho thuật toán.

Để hiểu rõ phương pháp ACO, dưới đây giới thiệu các thuật toán ACO điển hình giải bài toán người chào hàng.

2.3. Phương pháp ACO giải bài toán TSP

Bài toán người chào hàng (Traveling Salesman Problem: TSP) là bài toán có nhiều ứng dụng trong thực tế. Nó thuộc loại NP-khó và được xem là bài toán chuẩn để đánh giá hiệu quả của các thuật toán giải các bài toán TƯTH mới. Thuật toán ACO đầu tiên được gọi là hệ kiến (Ant System: AS), các thuật toán ACO về sau là cải tiến của AS và đều dùng bài toán TSP để thử nghiệm chất lượng.

2.3.1. Bài toán TSP và đồ thị cấu trúc

Bài toán TSP xuất phát từ thực tế, một người giới thiệu sản phẩm muốn tìm một hành trình ngắn nhất xuất phát từ thành phố của mình đi qua tất cả các thành phố mà khách hàng cần giới thiệu sản phẩm vàsau đó trở về thành phố xuất phát với điều kiện các thành phố của khách hàng chỉ đi qua đúng một lần.

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 có trọng số $G = (N, A)$, trong đó N là tập các đỉnh tương ứng với tập các thành phố, A là tập các 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 đểhành trình tối ưu trên G' cũng là hành trình tối ưu trên G. Ta ký hiệu độ dài mỗi cạnh $(i,j) \in A$ là d_{ij} tương ứng với khoảng cách giữa thành phố i và thành phố j (với mọi $i,j \in N$). Trong trường hợp tổng quát là bài toán TSP được xét trên đồ thị có hướng vàkhoảng cách giữa cặp đỉnh i,j có thể phụ thuộc vào hướng của cạnh, khicó ít nhất một cạnh mà $d_{ij} \neq d_{ji}$ thì ta nói bài toán là không đối xứng và ký hiệu là ATSP, ngược lại thì là bài toán đối xứng (luôn có $d_{ij} = d_{ji}\ \forall i,j$).

Nhắc lại rằng 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 $f(\pi)$ là nhỏ nhất, trong đó $f(\pi)$ bằng:

$$f(\pi) = \sum_{i=1}^{n-1} d_{\pi(i)\pi(i+1)} + d_{\pi(n)\pi(1)} \qquad (2.3)$$

Thông tin heuristic trên cạnh (i,j) theo truyền thống thay cho h_{ij}sẽ ký hiệu là η_{ij} và là nghịch đảo của độ dài cạnh : $\eta_{ij} = \frac{1}{d_{ij}}\ \forall\ (i,j)$. Trong tất cả các thuật toán ACO giải bài toán TSP, vết mùi được đính trên các cạnh và do đó τ_{ij} dùng để chỉ thông tin học tăng cường cho mở rộng trực tiếp lời giải từ i đến j.

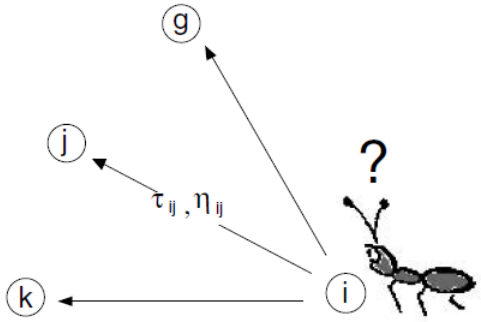
Khi đó đồ thị cấu trúc của bài toán là đồ thị đầy G = (N, E, H,τ). Nếu $x_k =< u_0,\ldots,u_k >$ là đường đi mở rộng được, tức là các đỉnh u_i đều khác nhau và $k < n$ thì $J(x_k) = N_{u_k}$ là các đỉnh mà đường x_k chưa đi đến. Các thuật toán ACO cho bài toán TSP đã có đều thực hiệu trên đồ thị cấu trúc này. Các kết quả thực nghiệm [12] đã chỉ ra hiệu quả nổi trội của phương pháp này so với các tiếp cận khác.

2.3.2. Các thuật toán ACO cho bài toán TSP

Thuật toán ACO có thể áp dụng trực tiếp cho bài toán TSP với đồ thị cấu trúc $G = (V,E,H,\tau)$ đã nói ở trên có không gian tìm kiếm là các hành trình có thể. Ràng buộc Ω thỏa mãn khi mà hành trình do kiến xây dựng là một hành trình đúng, tức là chu trình Hamilton biểu thị một hoán vị chỉ số các thành phố.

Quá trình mỗi con kiến xây dựng lời giả theo thủ tục bước ngẫu nhiên như sau:

- Lựa chọn thành phố xuất phát cho kiến (có thể theo một số tiêu chí nào đó).
- Thực hiện lặp thủ tục mở rộng bằng cách lặp đi lặp lại việc thêm một thành phố mà kiến chưa đi qua (xem hình 3.5) cho đến khi tất cả các thành phố đều được thăm: tính xác suất lựa chọn đỉnh mới nhờ giá trị thông tin mùi và thông tin heuristic rồi chọn ngẫu nhiên đỉnh mới thêm vào theo phân bố ngẫu nhiên này.
- Quay trở lại thành phố xuất phát.



Hình 2.5. Lựa chọn đỉnh đi tiếp theo khi kiến

Sau khi tất cả các kiến xây dựng xong hành trình, các kiến sẽ để lại vết mùi trên các cạnh mà kiến đã đi. Trong một số trường hợp, trước khi thêm mùi, các hành trình xây dựng được có thể cải tiến bằng cách áp dụng thủ tục tìm kiếm cục bộ. Lược đồ thuật toán ACO giải bài toán TSP có áp dụng tìm kiếm cục bộ (theo lược đồ thuật toán memetic trong chương trước) được đặc tả trong hình 2.6

```
Procedure Thuật toán ACOTSP
Begin
   Initialize: Khởi tạo vết mùi
   while Khi điều kiện dừng chưa thỏa mãn do
        for i=1 to n_ants do
                Xây dựng lời giải;
                Cải tiến lời giải do kiến xây dựng bằng tìm kiếm cục bộ;
        Cập nhật lời giải tốt
        Cập nhật mùi
   end while
End
```

Hình 2.6. Đặc tả thuật toán ACO giải bài toán TSP.

Thuật toán ACO đầu tiên là thuật toán AS (Dorigo năm 1991 được giới thiệu để giải bài toán TSP đã đạt được hiệu quả khả quan ban đầu nhưng chưa phải là thuật toán tốt nhất cho bài toán này. Tuy nhiên AS đóng vai trò rất quan trọng, các thuật toán mở rộng, cải tiến của thuật toán ACO sau này đều dựa trên thuật toán AS. Sự khác biệt chính giữa *c*ác thuật toán mở rộng của AS như thuật toán elitist AS, rank-based AS và MAX-MIN…là cách thức cập nhật mùi cũng như việc quản lý vết mùi. Bảng 2.1 cho thấy đa số các thuật toán ACO mới đều được thử nghiệm trên TSP trừ thuật toán ANTS và hyper-cube framework. Trong bảng này, dòng giao với cột TSP là có biểu thị thuật toán tương ứng có áp dụng giải bài toán TSP và không có nghĩa không thử nghiệm cho TSP.

Bảng 2.1.Thuật toán ACO theo thứ tự thời gian xuất hiện

Thuật toán ACO	**TSP**	**Tác giả và thời gian công bố**
Ant System (AS)	Có	Dorigo (1992); Dorigo, Maniezzo & Colorni (1991, 1996)
Elitist AS	Có	Dorigo (1992); Dorigo, Maniezzo & Colorni (1991, 1996)
Ant-Q	Có	Gambardella & Dorigo(1995); Dorigo & Gambardella (1996)
Ant Colony System	Có	Dorigo & Gambardella (1997a,b)
Max-Min Ant System	Có	Stützle & Hoos (1996, 2000); Stützle (1999)
Rank-based AS	Có	Bullnheimer, Hartl & Strauss (1997, 1999c)
ANTS	Không	Maniezzo (1999)
Hyper-cube AS	Không	Blum, Roli, & Dorigo (2001); Blum & Dorigo (2004)

Trong các thuật toán ACO hiện nay thông dụng nhất là hệ kiến *MAX-MIN* (MMAS) và ACS, tuy hiệu quả của chúng như nhau nhưng *MAX-MIN* dễ dùng hơn còn ACS được nhóm của Dorigo quan tâm hơn. Dưới đây, chúng tôi giới thiệu các thuật toán AS, MMAS và ACS theo trình tự thời gian xuất hiện.

2.3.2.1. 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 nhật 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-density 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 giả tìm được bằng phương pháp heuristic nào đó (có thể dung 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:$\tau_{ij} = \tau_0 = \frac{m}{C^{nn}} \forall (i,j)$, trong đó m là số kiến, C^{nn} là độ dài lời giải tìm được của thuật toán heuristic. Lý do cho việc lựa chọn này là nếu khởi tạo vết mùi τ_0 quá thấp thì việc tìm kiếm có khuynh hướng nhanh chóng hội tụ quanh những hành trình đầu tiên tìm được, đẫn 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:

$$p_{ij}^k = \begin{cases} \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in N_i^k} [\tau_{il}]^\alpha [\eta_{il}]^\beta}, \text{nếu} j \in N_i^k \\ 0, \text{nếu} j \in \overline{\in} N_i^k \end{cases} \quad (2.4)$$

Trong đó $\eta_{ij} = \frac{1}{d_{ij}}$ là giá trị heuristic như đã nói ở trên, α, β là hai tham số quyết định đến sự ảnh hưởng tương quan giữa thông tin mùi và thông tin heuristic, N_i^k là các đỉnh lân cận của đỉnh 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 N_i^k bằng 0). Theo quy tắc ngẫu nhiên này, xác suất lựa chọn cạnh (i,j) tăng theo giá trị thông tin mùi τ_{ij} và thông tin heuristic η_{ij}. Vai trò của hai tham số α, β như sau: nếu $\alpha = 0$, thành phố gần nhất sẽ được lựa chọn nhiều hơn, khi đó thuật toán tương đương với thuật toán chọn ngẫu nhiên theo nghịch đảo độ dài cạnh mà không có học tăng cường. Nếu $\beta = 0$, chỉ có thong tin học tăng cườngbiểu thị

qua vết mùi được sử dụng màkhông có thông tin heuristic. Với $\alpha > 1$, thuật toán nhanh chóng bị tắc nghẽn (tất cả các kiến sẽ lựa chọn cũng một hành trình) và lời giả tìm được hội tụ về lời giải tối ưu cục bộ (Dorigo và cộng sự [12]).

Để cài đặt, mỗi kiến k sẽ duy trì một bộ nhớ M^k 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 N_i^k. Hơn nữa, thông tin trong bộ nhớ M^k giúp cho kiến tính được độ dài hành trình T^k 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:

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij} \quad \forall (i,j) \in E, \tag{2.5}$$

Trong đó $0 < \rho \leq 1$ 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 vết mùi mà nó đi qua:

$$\tau_{ij} \leftarrow \tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^k \quad \forall (i,j) \in E, \tag{2.6}$$

Trong đó $\Delta\tau_{ij}^k$ là lượng mùi do kiến k cập nhật trên cạnh mà kiến k đi qua. Giá trị này bằng:

$$\Delta\tau_{ij}^k = \begin{cases} \frac{1}{C^k} \; nếu \; cạnh \; (i,j) thuộc \; T^k \\ 0 \; ngược \; lại \end{cases} \tag{2.7}$$

Trong đó C^k là độ dài hành trình T^k do kiến k xây dựng, giá trị này được tính bằng tổng độ dài các cạnh thuộc hành trình. Theo công thức (2.6), 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 chung cải tiến thuật toán AS.

2.3.2.2. Hệ đàn kiến (ACS)

Thuật toán ACS (Dorigo & Gambardella [11]) khác với AS ở ba điểm chính. Thứ nhất, đó là sự khai thác 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 vàolời 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 đến 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 thuật toán ACS, khi kiếm k đang đứng ở đỉnh i lựa chọn di chuyển đến đỉnh j theo qui tắc:

$$j = \begin{cases} argmax_{l \in N_i^k}\{\tau_{il}[\eta_{il}]^\beta\}, nếu\ q \le q_0 \\ J, ngược\ lại \end{cases} \qquad (2.8)$$

trong đó q là một biến ngẫu nhiên phân bố đều trong [0,1], q_0 $(0 \le q_0 \le 1)$ là một tham số cho trước và J là một biến ngẫu nhiên lựa chọn theo phân bố xác suất như trong (2.4) với $\alpha = 1$. Hay nói cách khác, với xác suất q_0 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 $(1 - q_0)$ kiến thực hiện khám phá trên các cạnh. Điều chỉnh tham số q_0 cho phép thay đổi mức độ khai tác 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 ant) được phép để lại mùi sau mỗi lần lặp. Việc cập nhật mùi trong ACS được thực hiện như sau:

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij} + \rho\Delta\tau_{ij}^{bs}, \forall(i,j) \in T^{bs}, \qquad (2.9)$$

trong đó $\Delta\tau_{ij}^{bs} = \frac{1}{C^{bs}}$, C^{bs} là độ dài lời giải tốt nhất, T^{bs} 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 T^{bs}, 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 $O(n^2)$ còn $O(n)$ trong đó n là số thành phố. Tham số ρ là tham số bay hơi. Không giống như AS , trong (2.5) và (2.6) trong (2.9) vết mùi được 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 giả tốt nhất trong bước lặp (iteration-best: i-best) để cập nhật mùi.Với các bộ dữ liệu TSP nhỏ thì việc sử dụng iteration-best và best-so-far không nhiều, nhưng với dữ liệu lớn (số thành phố lớn 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 cục bộ được thực hiện ngay lập tức khi cạnh (i,j) có kiến đi qua theo công thức:

$$\tau_{ij} \leftarrow (1-\xi)\tau_{ij} + \xi\tau_0, \qquad (2.10)$$

trong đó $\xi(0 < \xi < 1)$ và τ_0 là hai tham số. Giá trị τ_0 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 τ_0 là $\frac{1}{nC^{nn}}$, trong đó n là số thành phố, C^{nn} 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 mùi 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 sự lựa chọn nào tốt hơn.

Tồn tại một quan hệ thú vị giữa MMAS và ACS: cả hai thuật toán đều sử dụng giới hạn vết mùi mặc dù trong ACS không rõ ràng như trong MMAS. Trên thực tế, trong ACS vết mùi không bao giờ nhỏ hơn τ_0 bởi vì khi cập nhật theo hai công thức (2.9) và (2.10) thì vết mùi luôn lớn hơn hoặc bằng τ_0 và việc khởi tạo mùi ban đầu là τ_0. Hơn nữa, vết mùi không bao giờ vượt quá $\frac{1}{C^{bs}}$. Do đó, vết mùi $\tau_0 \leq \tau_{ij} \leq \frac{1}{C^{bs}}$.

Cuối cùng, ACS là thuật toán ACO 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 các 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 dần 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 trong số các ứng cử viên chưa được thăm, trong trường hợp tất cả các thành phố trong danh sách ứng cử viên đều 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.

2.3.2.3. Hệ kiến Max-Min

Thuật toán Max-Min Ant System – MMAS được Stutzle và Hoos [36]) đề xuất với bốn điểm 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 là đưa ra miền giới hạn cho vết mùi, vết mùi sẽ thuộc $[\tau_{min}, \tau_{max}]$.

Thứ ba là vết mùi ban đầu được khởi tạo bằng τ_{max} 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 (công thức 2.4), sau đó được thêm một lượng mùi như sau:

$$\tau_{ij} \leftarrow \tau_{ij} + \Delta\tau_{ij}^{best} \qquad (2.11)$$

trong đó $\Delta\tau_{ij}^{best} = \frac{1}{C^{best}}$. Kiến được lựa chọn để thêm mùi có thể là G-best (khi đó $\Delta\tau_{ij}^{best} = \frac{1}{C^{bs}}$) hoặc iteration-best ant (khi đó $\Delta\tau_{ij}^{best} = \frac{1}{C^{ib}}$, trong đó C^{ib}là độ dài hành trình của i-best ant).Sau đó vết mùi sẽ bị giới hạn trong đoạn $[\tau_{min}, \tau_{max}]$ như sau:

$$\tau_{i,j} = \begin{cases} \tau_{max}\ nếu\ \tau_{i,j} > \tau_{max} \\ \tau_{i,j}\ nếu\ \tau_{i,j} \in [\tau_{min}, \tau_{max}] \\ \tau_{min}\ nếu\ \tau_{i,j} < \tau_{min} \end{cases} \qquad (2.12)$$

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 định hướngquanh T^{bs}, 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 khi chỉ sử dụng i-best để cập nhật. Trong khi với bộ dữ liệu TSP lớn với hàng trăm thành phố, hiệu quả tốt bằng cách tăng cường sự cập nhật bằng G-best. Điều này có thể thực hiện được bằng cách tăng cường tần suất sử dụng T^{bs} để cập nhật (Stutzle 1999).

Giới hạn vết mùi

Trong MMAS, giới hạn trên τ_{max} và giới hạn dưới τ_{min} cho vết mùi trên tất cả các cạnh để tránh tình trạng tắc nghẽn. Đặc biệt việc giới hạn vết mùi có ảnh hưởng đến giới hạn xác suất p_{ij} trong đoạn $[p_{min}, p_{max}]$cho lựa chọn đỉnh j khi kiến đang ở đỉnh i, với $0 < p_{min} \leq p_{ij} \leq p_{max} \leq 1$. Chỉ khi $\left|N_i^k\right| = 1$ thì $p_{min} = p_{max} = 1$

Dễ thấy, nếu chạy một trong thời gian dài thì cận trên của vết mùi là $\frac{1}{\rho C^*}$ trong đó C^* là độ dài hành trình tối ưu. Dựa trên kết quả đó, MMAS đặt τ_{max} bằng $\frac{1}{\rho C^{bs}}$, mỗi khi tìm được best-so-far tour mới τ_{max} được cập nhật lại. Cận dưới $\tau_{min} = \frac{\tau_{max}}{a}$ trong đó a là một tham số. Kết quả thực nghiệm chỉ ra rằng: để tránh tắc nghẽn cận dưới τ_{min} đóng vai trò quan trọng hơn τ_{max}. Tuy nhiên, τ_{max} 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 thiết đặt bằng ước lượng cận trên của vết mùi τ_{max}. Cách khởi tạo như vậy kết hợp với tham số bay hơi nhỏ làm chậm sự khác biệt 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 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.

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

2.4. Một số vấn đề khác khi áp dụng ACO

2.4.1. Đặc tính hội tụ

Gutjahr [15] khởi đầu cho nghiên 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 rằng với xác suất bằng 1 ta có :

1) $\lim_{t\to\infty} w(t) = w^*, \lim_{t\to\infty} P(t) = 1$ (2.13)

2) $\lim_{t\to\infty} \tau_{i,j} = \tau_{max}$ với mọi cạnh (i, j) thuộc lời giải tối ưu tìm được. (2.14)

Mô hình này của Gutjahr không áp dụng được cho ACS. Trường hợp MMAS không có thông tin heuristic, Stützle và Dorigo [35] đã chứng minh rằng:

$\forall \varepsilon > 0$, với t đủ lớn thì P(t)>1-ε, (2.15)

do đó: $\lim_{t\to\infty} P(t) = 1.$ (2.16)

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, Stützle và Dorigo suy ra rằng vết mùi của các cạnh thuộc lời giải tối ưu tìm được hội tụ đến τ_{max} còn vết mùi trên các cạnh không thuộc lời giải này hội tụ về τ_{min} hoặc τ_0. Suy luận này là tầm thường.

Plelegrini và Elloro [28] đã nhận thấy sau một thời gian chạy thì đa số vết mùi trên cạnh là bé và chỉ số ít là lớn nổi trội.

2.4.2. Thực hiện song song

Đặc tính tự nhiên của các thuật toán ACO giúp cho chúng có thể thực hiện song song theo dữ liệu hoặc theo quần thể. Trên thực tế, có nhiều mô hình song song được sử dụng cho các thuật toán dựa trên quần thể có thể dễ dàng tương thích với ACO. Hầu hết các chiến lược song song trực tiếp có thể chia thành chiến lược mịn (fine-grained) và thô (coarse-grained). Đặc tính của fine-grained là rất ít bộ xử lý được chỉ định để xử lý đơn và việc trao đổi thông tin giữa các bộ xử lý thường xuyên. Ngược lại, với coarse-grained thì một lượng lớn, thậm chí tất cả bộ xử lý được chỉ định để xử lý đơn và thông tin trao đổi là rất ít.

Mô hình song song fine-grained đã được Bolondi & Bondanza nghiên cứu cho AS giải TSP trên máy CM-2 kết nối thông qua cách tiếp cận gán một bộ xử lý cho mỗi kiến. Kết quả thực nghiệm cho thấy trao đổi thông tin lớn có thể là vấn đề đối với cách tiếp cận này vì phần lớn thời gian dùng để liên lạc nhằm cập nhật vết mùi. Kết quả tương tự được Bullnheimer, Kotsis, and Strauss đưa ra.

Nhiều tác giả (Bolondi & Bondanza; Bullnheimer và cộng sự,; Kruger, Merkle, & Middendorf; Middendorf, Reischle, & Schmeck; Stutzle) đã nghiên cứu mô hình song song cho chiến lược coarse-grained và cho thấy có nhiều hứa hẹn hơn cho ACO. Trong trường hợp này, p đàn kiến chạy song song trên p bộ vi xử lý.

Stutzle đã thực hiện theo cách không có sự truyền thông giữa các đàn kiến. Việc này tương đương với chạy độc lập song song của nhiều thuật toán ACO vàlà cách dễ nhất để song song thuật toán ngẫu nhiên. Các kết quả tính toán của Stutzle chỉ ra rằng phương pháp này là rất hiệu quả.

Một số nhà nghiên cứu khác đã xem xét trường hợp thông tin được các đàn kiến được trao đổi trong khoảng thời gian nhất định.Ví dụ, Bullnheimer đề xuất thực hiện một phần không đồng bộ song song (Papi). Trong Papi, thông tin mùi được trao đổi giữa các đàn kiến sau một số lần lặp cố định và tăng cường qua quan sát thực nghiệm. Kruger đã đưa ra thông tin cần trao đổi giữa các đàn kiến và cách thức để cập nhật mùi. Kết quả thực nghiệm cho thấy việc trao đổi lời giải tốt nhất tìm được và dùng để cập nhật mùi tốt hơn việc trao đổi ma trận mùi. Middendorf đã phát triển tiếp của Michel & Middendorf để đưa ra cách trao đổi lời giải của m đàn kiến, cho phép trao đổi thông tin sau một số bước cố định. Thông tin trao đổi bao gồm: (1) lời giải tốt nhất tìm được của tất cả các đàn kiến; (2) lời giải tốt nhất G-best hoặc lời giải i-besthoặc cả hai được gửi cho đàn kiến lân cận, trong đó lân cận là trực tiếp theo vòng. Kết quả chính đạt được là hạn chế được thông tin trao đổi.

Một số mô hình thực hiện song song cho thuật toán kiến trên kiến trúc chia sẻ bộ nhớ sử dụng OpenMP (Chandra, Dagum, Kohr, Maydan, McDonald, & Menon) cúng đã được khảo sát.

2.4.3. ACO kết hợp với 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 cách tiếp cận đầu 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 khả năng kết hợp tìm kiếm cục cải tiến được lời giải là khá cao.

2.4.3.1. Thông tin heuristic

Chúng ta biết rằng khi thuật toán ACO áp dụng giải TSP mà không sử dụng tìm kiếm cục bộ, thông tin heuristic là điều rất cần thiết để cho lời giải tốt.

Trên thực tế, trong giai đoạn đầu vết mùi được khởi tạo như nhau, vết mùi không thể giúp kiến nhân tạo tìm đường đi dẫn tới các lời giải tốt vì chúng chưa khác nhau nhiều. Vai trò chính của thông tin heuristic là tránh điều này, nó giúp kiến có thể xây dựng được hành trình tốt ngay trong giai đoạn đầu. Nhiều trường hợp, nhờ tìm kiếm cục bộ nên kiến vẫn có thể tìm được lời giải tốt trong giai đoạn đầu mà không cần sử dụng thông tin heuristic nào cả, mặc dù có chậm hơn. Do đó, thông tin heuristic có thể không còn quá cần thiết.

2.4.3.2. Số lượng kiến

Như đã nói ở trên, nếu không sử dụng tìm kiếm cục bộ và thông tin heuristic ít (hoặc không có), trong giai đoạn đầu vết mùi không thể giúp kiến nhân tạo tìm đường đi dẫn tới các lời giải tốt. Nếu sử dụng số lượng kiến ít, trong giai đoạn đầu sẽ không tìm được lời giải tốt và như vậy việc cập nhật mùi được cập nhật dựa trên các lời giải không tốt. Khi đó sẽ hướng việc tìm kiếm xung quanh lời giải không tốt và thuật toán sẽ không hiệu quả. Có thể khắc phục phần nào nhược điểm này bằng cách tăng số kiến để tăng khả năng tìm được lời giải tốt trong mỗi vòng lặp. Trường hợp có sử dụng tìm kiếm cục bộ hoặc thông tin heuristic mạnh việc sử dụng nhiều kiến là lãng phí. Do đó, theo kinh nghiệm của các tác giả khi làm thực nghiệm là khi có sử dụng tìm kiếm cục bộ hoặc có thông tin heuristic mạnh thì số kiến thường đặt từ 10 đến 30 kiến, trong trường hợp ngược lại số kiến thường tăng theo với kích thước bài toán.

2.4.3.3. Tham số bay hơi

Nếu trong mỗi vòng lặp, kiến có thể xây dựng được lời giải tốt (sử dụng tìm kiếm cục bộ hoặc thông tin heuristic mạnh) thì tham số bay hơi đặt lớn để giúp cho kiến quên đi những lời giải đã xây dựng tập chung tìm kiếm quanh lời giải tốt mới được xây dựng. Trong trường hợp ngược lại, nếu trong mỗi vòng lặp khả năng kiến tìm được lời giải tốt ít thì tham số bay hơi đặt nhỏ.

2.5. Kết luận chương

Phương pháp ACO là một phương pháp metaheuristic đang được sử dụng rộng rãi để giải các bài toán TƯTH khó và hiệu quả nổi trội của chúng đã được chứng tỏ bằng thực nghiệm. Phương pháp này mô phỏng cách tìm đường đi của kiến tự nhiên. Trong đó, lời giải chấp nhận được của bài toán được các con kiên nhân tạo xây dựng nhờ thủ tục bước ngẫu nhiên trên đồ thị cấu trúc. Việc tìm kiếm đỉnh mới của đường đi dựa trên sựkết hợp thong tin heuristic và thong tin học tăng cường biểu thị bởi vết mùi.

Khi áp dụng phương pháp này, có ba yếu tố quan trọng:

1) Xây dựng đồ thị cấu trúc

2) Xác định thông tin heuristic

3) Chọn quy tắc cập nhật mùi

Trong đó hai yếu tố đầu phụ thuộc vào từng bài toán cụ thể, còn yếu tố thứ ba có nhiều đề xuất và nghiên cứu cải tiến nhưng vẫn còn có thể nghiên cứu sâu hơn để có các cải tiến hiệu quả.

CHƯƠNG 3
CÀI ĐẶT THỬ NGHIỆM

Như đã trình bày trong chương 1, các thuật toán xấp xỉ tỏ ra khá hiệu quả trong việc tìm lời giải cho các bài toán thuộc loại NP-khó bao gồm cả giải thuật tối ưu đàn kiến. Chương này tập trung đánh giá chi tiết hiệu suất của các thuật toán đó trong từng bài toán cụ thể để thấy rõ được ưu, nhược điểm của thuật toán tối ưu đàn kiến so với các thuật toán khác đã được công bố.

3.1. Thuật toán r|p-ACO giải bài toán r|p trung tâm

Đã có nhiều thuật toán đúng, heuristics và metaheuristics được đề xuất cho bài toán này. Gần đây, Davydov cùng các cộng sự [9] đã đề xuất 2 thuật toán metaheuristics VNS (Variable Neighborhood Search) và STS (Stochastic Tabu Search) giải gần đúng nhanh bài toán *Trước*; Alekseeva cùng các cộng sự [4] đã đề xuất 2 thuật toán metaheuristics IEM (Iterative Exact Method) và MEM (Modified Exact Method) giải đúng bài toán cho *Trước*, sau đó phát triển thuật toán IM (Iteration Method) [5] hiệu quả hơn các phương pháp đã biết trước đó. Các thuật toán này đều giải bài toán quy hoạch 2 mức nhờ dùng phần mềm CPLEX để tìm lời giải tối ưu cho *Sau* mỗi khi biết các cơ sở của *Trước*. Chúng tôi đã đề xuất thuật toán r|p-ACO dựa trên thuật toán tối ưu đàn kiến giải để giải bài toán và so sánh với các công bố trên, kết quả này đã được đăng trong [2].

3.1.1. Lược đồ tổng quát

Trong thuật toán r|p-ACO, *Trước* và *Sau* thực hiện quá trình lặp tuần tự việc tìm lời giải gần đúng cho mỗi người chơi. Ký hiệu $n_{ant}T$ và $n_{ant}S$ tương ứng là số kiến được dùng để tìm lời giải gần đúng cho người chơi *Trước* và *Sau* trong mỗi vòng lặp, N_{total} là số vòng lặp tuần tự tìm lời giải của thuật toán. Khi đó với r và p đã cho thuật toán $r|p$-ACO thực hiện theo lược đồ như hình 3.1.

Bước 1. Khởi tạo ma trận vết mùi cho *Trước* và $n_{ant}T$, $n_{ant}S$, N_{total};
Bước 2. Thực hiện lặp:
  2.1. ACO- *Trước;* //Với mỗi kiến k tìm lời giải X_k cho *Trước*
  2.2. ACO- *Sau*; // Tìm lời giải Y_k cho *Sau* với lời giải của *Trước* là X_k;
  2.3. Chọn ra X* là lời giải tốt nhất trong số các lời giải X_k;
  2.4. LS(X*); // Tìm kiếm địa phương cho lời giải X* tốt nhất
  2.5. Cập nhật X* và trở lại 2.1 nếu chưa kết thúc.
Bước 3. Trích lời giải cho trước và sau.

Hình 3.1. Thuật toán $r|p$-ACO

Trong đó các thủ tục ACO- *Trước* và ACO- *Sau* được trình bày chi tiết trong mục sau.

3.1.2. Thủ tục ACO

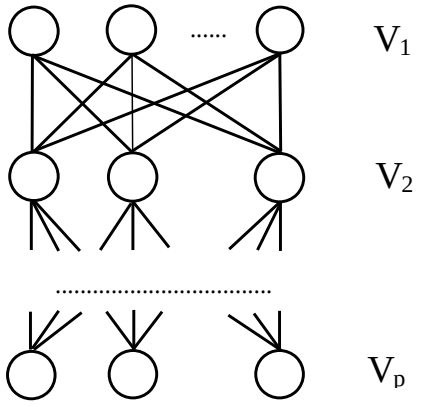
Các thủ tục ACO- *Trước* và ACO- *Sau* thực hiện như mô tả trong hình 3.1, chỉ khác nhau ở tập vị trí được chọn trong mỗi bước lặp. Trước khi mô tả cụ thể từng thủ tục, chúng ta cùng tìm hiểu 4 yếu tố quan trọng trong thuật toán tối ưu đàn kiến là: đồ thị cấu trúc và thủ tục bước ngẫu nhiên, thông tin heuristics, quy tắc cập nhật mùi, kỹ thuật tìm kiếm địa phương.

Đồ thị cấu trúc và thủ tục bước ngẫu nhiên.

Đồ thị cấu trúc cho ACO-*Trước* (hoặc ACO-*Sau*) là một đồ thị $G(V,E)$ tương ứng gồm p (hoặc r tầng), các đỉnh ở mỗi tầng có cấu trúc như nhau ($V_i = I$) là tập vị trí có thể đặt cơ sở, các đỉnh ở tầng trước có cạnh kết nối với các đỉnh ở tầng liền sau nó như được mô tả trong Hình 3.2. Khi xây dựng lời giải theo thủ tục bước ngẫu nhiên, kiến chọn ngẫu nhiên một đỉnh thuộc tập ứng cử *allow* ở tầng hiện tại dựa trên vết mùi và thông tin heuristics sau đó đỉnh này được loại khỏi tập ứng cử cho việc chọn đỉnh kế tiếp ở tầng sau. Ở tầng thứ nhất, tập *allow* của *Trước* bằng $\{I\}$, còn tập *allow* của $Sau = \{I - X\}$. Nếu kiến k ở đỉnh y nào đó của tầng i thì xác suất nó chọn đỉnh x trong tập $allow(y)$ ở tầng sau được cho bởi công thức (3.1).

$$p_x^k = \frac{\tau_x^\alpha n_x^\beta}{\sum_{z \in allow(y)} \tau_z^\alpha n_z^\beta} \qquad (3.1)$$

Lưu ý rằng khi một đỉnh (vị trí) đã được kiến đi qua thì đỉnh tương ứng bị loại khỏi tập ứng cử cho mỗi tầng sau khi xây dựng lời giải. Việc xây dựng lời giải của kiến kết thúc khi qua hết các tầng.



Hình 3.2. Đồ thị cấu trúc

Thông tin heuristic

Thông tin heuristics η_x của đỉnh x được tính bằng tổng lợi nhuận của l khách hàng gần đỉnh x nhất chia cho tổng độ dài từ l khách hàng này tới x như trong công thức (3.2)

$$n_x = \frac{\sum_{i=1}^{l} W_i}{\sum_{i=1}^{l} dists[x,i]}, \qquad (3.2)$$

trong đó $l = \frac{|I|}{p}$ là tỷ số giữa số lượng cơ sở trong I và p, τ_x là giá trị vết mùi trên đỉnh x ($x \in V_i$).

Quy tắc cập nhật vết mùi

Sau mỗi vòng lặp của thủ tục ACO- *Trước* / ACO- *Sau*, cường độ vết mùi trên mỗi đỉnh sẽ được cập nhật theo quy tắc SMMAS (xem [1]) cho bởi công thức (3.3)

$$\tau_i \leftarrow (1-\rho)\tau_i + \Delta\tau_i \qquad (3.3)$$

$$\text{với } \Delta\tau_i = \begin{cases} \rho\tau_{min} \text{ nếu } (i) \notin w(t) \\ \rho\tau_{max} \text{ nếu } (i) \in w(t) \end{cases},$$

w(t) là lời giải tốt nhất ở bước lặp thứ t sau khi đã thực hiện tìm kiếm địa phương, và τ_{max} và τ_{min} là 2 tham số được xác định trước.

ACO-Trước

Thủ tục ACO- *Trước* thực hiện quá trình tìm lời giải cho người chơi *Trước* được đặc tả trong hình 3.3, trong đó mỗi kiến sẽ lần lượt xây dựng lời giải cho riêng mình.

```
Procedure ACO- Trước
Begin
    for mỗi kiến k ∈ n_ant T do
      Xây dựng lời giải X_k cho kiến thứ k;
    Return X;
End;
```

Hình 3.3. Thủ tục ACO- Trước

ACO- Sau

Để đánh giá được lợi nhuận mà *Trước* nhận được khi chọn phương án *X* thì bài toán *Sau* phải được giải quyết. Như đã được trình bày ở trên, bài toán *Sau*

hay còn được gọi là bài toán $(r|Xp) - medianoid$ đã được chứng minh là NP-khó. Thủ tục ACO- *Sau* được xây dựng nhằm tìm ra r cơ sở tối ưu cho *Sau* khi biết được p cơ sở của phương án X do *Trước* chọn như trong hình 3.4. Với mỗi phương án X, chúng ta sử dụng số lượng kiến là $n_{ant}S$, mỗi kiến sẽ xây dựng phương án Y cho *Sau* cho riêng mình. Lời giải tốt nhất của phương án Y sẽ được sử dụng để tìm kiếm địa phương nhằm tăng chất lượng lời giải tại mỗi bước lặp. Kết thúc quá trình lặp sẽ trả về lời giải Y tốt nhất (Y*) cho người chơi *Sau*.

Cấu trúc đồ thị, thông tin heuristic và quy tắc cập nhật vết mùi trong thuật toán ACO- *Sau* tương tự như trong thuật toán ACO- *Trước*.

```
Procedure ACO- Sau(X)
Begin
    Khởi tạo ma trận vết mùi cho Sau và n_ant S kiến;
    repeat
      for mỗi kiến k ∈ n_ant S do
          Xây dựng lời giải Y_k cho kiến thứ k;
      Chọn ra Y* là lời giải tốt nhất trong các Y_k;
      LS(Y*); //Tìm kiếm địa phương cho phương án Y*
       Cập nhật Y*;
     until gặp điều kiện dừng
     return Y*;
End;
```

Hình 3.4. Thuật toán ACO-Sau

Kỹ thuật tìm kiếm địa phương

Kỹ thuật tìm kiếm địa phương thực hiện đối với phương án X như sau: Mỗi phần tử trong phương án X sẽ được thay thế bởi một phần tử trong tập ứng cử U, nếu hàm mục tiêu thu được là cao hơn thì ghi nhận lại phương án X. Quá trình này được lặp lại cho đến khi mọi phần tử trong U được thay thế vào mọi vị trí trong X.

```
Procedure LS(X)
Begin
    U = I – X;
    for mỗi x ∈ X do
        for mỗi u ∈ U do
        Thay thế x bằng u;
        If (lợi nhuận thu được là tốt hơn) then Cập nhật X;
    Return X;
End;
```

Hình 3.5. Thuật toán tìm kiếm địa phương

Thuật toán tìm kiếm địa phương giúp cho việc cải thiện kết quả được tốt hơn, tuy nhiên độ phức tạp của thuật toán lại khá lớn, vì thế chúng ta nên áp dụng tìm kiếm địa phương với kiến tốt nhất tại mỗi bước lặp nhằm mục đích tìm kiếm được lời giải tối ưu toàn cục.

3.1.3. Kết quả thử nghiệm

Thuật toán $r|p$-ACO được cài đặt thử nghiệm trên bộ dữ liệu từ thư viện Discrete Location Problems[1]. Tất cả các bộ thử nghiệm đều có kích thước như nhau $|I| = |J| = 100$. Có 2 bộ dữ liệu là *Eclidean* và *Uniform*. Trong loại *Eclidean*, ma trận (d_{ij}) xác định khoảng cách Eclide giữa các điểm trên một mặt phẳng, và tất cả các điểm đó đều thuộc phạm vi 7000 x 7000. Trong loại *Uniform*, mỗi phần tử của ma trận (d_{ij}) có giá trị ngẫu nhiên trong khoảng từ 0 đến 10^4. Trong mỗi bộ dữ liệu đều có hai loại lợi nhuận, trường hợp thứ nhất $w_j = 1$ với mọi $j \in J$, còn trong trường hợp thứ hai thì các giá trị này được lựa chọn ngẫu nhiên trong khoảng từ 0 đến 200. Kết quả thử nghiệm với các bộ $p = r = \{10, 15\}$ trên bộ dữ liệu *Eclidean* và $p = r = \{7\}$ trên bộ dữ liệu *Uniform* được trình bày lần lượt ở các bảng dưới đây.

Thử nghiệm về thuật toán r|p-ACO được tiến hành chạy 20 lần trên cùng một máy tính Intel Pentium G3220 3.0GHz, RAM 4GB, Window 7 Professional. Mục đích của thử nghiệm là đánh giá hiệu suất của thuật toán đề xuất thông qua so sánh lợi nhuận lớn nhất của *Trước* nhận được và độ phức tạp thời gian (phút) của thuật toán đề xuất với các giá trị tương ứng của thuật toán IM [5] (cài đặt trên máy tính Intel Xeon X5675, 3 GHz, RAM 96 GB, Windows

[1] http://math.nsc.ru/AP/benchmarks/Competitive/p_med_comp_eng.html

Server 2008 và phần mềm CPLEX 12.3) và VNS, STS [9] (cài đặt trên máy tính Pentium Intel Core Dual PC, 2.66 GHz, 2GB RAM).

Các tham số trong thử nghiệm được thiết đặt như sau:

$|I| = |J| = 100$, $n_{ant}T$ = 50, $n_{ant}S$ = 10, N_{total}=100 ;

$\tau_{max} = 1.0, \tau_{min} = \frac{\tau_{max}}{2*|I|}\alpha = 1, \beta = 2, \rho = 0.1.$

Các bảng 3.1, 3.2, 3.3 là các kết quả tính toán trung bình tương ứng của các thuật toán $r|p$-ACO, IM, VNS và STS . Trong đó, bảng 3.1, 3.2 là kết quả khi chạy trên bộ dữ liệu *Euclidean*, bảng 3.3 là kết quả khi chạy trên bộ dữ liệu *Uniform*. Trong mỗi bảng, cột Bộ test thể hiện mã của mỗi bộ test thử nghiệm trong bộ dữ liệu, cột W*(X) và Time tương ứng là lợi nhuận lớn nhất mà *Trước* thu được và thời gian chạy của mỗi thuật toán được tính theo đơn vị phút.

Bảng 3.1. Bộ dữ liệu Eclidean, $p = r = 10$

Bộ test	$W_j = 1$				$W_j \in 0\ldots200$							
	W*(X)		Time (phút)		W*(X)				Time (phút)			
	$r\|p$-ACO	IM	$r\|p$-ACO	IM	$r\|p$-ACO	IM	VNS	STS	$r\|p$-ACO	IM	VNS	STS
111	50	50	13.28	13	4,361	4,361	4,361	4,361	5.83	60	0.35	1.07
211	49	49	5.28	20	5,310	5,310	5,310	5,310	4.28	42	0.42	0.4
311	48	48	13.83	195	4,483	4,483	4,483	4,483	1.48	146	0.35	0.35
411	49	49	20.9	135	4,994	4,994	4,994	4,994	1.53	33	3.33	0.33
511	48	48	0.5	270	4,906	4,906	4,906	4,906	1.37	399	1.78	0.47
611	47	47	9.13	900	4,595	4,595	4,595	4,595	0.9	143	1.8	0.75
711	51	51	0.9	12	5,586	5,586	5,586	5,586	7.22	73	0.93	1.7
811	48	48	20.3	145	4,609	4,609	4,609	4,609	3.65	152	3.47	1.48
911	49	49	1.45	102	5,302	5,302	5,302	5,302	2.35	6	0.4	0.33
1011	49	49	5.92	180	5,005	5,005	5,005	5,005	3.28	97	3.57	1.73

Trong Bảng 3.1, với trường hợp W_j = 1 thì thuật toán $r|p$-ACO cho kết quả tương đồng với thuật toán IM nhưng với thời gian nhỏ hơn nhiều. Còn đối với trường hợp $W_j \in 0\ldots200$ thì thuật toán VNS, STS và $r|p$-ACO hơn nhau không đáng kể về mặt thời gian, bởi vì khi số lượng cơ sở được chọn cho *Trước* và *Sau* là thấp thì độ phức tạp của bài toán còn nhỏ, nhưng với số lượng cơ sở được chọn tăng dần thì độ phức tạp của bài toán tăng lên đáng kể. Trong Ekaterina Alekseeva [4] đã chứng minh độ phức tạp của bài toán là lớn nhất khi p = r = {15, 16, 17}.

Bảng 3.2. Bộ dữ liệu Eclidean $p = r = 15$

Bộ test	$W_j \in 0 \ldots 200$							
	W*(X)				Time (phút)			
	$r\|p$-ACO	IM	VNS	STS	$r\|p$-ACO	IM	VNS	STS
111	4,596	4,596	4,596	4,596	20.08	72	4.97	2.9
211	5,373	5,373	5,373	5,373	45.17	3,845	3.35	1.4
311	4,800	4,800	4,800	4,800	7.03	395.00	0.38	1.5
411	5,064	5,064	5,058	5,064	14.47	1,223	1.85	2.03
511	5,131	5,131	5,123	5,131	26.83	2,120	3.24	3.62
611	4,881	4,881	4,881	4,881	18.92	2,293	1.4	1.92
711	5,827	5,827	5,827	5,827	13.6	1,320	4.69	3.52
811	4,675	4,675	4,620	4,675	6.35	4,570	5.03	2.1
911	5,158	5,158	5,157	5,158	41.67	>600	4.23	2.63
1011	5,195	5,195	5,195	5,195	81.73	>600	0.4	0.82

Kết quả trong Bảng 3.2 cho thấy thuật toán $r|p$-ACO cho kết quả chính xác trong thời gian ngắn hơn nhiều so với thuật toán IM khi số lượng cơ sở được chọn cho *Trước* và *Sau* ngày càng tăng. Với $p = r = 15$, trong một số bộ test, thuật toán IM mặc dù với cấu hình mạnh nhưng phải chạy trong vài nghìn phút mới thu được kết quả (ví dụ bộ test 811 yêu cầu một khoảng thời gian lên đến 4,570 phút), nhưng thuật toán $r|p$-ACO chạy trong khoảng thời gian chấp nhận được (trung bình khoảng 27 phút). So với thuật toán VNS và STS thì thuật toán $r|p$-ACO chạy chậm hơn bởi lý do chính là $r|p$-ACO được dùng để giải cả hai bài toán cho *Trước* và *Sau*, trong khi đó VNS và STS chỉ giải bài toán của *Trước* (còn bài toán của Sau được giải bởi phần mềm CPLEX).

Bảng 3.3. Bộ dữ liệu Uniform $p = r = 7$

Bộ test	$W_j \in 0 \ldots 200$					
	W*(X)			Time (phút)		
	$r\|p$-ACO	VNS	STS	$r\|p$-ACO	VNS	STS
123	5,009	5,009	5,009	38.26	5.08	1.1
223	5,459	5,459	5,459	36.92	3.05	1.1
323	5,019	5,009	5,019	79.28	2.42	0.92
423	4,908	4,908	4,908	175.53	4.95	2.44
523	5,208	5,198	5,208	8.4	4.88	0.38

623	5,032	5,032	5,032	11.67	4.95	3.3
723	5,055	5,055	5,055	18.62	4.78	1.05
823	4,951	4,860	4,951	4.2	4.93	1.25
923	5,127	5,060	5,127	14.43	3.63	1.87
1023	5,084	5,067	5,084	59.1	5.38	4.65

Từ kết quả thử nghiệm trong Bảng 3.3 có thể kết luận rằng thuật toán $r|p$-ACO đề xuất cho kết quả tương đương với thuật toán STS trong khi đó thuật toán VNS chỉ đúng với các bộ test 123, 223, 423, 623, 723. Thời gian chạy của thuật toán không tối ưu bởi lý do tương tự như phân tích ở trên.

3.2. So sánh các thuật toán giải bài toán CSLP

Như đã được trình bày ở chương 1, bài toán CSLP là một bài toán thuộc lớp NP-khó và đã có nhiều thuật toán được đề xuất giải bài toán. Trong đó, thuật toán di truyền (GA) được H. Li và P. E. Love đề xuất giải bài toán ở TH2 và TH3. H. Zhang và J. Y. Wang [39] đã đề xuất thuật toán PSO (Particle Swarm Optimization) giải bài toán ở TH3. Thuật toán ACO được rất nhiều các tác giả đề xuất, tuy nhiên năm 2015 G. Calis và O. Yuksel [7] đưa ra một thuật toán kết hợp giữa ACO và phân tích tham số cho thấy sự mạnh mẽ của thuật toán ACO khi được phân tích và tối ưu tham số. Năm 2016, Quang cùng các cộng sự [30] đã đề xuất thuật toán lopt-aiNet giải quyết bài toán CSLP, đồng thời so sánh và đánh giá cả bốn giải thuật GA, PSO, ACO và lopt-aiNet với từng trường hợp cụ thể như trong các bảng 3.4, 3.5.

Bảng 3.4. So sánh kết quả của các TH1, TH2 và TH3

	TH1	**TH2**	**TH3**
GA [20]		15,090	
GA [21]			15,160
PSO [39]			16,060
ACO [14]	12,546		
ACO [6]			12,628
ACO-PA [7]	12,150	12,578	12,606
opt-aiNet [30]	12,436	12,582	12,616
lopt-aiNet [30]	12,150	12,546	12,606

Nhìn bảng 3.4. ta có thể thấy trong TH1 và TH3 thì kết quả của lopt-aiNet và ACO-PA là tương đương nhau (bằng 12150 ở TH1 và 12606 ở TH2). Tuy nhiên, trong TH2 thuật toán ACO-PA tỏ ra kém hiệu quả hơn so với thuật toán

lopt-aiNet cụ thể là thuật toán lopt-aiNet cho kết quả tốt hơn 0.25% so với thuật toán ACO-PA. Về mặt thời gian, để tìm ra lời giải cho mỗi lần chạy thì thuật toán ACO-PA mất 1.15 giây với máy tính Intel Core 2 Duo 2.66GHz và 4GB RAM. Trong khi đó, lopt-aiNet chỉ mất 0.15 giây trên máy tính cấu có cấu hình thấp hơn là CPU Pentium P6200 2.13GHz, RAM 2GB.

Bảng 3.5. So sánh kết quả trong TH4 và TH5

TH4										
Run	**GA [3]**		**PSO [3]**		**ACO [3]**		**opt-aiNet [30]**		**lopt-aiNet [30]**	
	result	**time**	**result**	**time**	**result**	**time**	**result**	**Time**	**result**	**time**
1	91	0.53	90	1.93	90	0.37	100	0.19	90	0.22
2	90	0.52	91	1.97	90	0.34	101	0.18	90	0.18
3	93	0.56	90	1.97	93	0.32	100	0.18	90	0.20
4	90	0.58	90	1.93	91	0.33	102	0.21	90	0.21
5	91	0.52	92	1.96	90	0.32	103	0.20	90	0.20
Ave	91.0	0.54	90.6	1.96	90.8	0.33	101.2	0.19	90.0	0.20
TH5										
1	90	0.52	93	1.82	90	0.37	103	0.19	90	0.21
2	92	0.55	90	1.87	91	0.35	104	0.20	90	0.20
3	90	0.54	90	1.89	93	0.35	105	0.18	90	0.22
4	96	0.54	91	1.88	91	0.33	100	0.19	90	0.24
5	90	0.52	90	1.89	90	0.35	104	0.21	90	0.20
Ave	91.6	0.54	90.8	1.87	91.0	0.35	103.2	0.19	90.0	0.21

Bảng 3.5. thể hiện hiệu suất của các thuật toán GA, PSO, ACO, opt-aiNet và lopt-aiNet trong 5 lần chạy. Thuật toán tốt nhất cho kết quả tối ưu (= 90) là thuật toán lopt-aiNet do Quang cùng các cộng sự [30] đề xuất, sau đó đến thuật toán ACO với kết quả trung bình là 90.8 trong TH1 và 91.0 với TH2. Về mặt thời gian, thuật toán opt-aiNet và thuật toán lopt-aiNet chạy trong thời gian ngắn nhất chỉ mất 0.19 – 0.2 giây trong TH4 và 0.19 – 0.21 giây trong TH5, thuật toán ACO mất 0.33 giây trong TH4 và 0.35 giây trong TH5, thuật toán GA sử dụng 0.54 giây trong cả hai loại TH4 và TH5, thuật toán PSO được đánh giá là thuật toán chậm nhất với 1.96 giây trong TH4 và 1.87 giây trong TH5.

3.3. Áp dụng thuật toán ACO-SRFL giải bài toán SRFL

3.3.1. Mô tả thuật toán

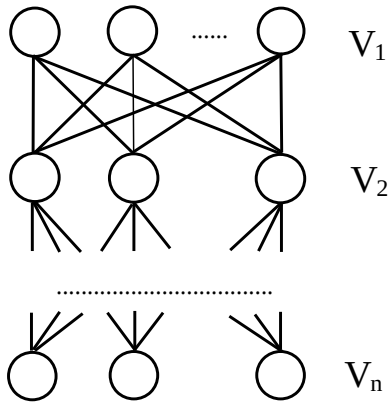
Thuật toán ACO-SRFL được xây dựng dựa trên thuật toán ACO có lược đồ tổng quan như hình 3.6.

```
Procedure ACO-SRFL;
Begin
    khởi tạo ma trận mùi, khởi tạo m con kiến;
    Repeat
        Xây dựng lời giải cho mỗi con kiến;
        Chọn ra kiến k cho lời giải tốt nhất;
        Áp dụng tìm kiếm địa phương cho kiến k;
        Cập nhật vết mùi;
    Until gặp điều kiện kết thúc
End;
```

Hình 3.6. Thuật toán ACO-SRFL

3.3.2. Đồ thị cấu trúc và thủ tục xây dựng lời giải

Tương tự như đồ thị cấp trúc của thuật toán r|p-ACO, đồ thị cấu trúc của thuật toán ACO-SRFL được chia thành n tầng, với mỗi tầng i là tập V_i gồm $n - i + 1$ đỉnh từ được đánh số từ 1 ... n như hình 3.7.



Hình 3.7. Đồ thị cấu trúc thuật toán ACO-SRFL

Mỗi kiến sẽ xây dựng lời giải bằng cách di chuyển từ tầng 1 xuống tầng n trong đồ thị cấu trúc, tại mỗi tầng kiến sẽ chọn một đỉnh bất kỳ dựa trên xác suất được tính dựa theo giá trị vết mùi và thông tin heuristic như trong công thức 3.4. Đỉnh được chọn sẽ được loại bỏ ra khỏi tầng tiếp theo của đồ thị. Quá trình này được lặp đi lặp lại cho đến khi đến tầng n của đồ thị, lúc này tại tầng n thì chỉ

còn duy nhất một đỉnh và đỉnh đó được kết nạp nốt vào lời giải. Thứ tự chọn các đỉnh từ tầng 1 đến tầng n trong đồ thị cấu trúc của mỗi kiến sẽ chính là lời giải của kiến đó.

$$p_x^k = \frac{\tau_x^\alpha n_x^\beta}{\sum_{z \in allow(y)} \tau_z^\alpha n_z^\beta} \quad (3.4)$$

3.3.3 Quy tắc cập nhật vết mùi

Thuật toán ACO-SRFL sử dụng quy tắc cập nhật vết mùi là SMMAS tương tự như trong thuật toán r|p-ACO, và quy tắc này chỉ áp dụng cho kiến có lời giải tốt nhất tại mỗi bước lặp.

3.3.4. Tìm kiếm địa phương

Để tăng hiệu quả thuật toán, trong mỗi lần lặp chúng tôi dụng thuật toán tìm kiếm địa phương (local search) cho lời giải tốt nhất tìm được tại mỗi bước lặp theo chiến lược 2-opt.

3.3.5. Kết quả thử nghiệm

Trong phần này, thuật toán ACO-SRFL được cài đặt và thử nghiệm trên một số bộ dữ liệu của bài toán SRFL là: LW5, S8H, S10, LW11, H20, H30. Bộ dữ liệu LW5 và LW11 được đưa ra bởi Love và Wong [22]. Bộ dữ liệu S8H, S10 được Simmons [32] đề xuất. Hai bộ dữ liệu H20 và H30 lần lượt được Nugent [26] và Heragu [17] công bố. Bảng 3.6 cho thấy số lượng cơ sở và lời giải tối ưu của mỗi bộ dữ liệu.

Bảng 3.6. Lời giải tối ưu của 6 bộ dữ liệu

Bộ dữ liệu	**Số lượng cơ sở**	**Lời giải tối ưu**
LW5	5	151.0
S8H	8	2,324.5
S10	10	2,781.5
LW11	11	6,933.5
H20	20	15,549.0
H30	30	44,965.0

Mỗi bộ dữ liệu đều được tiến hành chạy 20 lần thuật toán ACO-SRFL trên cùng một máy tính Intel Pentium P6200 2.13GHz, 2GB RAM. Kết quả và thời gian chạy trung bình của thuật toán được so sánh với kết quả trong [34], [33], [17], [18].

Các tham số trong bài được thiết lập như sau: $\alpha = 1$, $\beta = 2$, $\rho = 0.2$, $numAnt = 10$, điều kiện dừng của thuật toán là sau khi chạy hết 100 bước lặp.

Bảng 3.7. So sánh kết quả thuật toán ACO- SRFL với các thuật toán khác.

Bộ dữ liệu	**[17]**	**[18]**	**[34]**	**[33]**	**ACO-SRFL**
LW5	151	151	151	151	151
S8H	2324.5	2324.5	2324.5	2329.82	2324.5
S10	2781.5	2781.5	2781.5	2836	2781.5
LW11	6933.5	7265.5	6933.5	7139.2	6933.5
H20	15602	15549.0	15549.0	16052.2	15549.0
H30	45111	-	-	50143.2	45019.0

Nhìn bảng 3.7 ta có thể thấy, thuật toán ACO-SRFL cho kết quả chính xác ở các bộ dữ liệu LW5, S8H, S10, LW11, H20 và cho kết quả tốt hơn [17] [33]. Khi so sánh tốc độ thực hiện của thuật toán ACO-SRFL với thời gian chạy của các thuật toán khác thì chúng ta có thể thấy rằng thuật toán ACO-SRFL chạy nhanh hơn tất cả các thuật toán khác. Tuy nhiên, trong các thuật toán đã công bố chỉ có thuật toán đàn dơi [33] được thực hiện trên máy tính tốt hơn của tác giả. Do vậy, để đảm bảo tính khách quan, bảng 4.8 chỉ so sánh thời gian chạy thuật toán ACO-SRFL với thuật toán đàn dơi [33].

Bảng 3.8. So sánh thời gian chạy giữa thuật toán ACO- SRFL với thuật toán đàn dơi (Bat Algorithm)

Bộ dữ liệu	**[33]**	**ACO- SRFL**
LW5	9.21	0.0
S8H	11.29	0.0
S10	12.58	0.0
LW11	14.06	0.1
H20	27.68	0.9
H30	57.43	17

Trong bảng 3.8, thời gian chạy thuật toán ACO-SRFL nhỏ hơn nhiều so với thuật toán Bat đã được công bố gần đây nhất của Sinem [33] sử dụng máy tính có cấu hình cao hơn Intel Core 2 Duo 2.4 GHz và 4 GB RAM.

3.4. Kết luận chương

Chương này trình bày các kết quả thử nghiệm của thuật toán ACO khi áp dụng vào các bài toán cụ thể trong lớp các bài toán vị trí cơ sở. Trong đó, có thể thấy rằng thuật toán r|p-ACO giải quyết bài toán r|p-trung tâm chỉ xếp sau thuật toán STS nhưng tốt hơn thuật toán VNS về mặt kết quả và IM về mặt thời gian.

Đối với bài toán CSLP, thuật toán ACO-PA chỉ xếp sau thuật toán lopt-aiNet còn tốt hơn các thuật toán khác như GA, PSO. Còn với bài toán SRFL thì thuật toán ACO-SRFL vượt trội hơn hẳn thuật toán đàn dơi cả về phương diện kết quả lẫn thời gian.

KẾT LUẬN

Kết luận

Phương pháp tối ưu đàn kiến là phương pháp tương đối mới mẻ và tỏ ra đặc biệt hiệu quả, điều này đã được chứng minh thông qua thực nghiệm. Phương pháp tối ưu đàn kiến luôn được quan tâm, phát triển kể từ khi giới thiệu cho đến nay thể hiện qua sự phong phú, đa dạng của các thuật toán. Các thuật toán trực tiếp đưa ra hướng tiếp cận mới giải các bài toán tối ưu tổ hợp, qua đó có nhiều ứng dụng trong thực tiễn trên các lĩnh vực như: sản xuất, truyền thông, sinh học, hoạt động xã hội…

Bài toán vị trí cơ sở là một bài toán lớn bao hàm nhiều bài toán con có ứng dụng thực tế cao, nó giúp chúng ta lựa chọn các vị trí cơ sở để đặt các trạm dịch vụ một cách tối ưu nhất.

Đối với bài toán r|p-trung tâm, bài toán CSLP và bài toán SRFL, chúng tôi đã đề xuất thuật toán dựa trên thuật toán ACO, đồng thời có so sánh đánh giá thuật toán với một số thuật toán khác để thấy được ưu, nhược điểm của thuật toán.

Hướng phát triển

Cải thiện tốc độ thực hiện của thuật toán thông qua cải tiến tìm kiếm địa phương và/hoặc kết hợp với phần mềm CPLEX.

Tiếp cận với các bài toán tương tự về mạng, khi khách hàng nằm ở các đỉnh của đồ thị còn các cơ sở có thể mở tại các điểm tùy ý trên các cạnh của nó.

Với bài toán r|p-trung tâm, nghiên cứu phương pháp giải bài toán với giá trị $p \neq r$ với các bài toán CSLP và SRFL thử nghiệm với các bộ dữ liệu có kích thước lớn hơn.

DANH MỤC CÁC CÔNG TRÌNH ĐÃ CÔNG BỐ CỦA TÁC GIẢ

[1]. Vũ Đức Quang, Hoàng Xuân Huấn, Đỗ Thanh Mai, (2016), *"*Một thuật toán hiệu quả dựa trên giải thuật tối ưu đàn kiến giải bài toán r|p trung tâm*"*, trong *Kỷ yếu Hội nghị Quốc gia lần thứ 9 về Nghiên cứu cơ bản và ứng dụng Công Nghệ thông tin (FAIR) tại Đại học Cần Thơ*. tr 488 – 494.

[2]. Duc Quang Vu, Van Truong Nguyen, Xuan Huan Hoang, (2016), "An Improved Artificial Immune Network For Solving Construction Site Layout Optimization*"*, in *Proceeding of the 12th IEEE-RIVF International Conference on Computing and Communication Technologies*, pp 37 – 42.

TÀI LIỆU THAM KHẢO

Tiếng Việt

1. Đ. Đ. Đông (2012), *Phương pháp tối ưu đàn kiến và ứng dụng*, Luận án Tiến sĩ, Đại học công nghệ, Đại học Quốc Gia Hà Nội.
2. V. Đ. Quang, H. X. Huấn và Đ. T. Mai (2016), Một thuật toán hiệu quả dựa trên giải thuật tối ưu đàn kiến giải bài toán r|p trung tâm, *Fundamental and Applied IT Research*, Đại học Cần Thơ, tr. 488-496.

Tiếng Anh

3. A. M. Adrian, A. Utamima và K. J. Wang (2014), "A comparative study of GA, PSO and ACO for solving Construction Site Layout Optimization", *KSCE Journal of Civil Engineering*. 1, tr. 520-527.
4. E. Alekseeva và Y. Kochetov (2013), "Metaheuristics and Exact Methods for the Discrete (r|p)-Centroid Problem", trong El-Ghazali Talbi, chủ biên, *Metaheuristics for Bi-level Optimization, Berlin, Springer Berlin Heidelberg*, Springer Berlin Heidelberg, Berlin, tr. 189-219.
5. E. Alekseeva, Y. Kochetov và A. Plyasunov (2015), "An exact method for the discrete (r|p)-centroid problem", *Springer Science+Business Media New York*. 63, tr. 445–460.
6. G. Calis và O. Yuksel (2010), A comparative study for layout planning of temporary construction facilities: optimization by using ant colony algorithms, *Proceedings of the International Conference on Computing in Civil and Building Engineering*.
7. G. Calis và O. Yuksel (2015), "An Improved Ant Colony Optimization Algorithm for Construction Site Layout Problem", *Building Construction and Planning Research*. 3, tr. 221-232.
8. I. A. Davydov (2012), "Local Tabu Search for the Discrete (r|p)-Centroid Problem", *Diskret. Anal. Issled. Oper*. 19(2), tr. 19-40.
9. I. A. Davydov và các cộng sự (2014), "Fast Metaheuristics for the Discrete (r|p)-Centroid Problem", *Automation and Remote Control*. 75(4), tr. 677–687.
10. I. A. Davydov, Y. Kochetov và E. Carrizosa (2013), "A Local Search Heuristic for the (r|p)-Centroid Problem", *Computers & Operations Research*. 52, tr. 334–340.
11. M. Dorigo và L. Gambardella (1997), "Ant colony system: A cooperative learning approach to the traveling salesman problem", *IEEE Trans. on evolutionary computation*. 1(1), tr. 53-66.
12. M. Dorigo, V. Maniezzo và A. Colorni (1996), "Ant system: optimization by a colony of cooperating agents", *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*. 26, tr. 29-41.

13. D. Erlenkotter (1978), "A dual-based procedure for uncapacitated facility location", *Operations Research*. 26(6), tr. 992-1009.
14. E. Gharaie, A. Afshar và M. R. Jalali (2006), Site Layout Optimization with ACO Algorithm, *Proceedings of the 5th WSEAS International Conference on Artificial Intelligence*.
15. W. Gutjahr (2002), "ACO algorithms with guaranteed convergence to the optimal solution", *Info.Proc. Lett.* 83(3), tr. 145-153.
16. S. L. Hakimi (1990), "Locations with Spatial Interactions: Competitive Locations and Games", *Discrete Location Theory, , London, Mirchandani P.B. and Francis R.L., Eds., London: Wiley*, tr. 439–478.
17. S. S. Heragu (1992), "Invited review. Recent models and techniques for solving the layout problem," *European Journal of Operational Research*. 57, tr. 136–144.
18. K. R. Kumar, G. C. Hadjinicola và T. L. Lin (1995), "A heuristic procedure for the single row facility layout problem", *European Journal of Operational Research*. 87, tr. 65–73.
19. K. C. Lam, X. Ning và M. C.-K. Lam (2009), "Conjoining MMAS to GA to Solve Construction Site Layout Planning Problem", *Construction Engineering and ManageConstruction Engineering and Managent*. 35, tr. 1049-1057.
20. H. Li và P. E. Love (1998), "Comparing Genetic Algorithms and Non-Linear Optimisation for Labor and Equipment Assignment", *Computing in Civil Engineering*. 12, tr. 227-231.
21. H. Li và P. E. Love (2000), "Genetic search for solving construction site level unequal area facility layout problems," *Automation in Construction*. 9, tr. 217-226.
22. R. F. Love và J. Y. Wong (1976), "On solving a one-dimensional space allocation problem with integer programming", *INFOR*. 14(2), tr. 139-143.
23. M. J. Mawdesley, S. H. Al-jibouri và H. Yang (2002), "Genetic algorithms for construction site layout in project planning", *Construction Engineering And Management*. 128, tr. 418-426.
24. X. Ning và W. H. Liu (2011), "Max-Min Ant System Approach for Solving Construction Site Layout", *Advanced Materials Research*. 328, tr. 128-131.
25. H. Noltemeier, J. Spoerhase và H. Wirth (2007), "Multiple Voting Location and Single Voting Location on Trees", *European Journal of Operational Research*. 181, tr. 654–667.
26. C. E. Nugent, T. E. Vollman và J. Ruml (1968), "An experimental comparison of techniques for the assignment of facilities to locations", *Oper. Res.* 16(1), tr. 150-173.
27. F. Ozcelik (2012), "A hybrid genetic algorithm for the single row layout problem", *International Journal of Production Research*. 50(20), tr. 5872-5886.

28. P. Pellegrini và A. Ellero (2008), The Small World of Pheromone Trails, *Proc. of the 6th international conference on Ant Colony Optimization and Swarm Intelligence, Brussels, Belgium*.
29. J. Poerhase và H. Wirth (2009), "(r, p)-Centroid Problems on Paths and Trees", *Journal of Theoretical and Computational Science,*. 410, tr. 5128–5137.
30. V. D. Quang, N. V. Truong và H. X. Huan (2016), An Improved Artificial Immune Network For Solving Construction Site Layout Optimization, *the 12th IEEE-RIVF International Conference on Computing and Communication Technologies*, Thuyloi University, HaNoi, VietNam, tr. 37-42.
31. H. Samarghandi, P. Taabayan và F. F. Jahantigh (2010), "A particle swarm optimization for the single row facility layout problem", *Computers & Industrial Engineering*. 58(4), tr. 529-534.
32. D. M. Simmons (1969), "One-dimensional space allocation: an ordering algorithm", *Oper Research*. 17(5), tr. 812-826.
33. B. Sinem (2015), "Bat Algorithm Application for the Single Row Facility Layout Problem", *Springer International Publishing*, tr. 101-120.
34. M. Solimanpur, P. Vrat và R. Shankar (2005), "An ant algorithm for the single row layout problem in flexible manufacturing systems", *Computers & Operations Research*. 32(3), tr. 583 –598.
35. T. Stützle và M. Dorigo (2002), "A short convergence proof for a class of ACO algorithms", *IEEE-EC*. 6(4), tr. 358-365.
36. T. Stützle và H. H. Hoos (2000), "Max-min ant system", *Future Gene. Comput. Syst.* 26(8), tr. 889-914.
37. E. G. Talbi (2013), *Metaheuristics for Bi-level Optimization*, Studies in Computational Intelligence, Berlin, Springer Publishing Company, Incorporated, 189–219.
38. I. C. Yeh (1995), "Construction-Site Layout Using Annealed Neural Network", *Computing in Civil Engineering*. 9, tr. 201-208.
39. H. Zhang và J. Y. Wang (2008), "Particle Swarm Optimization for Construction Site Unequal-Area Layout", *Construction Engineering and Management* 9, tr. 739-748.