ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

ĐỖ ĐỨC ĐÔNG

PHƯƠNG PHÁP TỐI ƯU ĐÀN KIẾN VÀ ỨNG DỤNG

LUẬN ÁN TIẾN SĨ CÔNG NGHỆ THÔNG TIN

Hà nội – 2012

ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

ĐỖ ĐỨC ĐÔNG

PHƯƠNG PHÁP TỐI ƯU ĐÀN KIẾN VÀ ỨNG DỤNG

Chuyên ngành: Khoa học máy tính
Mã số: 62.48.01.01

LUẬN ÁN TIẾN SĨ CÔNG NGHỆ THÔNG TIN

NGƯỜI HƯỚNG DẪN KHOA HỌC:
PGS.TS. Hoàng Xuân Huấn

Hà nội – 2012

Lời cam đoan

Tôi xin cam đoan đây là công trình nghiên cứu của riêng tôi. Các kết quả được viết chung với các tác giả khác đều được sự đồng ý của đồng tác giả trước khi đưa vào luận án. Các kết quả nêu trong luận án là trung thực và chưa từng được ai công bố trong các công trình nào khác.

Tác giả

Lời cảm ơn

Luận án được thực hiện tại trường ĐH Công nghệ - ĐHQG Hà nội, dưới sự hướng dẫn của PGS.TS Hoàng Xuân Huấn.

Tôi xin bày tỏ lòng biết ơn sâu sắc tới thầy Hoàng Xuân Huấn, người đã có những định hướng giúp tôi thành công trong việc nghiên cứu của mình. Thầy cũng đã động viên và chỉ bảo giúp tôi vượt qua những khó khăn để tôi hoàn thành được luận án này. Tôi cũng chân thành cảm ơn tới thầy Nguyễn Thanh Thuỷ, thầy Lê Sỹ Vinh, thầy Lê Anh Cường và thầy Nguyễn Phương Thái. Các thầy đã cho tôi nhiều kiến thức quý báu về nghiên cứu khoa học. Nhờ sự chỉ bảo của các thầy tôi mới hoàn thành tốt luận án.

Tôi cũng xin cảm ơn tới các Thầy, Cô thuộc khoa Công nghệ thông tin – ĐH Công nghệ, đã tạo mọi điều kiện thuận lợi giúp tôi trong quá trình làm nghiên cứu sinh.

Cuối cùng, tôi xin gửi lời cảm ơn sâu sắc tới gia đình, bạn bè nơi đã cho tôi điểm tựa vững chắc để tôi có được thành công như ngày hôm nay.

MỤC LỤC

Danh mục các ký hiệu và chữ viết tắt

τ_{max}	*Cận trên của vết mùi*
τ_{min}	*Cận dưới của vết mùi*
τ_{mid}	*Cận giữa của vết mùi*
τ_0	*Vết mùi được khởi tạo ban đầu*
τ_{ij}	*Vết mùi trên cạnh* (i, j)
η_{ij}	*Thông tin heuristic trên cạnh* (i, j)
N_c	*Số vòng lặp trong thuật toán ACO*
N_a	*Số kiến sử dụng trong thuật toán ACO*
ρ	*Tham số bay hơi*
3-LAS	*Three-Level Ant System (Hệ kiến ba mức)*
ACO	*Ant Colony Optimization (Tối ưu đàn kiến)*
ACOHAP	*Thuật toán tối ưu đàn kiến giải bài toán suy diễn haplotype*
AcoSeeD	*Thuật toán tối ưu đàn kiến tìm tập hạt giống tối ưu*
ACOSVM	*Thuật toán tối ưu đàn kiến tìm tham số trong SVM được dùng để dự báo hoạt động điều tiết gen*
ACS	*Ant Colony System (Hệ đàn kiến)*

AS	*Ant System (Hệ kiến)*
CRM	*Cis-Regulatory Module (Mô-đun điều tiết)*
EC	*Evolutionary Computing (Tính toán tiến hoá)*
GA	*Genetic Algorithm (Thuật toán di truyền)*
GASVM	*Thuật toán di truyền tìm tham số trong SVM được dùng để dự báo hoạt động điều tiết gen*
G-best	*Global-best (Lời giải tốt nhất tính đến thời điểm hiện tại)*
HI	*Haplotype Inference (Suy diễn haplotype)*
HIPP	*Haplotype Inference by Pure Parsimony (Suy diễn haplotype theo tiêu chuẩn Pure Parsimony)*
I-best	*Iteration-best (Lời giải tốt nhất trong bước lặp hiện tại)*
JSS	*Job Shop Scheduling (Bài toán lập lịch sản xuất)*
MLAS	*Multi-level Ant System (Hệ kiến đa mức)*
MMAS	*Max-Min Ant System (Hệ kiến Max Min)*
PSO	*Particle Swarm Optimization (Tối ưu bầy đàn)*
SA	*Simulated Annealing (Thuật toán mô phỏng luyện kim)*
SMMAS	*Smoothed Max-Min Ant System (Hệ kiến Max Min trơn)*
SpEED	*Thuật toán leo đồi tìm tập hạt giống tối ưu*

SVM	*Support Vector Machine (Phương pháp học máy SVM)*
TSP	*Traveling Salesman Problem (Bài toán người chào hàng)*
TƯTH	*Tối ưu tổ hợp*
UBQP	*Unconstrained Binary Quadratic Programming (Bài toán quy hoạch toàn phương nhị phân không ràng buộc)*

Danh mục các bảng

Danh mục các hình vẽ, đồ thị

MỞ ĐẦU

Trong thực tế và khi xây dựng các hệ thông tin, ta thường gặp các bài toán tối ưu tổ hợp (TƯTH), trong đó phải tìm các giá trị cho các biến rời rạc để làm cực trị hàm mục tiêu nào đó (xem [31,60]). Đa số các bài toán này thuộc lớp NP-khó. Trừ các bài toán cỡ nhỏ có thể tìm lời giải bằng cách tìm kiếm vét cạn, còn lại thì thường không thể tìm được lời giải tối ưu.

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

1) Tìm kiếm heuristic, trong đó dựa trên phân tích toán học, người ta đưa ra các quy tắc định hướng tìm kiếm một lời giải đủ tốt.

2) Sử dụng các kỹ thuật tìm kiếm cục bộ để tìm lời giải tối ưu địa phương.

3) Tìm lời giải gần đúng nhờ các thuật toán mô phỏng tự nhiên (xem [31,57,60]) như mô phỏng luyện kim (*Simulated Annealing - SA*), giải thuật di truyền (*Genetic Algorithm - GA*), tối ưu bầy đàn (*Particle Swarm Optimization - PSO*)…

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

Trong các phương pháp mô phỏng tự nhiên, tối ưu đàn kiến (*Ant Colony Optimization - ACO*) là cách tiếp cận metaheuristic tương đối mới, được giới thiệu bởi Dorigo năm 1991 (xem [28,29,31]) đang được nghiên cứu và ứng dụng rộng rãi cho các bài toán TƯTH khó (xem [7,9,10,31,36,37,55,59,63]).

Các thuật toán ACO mô phỏng cách tìm đường đi của các con kiến thực. Trên đường đi, mỗi con kiến thực để lại một vết hoá chất gọi là vết mùi (pheromone trail) và

theo vết mùi của các con kiến khác để tìm đường đi. Đường có nồng độ vết mùi càng cao thì càng có nhiều khả năng được các con kiến chọn. Nhờ cách giao tiếp gián tiếp này [31], đàn kiến tìm được đường đi ngắn nhất từ tổ tới nguồn thức ăn. Theo ý tưởng đó, các thuật toán ACO sử dụng kết hợp thông tin kinh nghiệm (hay còn gọi là thông tin heuristic) và học tăng cường qua các vết mùi của các con kiến nhân tạo để giải các bài toán TƯTH bằng cách đưa về bài toán tìm đường đi tối ưu trên đồ thị cấu trúc tương ứng của bài toán.

Với mỗi bài toán TƯTH (S, f, Ω), trong đó S là tập hữu hạn các phương án chấp nhận được, f là hàm mục tiêu xác định trên S và Ω là để xác định S qua các thành phần của tập hữu hạn C và các liên kết của tập này, mỗi lời giải s trong S sẽ tương ứng với một hoặc một tập hữu hạn các vectơ $x =< u_0, \ldots, u_k >$ $(u_i \in C \; \forall i \leq k \leq h)$ có độ dài bị chặn thỏa mãn các ràng buộc Ω. Như vậy, việc tìm kiếm các lời giải trong S được đưa về xây dựng lời giải trên các vectơ độ dài không quá h thỏa mãn ràng buộc đã cho. Về lý thuyết, quá trình giải có thể thực hiện trên đồ thị đầy đủ có tập đỉnh được gán nhãn bởi tập C với một số thông tin thêm (được gọi là đồ thị cấu trúc). Tùy theo các ràng buộc mà trong nhiều trường hợp ta có thể xét bài toán tìm trên đồ thị cấu trúc đơn giản hơn để thu hẹp miền tìm kiếm.

Trong mỗi lần lặp của các thuật toán, một đàn kiến nhân tạo sẽ xây dựng lời giải theo thủ tục phát triển tuần tự trên đồ thị cấu trúc, sau đó so sánh lời giải tìm được để cập nhật vết mùi như là thông tin học tăng cường dùng cho các vòng lặp sau. Nhờ đó mà lời giải được cải tiến dần. Các thuật toán này được áp dụng rộng rãi để giải nhiều bài toán khó và hiệu quả nổi trội của chúng so với các phương pháp mô phỏng tự nhiên khác đã được chứng tỏ bằng thực nghiệm.

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;

2) Chọn thông tin heuristic;

3) Chọn quy tắc cập nhật mùi.

Hai yếu tố đầu phụ thuộc vào đặc điểm của từng bài toán cụ thể, còn quy tắc cập nhật mùi là yếu tố phổ dụng cho các bài toán và nó thường dùng làm tên để phân biệt các thuật toán ACO.

Thuật toán ACO đầu tiên [28] là thuật toán hệ kiến (*Ant System - AS*) giải bài toán người chào hàng (*Traveling Salesman Problem - TSP*). Đến nay đã có nhiều biến thể được đề xuất, thông dụng nhất là hệ đàn kiến (*Ant Colony System - ACS*) [30,31] và hệ kiến Max-Min (*Max-Min Ant System - MMAS*) [66]. Ngoài các kết quả được kiểm chứng bằng thực nghiệm trên các bài toán ứng dụng, đã có nhiều nghiên cứu về đặc tính của các thuật toán [8,10,36-38,55,65] như ảnh hưởng của vết mùi, thời gian chạy và tính hội tụ, tính bất biến đối với biến đổi của hàm mục tiêu,… nhằm hiểu rõ và cải tiến các thuật toán.

Khi áp dụng các thuật toán thông dụng như ACS và MMAS, người ta phải tìm một lời giải đủ tốt, trên cơ sở đó xác định các tham số cho cận trên và cận dưới của vết mùi. Điều này gây nhiều khó khăn khi áp dụng thuật toán cho các bài toán mới. Ngoài ra, vấn đề lượng mùi cập nhật cho mỗi thành phần trong đồ thị tỷ lệ với giá trị hàm mục tiêu của lời giải chứa nó liệu có phản ánh đúng thông tin học tăng cường hay không cũng còn phải thảo luận.

Nhiệm vụ tác giả luận án đặt ra là:

1) Phân tích xu thế biến thiên của vết mùi trong các thuật toán ACO, trên cơ sở đó đề xuất các quy tắc cập nhật mùi dễ sử dụng và hiệu quả hơn;

2) Áp dụng kết quả đạt được, đề xuất các thuật toán giải một số bài toán thời sự trong công nghệ sinh học.

Trong luận án này, dựa trên các phân tích toán học, chúng tôi đề xuất quy tắc Max-Min trơn (*Smoothed Max-Min Ant System - SMMAS*) như là cải tiến của thuật toán Max-Min. Ưu điểm nổi trội của chúng được kiểm định bằng thực nghiệm qua đối với các bài toán chuẩn như: lập lịch sản xuất (*Job Shop Scheduling - JSS*), người chào hàng (*Traveling Salesman Problem - TSP*), quy hoạch toàn phương nhị phân không ràng buộc (*Unconstrained Binary Quadratic Programming- UBQP*). Trường hợp các thông tin heuristic có ảnh hưởng nhiều tới kết quả tìm kiếm, chúng tôi đề xuất quy tắc 3 mức (*Three-Level Ant System - 3-LAS*) và kiểm định hiệu quả của nó trong bài toán người chào hàng. Ngoài ra, để điều tiết linh hoạt tính khám phá và khai thác của phương pháp ACO, chúng tôi cũng gợi ý phát triển thuật toán đa mức (*Multi-level Ant System - MLAS*). Trong các quy tắc này, SMMAS đơn giản, dễ sử dụng hơn.

Hiện nay, ứng dụng công nghệ thông tin để nghiên cứu sinh học phân tử đang được quan tâm. Nhờ quy tắc cập nhật mùi SMMAS, luận án đề xuất các thuật toán mới giải các bài toán thời sự trong sinh học: bài toán suy diễn haplotype, bài toán tìm tập hạt giống có cách tối ưu, tìm tham số trong phương pháp SVM (*Support Vector Machine - SVM*) dùng cho bài toán dự báo hoạt động điều tiết gen. Ưu điểm nổi trội của các đề xuất mới được kiểm nghiệm bằng thực nghiệm trên dữ liệu tin cậy.

Các kết quả của luận án đã được công bố trong 7 báo cáo hội nghị quốc tế [20-26], một bài báo ở tạp chí "*Tin học và điều khiển học"* [1] và một hội thảo toàn quốc "*Các chủ đề chọn lọc của công nghệ thông tin* "[2].

Ngoài phần kết luận, luận án được tổ chức như sau. Chương 1 giới thiệu phát biểu bài toán tối ưu tổ hợp dạng tổng quát. Những nét chính của phương pháp tối ưu đàn kiến được giới thiệu trong chương 2. Chương 3, dựa trên phân tích toán học về

biến thiên vết mùi, luận án đề xuất các thuật toán mới: MLAS, SMMAS và 3-LAS. Hiệu quả của thuật toán được kiểm nghiệm trên hai bài toán cổ điển TSP và UBQP. Chương 4 trình bày thuật toán ACOHAP giải bài toán suy diễn haplotype và so sánh hiệu quả của nó với hai thuật toán thông dụng, được xem là tốt nhất hiện nay: RPoly [32] và CollHap [68]. Chương 5 trình bày thuật toán AcoSeeD giải bài toán tìm tập hạt giống có cách tối ưu và so sánh hiệu quả của nó với hai thuật toán SpEED [50] và SpEEDfast [51], được xem là tốt nhất hiện nay. Chương 6 giới thiệu lược đồ ACOSVM và GASVM sử dụng phương pháp ACO và thuật toán di truyền để cải tiến dự báo hoạt động điều tiết gen. Hiệu quả của nó được so sánh với phương pháp lưới thông dụng đã được Zinzen đã đề xuất sử dụng trong [71].

Chương 1. TỐI ƯU TỔ HỢP

Trong các bài toán thực tế cũng như trong lý thuyết, ta thường phải tìm các giá trị cho các biến rời rạc để cực trị hàm mục tiêu nào đó. Các bài toán này thường dễ phát biểu nhưng lại khó giải do chúng thuộc loại tối ưu tổ hợp (TƯTH) NP-khó. Chương này giới thiệu các bài toán tối ưu tổ hợp dưới dạng tổng quát, sẽ sử dụng trong phương pháp tối ưu đàn kiến, các ví dụ minh họa và những vấn đề liên quan cần dùng về sau.

1.1. Bài toán tối ưu tổ hợp tổng quát

Trong đời sống và trong các hệ thông tin, ta thường phải giải nhiều bài toán tối ưu tổ hợp quan trọng. Chẳng hạn như: tìm đường đi ngắn nhất nối hai điểm trên một đồ thị đã cho, lập kế hoạch phân phối nguồn hàng tới nơi tiêu thụ với chi phí cực tiểu, lập thời khóa biểu cho giáo viên và học sinh thuận lợi nhất, định tuyến cho các gói dữ liệu trong Internet, lập lịch hợp lý cho các hệ thống sản xuất, đối sánh các chuỗi gen trong sinh học phân tử v.v…

Về mặt hình thức, mỗi bài toán TƯTH ứng với một bộ ba (S, f, Ω), trong đó S là tập hữu hạn trạng thái (lời giải tiềm năng hay phương án), f là hàm mục tiêu xác định trên S, còn Ω là tập các ràng buộc (xem [31]). Mỗi phương án $s \in S$ thỏa mãn các ràng buộc Ω gọi là phương án (hay lời giải) chấp nhận được. Mục đích của ta là tìm phương án chấp nhận được s^* tối ưu hóa toàn cục hàm mục tiêu f. Chẳng hạn với bài toán cực tiểu thì $f(s^*) \leq f(s)$ với mọi phương án chấp nhận được s. Đối với mỗi bài toán, đều có thể chỉ ra một tập hữu hạn gồm n thành phần $C = \{c_1, \dots, c_n\}$ sao cho mỗi phương án s trong S đều biểu diễn được nhờ liên kết các thành phần trong nó. Cụ thể hơn, các tập S, C và Ω có các đặc tính sau:

1) Ký hiệu X là tập các vectơ trên C có 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 vectơ trong X như ở điểm 2).

2) Tồn tại tập con X^* của X và ánh xạ φ từ X^* lên S sao cho $\varphi^{-1}(s)$ 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 thủ tục mở rộng tuần tự dưới đây.

3) Từ C_0 ta mở rộng tuần tự thành X^* như sau:

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

ii) Giả sử $x_k = < u_0, \ldots, u_k >$ là mở rộng được và chưa thuộc X^*. Từ tập ràng buộc Ω, xác định 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.

iii) Áp dụng thủ tục mở rộng từ các phần tử $u_0 \in C_0$ cho phép ta xây dựng được mọi phần tử của X^*.

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

Chú ý.

1) Trong bài toán suy diễn haplotype ở chương 4, mỗi lời giải được biểu diễn qua $2n$ xâu độ dài m. Cách biểu diễn này không mâu thuẫn với phát biểu bài toán ở trên vì $2n$ xâu này ứng với một vectơ có độ dài $2n.m$, trong đó mỗi thành phần của vectơ tương ứng với một ký tự trong các xâu con của xâu kết hợp.
2) Với các bài toán TƯTH có dạng giải tích: Tìm cực trị hàm $F(x_1, \ldots, x_n)$ trong đó mỗi biến x_i $(i \leq n)$ nhận giá trị trong tập hữu hạn V_i tương ứng và các biến

này thỏa mãn các ràng buộc Ω nào đó, thì C là tập $V = \bigcup_{i=1}^{n} V_i$ và X là các vectơ n-chiều $x = (x_1, \dots, x_n)$, trong đó thành phần x_i nhận giá trị trong tập V_i, C_0 là tập V_1 còn X^* là tập các vectơ thỏa mãn các ràng buộc Ω.

1.2. Các ví dụ

Để thuận tiện trong các trình bày về sau, mục này giới thiệu hai bài toán TƯTH điển hình: Bài toán người chào hàng (*Traveling Salesman Problem - TSP*) và bài toán Quy hoạch toàn phương nhị phân không ràng buộc (*Unconstrained Binary Quadratic Programming - UBQP*).

1.2.1. Bài toán người chào hàng

Bài toán người chào hàng (*Traveling Salesman Problem - TSP*) là bài toán TƯTH điển hình, được nghiên cứu nhiều và được xem là bài toán chuẩn để đánh giá hiệu quả các lược đồ giải bài toán TƯTH mới (xem [30,31]).

Bài toán được phát biểu như sau*:*

Có một tập gồm n thành phố (hoặc điểm tiêu thụ) $C = \{c_i\}_{i=1}^{n}$*, độ dài đường đi trực tiếp từ* c_i *đến* c_j *là* $d_{i,j}$*. Một người chào hàng muốn tìm một hành trình ngắn nhất từ nơi ở, đi qua mỗi thành phố đúng một lần để giới thiệu sản phẩm cho khách hàng, sau đó trở về thành phố xuất phát.*

Như vậy, bài toán này chính là bài toán tìm chu trình Hamilton có độ dài ngắn nhất trên đồ thị đầy đủ có trọng số $G = (V, E)$, trong đó V là tập đỉnh với nhãn là các thành phố trong C, E là các cạnh nối các thành phố tương ứng, độ dài các cạnh chính là độ dài đường đi giữa các thành phố. Trong trường hợp này, tập S sẽ là các chu trình Hamilton trên G, f là độ dài của chu trình, Ω là ràng buộc đòi hỏi chu trình là chu trình Hamilton (qua tất cả các đỉnh, mỗi đỉnh đúng một lần), C là tập thành phố được xét

(trùng với V), C_0 trùng với C, tập X là vectơ độ dài n: $x = (x_1, \dots, x_n)$ với $x_i \in C \; \forall \; i \leq n$, còn X^* là các vectơ trong đó x_i khác x_j đối với mọi cặp (i, j).

Do đó, lời giải tối ưu của bài toán TSP là một hoán vị π của tập đỉnh $\{c_1, c_2, .., c_n\}$ sao cho hàm độ dài $f(\pi)$ là nhỏ nhất, trong đó $f(\pi)$ được tính theo (1.1):

$$f(\pi) = \sum_{i=1}^{n-1} d(\pi(i), \pi(i+1)) + \; d(\pi(n), \pi(1)) \qquad (1.1)$$

ở đây $d(u, v)$ là khoảng cách từ u đến v.

Bài toán TSP được xem là bài toán chuẩn để kiểm định hiệu quả của các phương pháp giải bài toán TƯTH mới với thư viện dữ liệu chuẩn TSPLIB (Reinelt, 1991) tại địa chỉ [77] (Dữ liệu trong nó sẽ được sử dụng trong luận án này).

Bài toán này có nhiều ứng dụng thực tiễn, chẳng hạn như: khoan các lỗ trên bảng mạch in (Reinelt, 1994) hay định vị các thiết bị X-quang (Bland & Shallcross, 1989)… [31].

1.2.2. Bài toán quy hoạch toàn phương nhị phân không ràng buộc

Bài toán quy hoạch toàn phương nhị phân không ràng buộc (*Unconstrained Binary Quadratic Programming - UBQP*) được phát biểu như sau:

Cho ma trận $Q = (q_{ij})$ là ma trận đối xứng kích thước $n \times n$. Cần tìm vectơ nhị phân x gồm n thành phần, $x = (x_1, x_2, .., x_n)$, trong đó $x_i = 0$ hoặc 1 sao cho hàm $f(x)$ đạt giá trị lớn nhất:

$$f(x) = x^t Q x = \sum_{i=1}^{n} \sum_{j=1}^{n} q_{ij} x_i x_j \qquad (1.2)$$

Trong bài toán này, tập S là tập các vectơ nhị phân độ dài n, hàm f đã xác định như trên, tập ràng buộc Ω là rỗng. $C = \{0,1\}$, C_0 trùng với C, tập X là vectơ độ dài n: $x = (x_1, \dots, x_n)$ với $x_i \in C$ $(i = 1,2, .., n)$, còn X^* trùng với X.

1.3. Các cách tiếp cận

Trên đây cho thấy các bài toán TƯTH có thể đưa về bài toán tìm kiếm trên đồ thị. Các bài toán này có thể giải đúng hoặc gần đúng. Với những bài toán cỡ nhỏ hoặc có dạng đặc biệt người ta có thể tìm lời giải tối ưu nhờ tìm kiếm vét cạn hoặc bằng một thuật toán với thời gian đa thức, được xây dựng dựa trên các phân tích toán học. Nhiều bài toán trong số đó là NP-khó, nên với các bài toán cỡ lớn, người ta phải tìm lời giải gần đúng. Các thuật toán giải gần đúng các bài toán TƯTH khó thường dựa trên 2 kỹ thuật cơ bản: heuristic cấu trúc (construction heuristic) và tìm kiếm cục bộ (local search).

1.3.1. Heuristic cấu trúc

Khi không thể tìm được lời giải tối ưu của bài toán, trong thực hành người ta tìm lời giải gần đúng. Một kỹ thuật hay được dùng là heuristic cấu trúc, trong đó lời giải của bài toán TƯTH được xây dựng theo cách mở rộng tuần tự. Từ thành phần khởi tạo trong tập C_0 ở mục 1.1, từng bước mở rộng không quay lui, bằng cách thêm vào các thành phần mới theo phương thức ngẫu nhiên hay tất định dựa trên các quy tắc heuristic đã chọn. Các quy tắc heuristic này thường được xây dựng dựa trên các kết quả phân tích toán học hoặc kinh nghiệm. Phương pháp heuristic cấu trúc tham ăn sau đây cho ta hình dung được cách tiếp cận này (Hình 1.1).

```
Procedure Heuristic cấu trúc tham ăn;
Begin
        s_p ← chọn thành phần u_0 trong C_0;
        while (chưa xây dựng xong lời giải) do
                c ← GreedyComponent(s_p);
                s_p ← s_p ^ c;
        end-while
        s ← s_p;
        Đưa ra lời giải s;
End;
```

Hình 1.1: Phương pháp heuristic cấu trúc tham ăn

trong đó GreedyComponent(s_p) có nghĩa là chọn thành phần bổ sung vào s_p theo quy tắc heuristic đã có. Ký hiệu s_p ^ c là kết quả phép toán thêm thành phần c vào s_p.

Dễ dàng hình dung phương pháp này khi áp dụng thuật toán cho bài toán TSP với đồ thị đầy đủ và sử dụng quy tắc heuristic *láng giềng gần nhất để chọn đỉnh thêm vào* (tức là chọn đỉnh gần nhất chưa đi qua để thêm vào hành trình). Các thuật toán này có ưu điểm là tốn ít thời gian chạy nhưng nhược điểm chính là không cải tiến lời giải được.

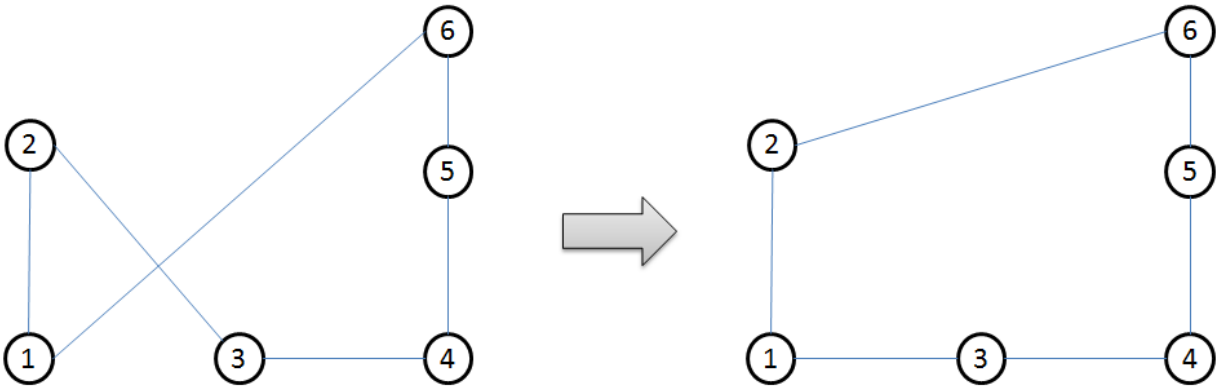
1.3.2. Tìm kiếm cục bộ

Kỹ thuật tìm kiếm cục bộ hay còn gọi là tìm kiếm địa phương, thực hiện bằng cách bắt đầu từ một phương án chấp nhận được, lặp lại bước cải tiến lời giải nhờ các thay đổi cục bộ. Để thực hiện kỹ thuật này, ta cần xác định được *cấu trúc lân cận* của mỗi phương án (lời giải) đang xét, tức là những phương án chấp nhận được, gần với nó nhất, nhờ thay đổi một số thành phần. Cách thường dùng là lân cận *k-thay đổi*, tức là *lân cận* bao gồm các phương án chấp nhận được khác với phương án đang xét nhờ thay đổi nhiều nhất k thành phần.

Ví dụ. Lân cận *2-thay đổi* của một lời giải s trong bài toán TSP bao gồm tất cả các lời giải s' có thể nhận được từ s bằng cách đổi hai cạnh. Hình 1.2 chỉ ra một ví dụ một lời giải nhận được bằng cách thay hai cạnh (2,3), (1,6) bằng hai cạnh (1,3), (2,6).

Việc cải tiến trong các bước lặp thường chọn theo phương pháp leo đồi dựa theo hai chiến lược: Chiến lược *tốt nhất* và chiến lược *tốt hơn*. Với chiến lược *tốt nhất*, người ta thực hiện chọn lời giải tốt nhất trong lân cận để làm lời giải cải tiến. Tuy nhiên, khi bài toán cỡ lớn có thể không tìm được lời giải tốt nhất do bị hạn chế về thời gian. Còn với chiến lược *tốt hơn*, ta chọn phương án đầu tiên trong lân cận, cải thiện

được hàm mục tiêu. Nhược điểm của tìm kiếm cục bộ là thường chỉ cho cực trị địa phương.



Hình 1.2: Lời giải nhận được nhờ thay 2 cạnh (2,3), (1,6) bằng (1,3), (2,6)

Các kỹ thuật trên thường được kết hợp, tạo thành các hệ lai trong các phương pháp mô phỏng tự nhiên dựa trên quần thể, chẳng hạn như thuật toán di truyền (GA) hoặc tối ưu đàn kiến (ACO).

1.3.3. Phương pháp metaheuristic

Phương pháp metaheuristic là một phương pháp heuristic tổng quát được thiết kế, định hướng cho các thuật toán cụ thể (bao gồm cả heuristic cấu trúc và tìm kiếm cục bộ). Như vậy, một metaheuristic là một lược đồ thuật toán tổng quát ứng dụng cho các bài toán tối ưu khác nhau, với một chút sửa đổi cho phù hợp với từng bài toán.

Memetic là một mô hình theo phương pháp metaheuristic. Trong các thuật toán được thiết kế theo memetic, người ta tạo ra nhiều thế hệ quần thể lời giải chấp nhận được. Trong mỗi quần thể của thế hệ tương ứng, ta chỉ chọn ra một số lời giải (chẳng hạn lời giải tốt nhất) để thực hiện tìm kiếm cục bộ nhằm cải thiện chất lượng. Quá trình tiến hóa này cho ta tìm được lời giải tốt nhất có thể. Hình 1.3 mô tả một thuật toán memetic sử dụng tính toán tiến hóa (*Evolutionary Computing - EC*):

```
Proedure Thuật toán memetic-EC;
Begin
    Initialize: Tạo ra quần thể đầu tiên;
    while điều kiện dừng chưa thỏa mãn do
        Đánh giá các cá thể trong quần thể;
        Thực hiện tiến hóa quần thể nhờ các toán tử cho trước;
        Chọn tập con Ω_il để cải tiến nhờ thủ tục tìm kiếm cục bộ;
        for mỗi cá thể trong Ω_il do
            Thực hiện tìm kiếm cục bộ;
        end-for
        Chọn phần tử tốt nhất;
    end-while;
    Đưa ra lời giải tốt nhất;
End;
```

Hình 1.3: Thuật toán memetic sử dụng EC

Trong ứng dụng thực tế, các thuật toán ACO thường được kết hợp với tìm kiếm cục bộ theo mô hình memetic này.

1.4. Kết luận chương

Các bài toán TƯTH (S, f, Ω) nhằm tìm cực trị hàm f trên tập hữu hạn trạng thái S, thỏa mãn ràng buộc Ω, có vai trò quan trọng trong nghiên cứu lý thuyết và ứng dụng. Nhiều bài toán trong chúng thuộc loại NP-khó, khi đó với các bài toán cỡ lớn thì không giải đúng được. Đã có nhiều phương pháp để giải quyết được đề xuất, trong đó các phương pháp giải gần đúng với các kỹ thuật bổ trợ như tìm kiếm cục bộ, memetic đang được sử dụng rộng rãi.

Về mặt giải tích, các bài toán TƯTH có dạng tường minh: tìm cực trị hàm $f(x)$, trong đó x là vectơ có các thuộc tính nhận giá trị trên tập hữu hạn và thỏa mãn tập ràng buộc Ω. Các bài toán này có thể đưa được về dạng tìm kiếm tuần tự đường đi trên đồ thị, cách phát biểu này sẽ được dùng cho phương pháp ACO.

Chương 2. PHƯƠNG PHÁP TỐI ƯU 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 phương pháp này được cải tiến đa dạng và có nhiều ứng dụng. Trước khi giới thiệu phương pháp ACO, luận án sẽ giới thiệu phương thức trao đổi thông tin gián tiếp của kiến tự nhiên và mô hình kiến nhân tạo.

2.1. Từ kiến tự nhiên 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 cách đàn kiến tìm đường đi như thế nào mà có thể giải quyết được các vấn đề tối ưu hóa.

2.1.1. Kiến tự nhiên

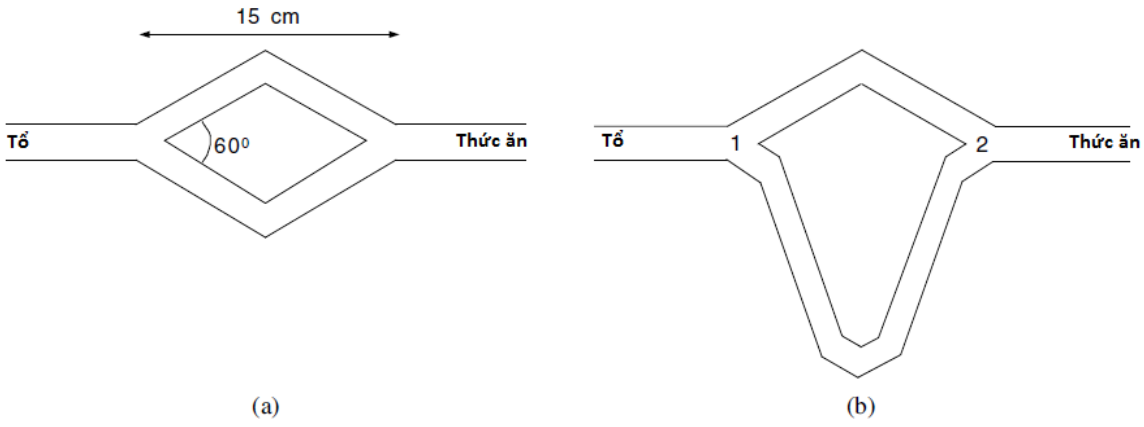
Trên đường đi, mỗi con kiến để lại một chất hóa học gọi là vết mùi (pheromone) dùng để đánh dấu đường đi. Bằng cách cảm nhận vết mùi, kiến có thể lần theo đường đi đến nguồn thức ăn được các con kiến khác khám phá theo phương thức chọn ngẫu nhiên có định hướng theo nồng độ vết mùi. Kiến chịu ảnh hưởng vết mùi của các con kiến khác chính là ý tưởng 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. Thực nghiệm, được thiết kế bởi Deneubourg và các đồng nghiệp [31], dùng một chiếc cầu đôi nối từ tổ kiến tới nguồn thức ăn, như minh họa trong hình 2.1.

Họ đã thực nghiệm với tỉ lệ độ dài đường $r = \frac{l_l}{l_s}$ giữa hai nhánh khác nhau của chiếc cầu đôi, 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 2.1.a). Ban đầu, kiến lựa chọn đường đi một cách tự do 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 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 nhánh còn 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 thấy là sự tương tác cục bộ giữa các con kiến với thông tin gián tiếp là vết mùi để lại cho phép điều chỉnh hoạt động vĩ mô của đàn kiến.

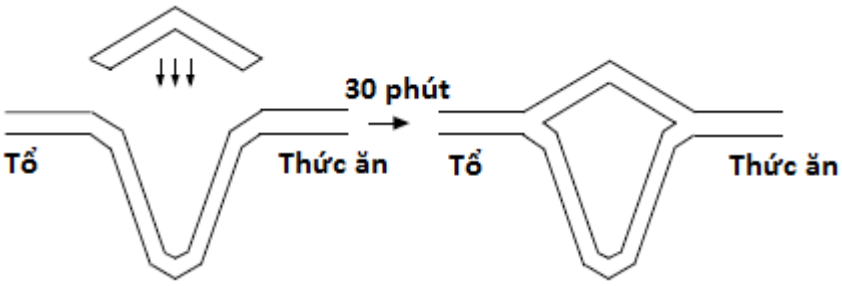


Hình 2.1: Thực nghiệm cây cầu đôi

(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 của nhánh dài gấp đôi độ dài nhánh ngắn (tỉ lệ $r = 2$). Trong trường hợp này, sau một thời gian tất cả các con kiến đều chọn đoạn đường ngắn hơn. Cũng như trong thực nghiệm thứ nhất, ban đầu đàn kiến lựa chọn hai nhánh đi như nhau, một nửa số kiến đi theo nhánh ngắn và một nửa đi theo nhánh dài (mặc dù trên thực tế, do tính ngẫu nhiên có thể một nhánh nào đó được nhiều kiến lựa chọn hơn nhánh kia). Nhưng thực nghiệm này có điểm khác biệt quan trọng với thực nghiệm thứ nhất: Những kiến lựa chọn đi theo nhánh ngắn sẽ nhanh chóng quay trở lại tổ và khi phải lựa chọn giữa nhánh ngắn và nhánh dài, kiến sẽ thấy nồng độ mùi trên nhánh ngắn cao hơn nồng độ mùi trên nhánh dài, do đó sẽ ưu tiên lựa chọn đi theo nhánh ngắn hơn. Tuy nhiên, trong thời gian đầu không phải tất cả các kiến đều đi theo nhánh ngắn hơn. Phải mất một khoảng thời gian tiếp theo nữa bầy kiến mới lựa chọn đi theo nhánh ngắn. Điều này minh chứng bầy kiến đã sử dụng phương thức thăm dò, tìm đường mới.

Một điểm thú vị nữa là quan sát xem sẽ xảy ra điều gì khi quá trình tìm kiếm đang 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, thêm một nhánh ngắn (xem hình 2.2). Trong trường hợp này, nhánh ngắn thường không được kiến chọn mà chúng tập trung đi trên nhánh dài. Điều này có thể giải thích như sau: nồng độ vết mùi trên cạnh dài cao và sự bay hơi của vết mùi diễn ra chậm nên đại đa số các con kiến vẫn lựa chọn nhánh dài (có nồng độ vết mùi cao). Hành vi này tiếp tục được củng cố và kiến chọn đi theo nhánh dài, ngay cả khi có một nhánh ngắn xuất hiện. Việc bay hơi vết mùi là cơ chế tiện lợi cho việc tìm đường mới, nghĩa là việc bay hơi có thể giúp kiến quên đi đường đi tối ưu cục bộ đã được tìm thấy trước đây để tìm khám phá đường đi mới, tốt hơn.



Hình 2.2: Thí nghiệm bổ xung

Ban đầu chỉ có một nhánh và sau 30 phút thêm nhánh ngắn hơn

2.1.2. Kiến nhân tạo

Thực nghiệm cây cầu đôi cho thấy đàn kiến 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 cho phép liên tưởng tới cách học tăng cường (reinforcement learning) trong bài toán chọn tác động tối ưu [3], gợi mở 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 các tác tử (agent) là đàn kiến nhân tạo.

Tuy nhiên, trong các bài toán ứng dụng các đồ thị thường phức tạp hơn. Từ mỗi đỉnh có thể có nhiều cạnh, nên nếu mô phỏng thực sự hành vi của đàn kiến tự nhiên nhiều con kiến sẽ đi luẩn quẩn và do đó hiệu quả thuật toán sẽ rất kém. Vì vậy, người ta dùng kỹ thuật đa tác tử (multiagent) mô phỏng đàn kiến nhân tạo, trong đó mỗi con kiến nhân tạo có khả năng nhiều hơn so với kiến tự nhiên. Kiến nhân tạo (về sau trong luận án ta sẽ gọi đơn giản 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, kiến có thể trao đổi thông tin với nhau, thực hiện tính toán cần thiết, cập nhật mùi…

Sử dụng mô hình kiến nhân tạo này, Dorigo (1991) [28] đã xây dựng thuật toán *hệ kiến* (AS) giải bài toán người chào hàng. Hiệu quả của thuật toán so với các phương pháp mô phỏng tự nhiên khác như SA và GA đã được kiểm chứng bằng thực nghiệm.

Thuật toán này về sau được phát triển và có nhiều ứng dụng phong phú, được gọi chung là phương pháp ACO.

2.2. Phương pháp ACO cho bài toán TƯTH tổng quát

Mục này giới thiệu tóm lược phương pháp tối ưu đàn kiến. Để biết chi tiết hơn, có thể xem [31]. Trước khi mô tả thuật toán tổng quát, ta tìm hiểu đồ thị cấu trúc cho bài toán tối ưu tổ hợp.

2.2.1. Đồ thị cấu trúc

Xét bài toán TƯTH tổng quát được nêu trong mục 1.1 dưới dạng bài toán cực tiểu hoá (S, f, Ω), trong đó S là tập hữu hạn trạng thái (lời giải tiềm năng hay phương án), f là hàm mục tiêu xác định trên S, còn Ω là các ràng buộc để xác định tập S có các thành phần được lấy từ tập hữu hạn C. Các tập S, C và Ω có các đặc tính sau:

1) Ký hiệu X là tập các vectơ 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 bởi ít nhất một vectơ trong X như ở điểm 2).

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

3) Từ C_0 ta mở rộng tuần tự thành X^* như sau:

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

   ii) Giả sử $x_k = < u_0, \ldots, u_k >$ là mở rộng được và chưa thuộc X^*. Từ tập ràng buộc Ω, xác định 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.

iii) Áp dụng thủ tục mở rộng từ các phần tử $u_0 \in C_0$ cho phép ta xây dựng được mọi phần tử của X^*.

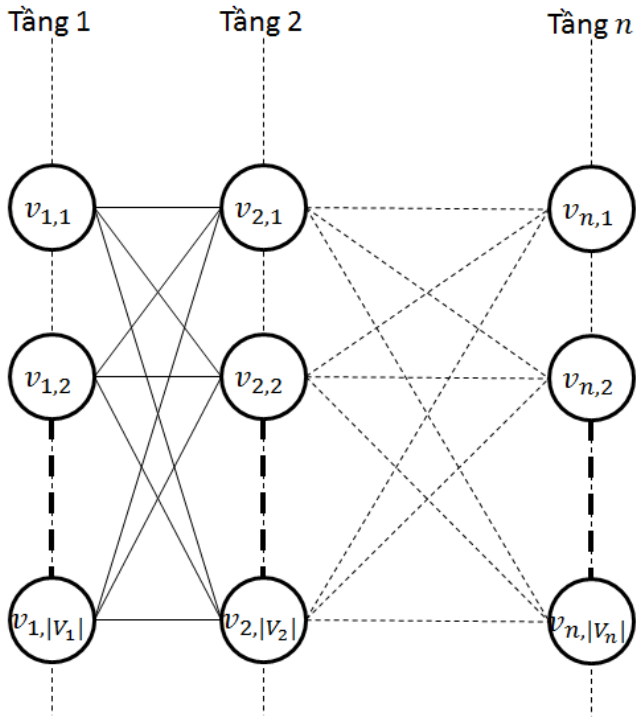
Như đã nói trong chương 1, mỗi bài toán TƯTH được xem như một bài toán tìm kiếm vectơ độ dài không quá h trên đồ thị đầy đủ có các đỉnh được gán nhãn trong tập C. Để tìm các lời giải chấp nhận được, ta xây dựng đồ thị đầy đủ với tập đỉnh V, mỗi đỉnh của nó tương ứng với mỗi thành phần của C. Các lời giải chấp nhận được sẽ là các vectơ được xác định theo thủ tục mở rộng tuần tự hay mở rộng ngẫu nhiên, như đã được mô tả chi tiết trong mục 2.2.2.

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

Mỗi cạnh nối 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 trạng thái hiện tại x_k là i (theo thủ tục mở rộng tuần tự đã nêu ở trên). Ký hiệu H là vectơ các trọng số heuristic của cạnh (trong bài toán TSP đó là vectơ có các thành phần là nghịch đảo của độ dài cạnh tương ứng), còn τ là vectơ biểu thị các thông tin học tăng cường $\tau_{i,j}$ (trong luận án từ nay về sau gọi là vết mùi, ban đầu được khởi tạo giá trị τ_0>0). Trường hợp đặc biệt $h_{i,j}$ và $\tau_{i,j}$ chỉ phụ thuộc vào j, các thông tin này sẽ gắn với các đỉnh. Không làm mất tính tổng quát, ta xét trường hợp các thông tin này gắn vào các cạnh.

Ta gọi đồ thị $G = (V, E, H, \tau)$ là *đồ thị cấu trúc* của bài toán tối ưu tổ hợp, trong đó V là tập đỉnh, E là tập cạnh, H và τ là các thông tin gắn với cạnh. Từ các cạnh, xây dựng tập X^* nhờ mở rộng tập C_0 theo thủ tục tuần tự. Nếu không có thông tin heuristic thì ta xem H có các thành phần như nhau và bằng 1.

Trường hợp tổng quát, G là đồ thị đầy đủ. Tuy nhiên, tùy theo ràng buộc của bài toán, các cạnh có thể lược bớt để giảm miền tìm kiếm lời giải theo thủ tục mở rộng tuần tự. Chẳng hạn, với bài toán tìm cực trị của hàm giải tích $f(x_1, \dots, x_n)$, với x_i thuộc tập giá trị hữu hạn V_i, đồ thị cấu trúc có n tầng, 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 2.3. Khi đó tập C_0 là tập V_1, mỗi mở rộng tuần tự của lời giải sẽ được xây dựng từ một đỉnh thuộc tập này.



Hình 2.3: Đồ thị cấu trúc tổng quát cho bài toán cực trị hàm $f(x_1, \dots x_n)$

2.2.2. Mô tả thuật toán ACO tổng quát

Sử dụng điều kiện kết thúc (có thể theo số bước lặp hoặc/và giới hạn thời gian chạy), ta dùng đàn kiến có m con, tiến hành lặp quá trình xây dựng lời giải trên đồ thị

cấu trúc $G = (V, E, H, \tau)$ như sau: Tại mỗi lần lặp, 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. 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 và chưa thuộc X^*. Từ tập ràng buộc Ω, xác định 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. Đỉnh $j = u_{k+1}$ để mở rộng, được chọn với xác suất $P(j)$ như sau:

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

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 $s^r = \varphi(x^r) \in S$.

Do giả thiết 2) và iii) ở trên, về sau ta sẽ xem x^r và s^r như nhau và không phân biệt X^* với S để tiện trình bày.

Cập nhật mùi:

Tùy theo chất lượng của lời giải tìm được, 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 các quy tắc cập nhật mùi khác nhau sẽ cho các thuật toán khác nhau. Vì vậy, quy tắc cập nhật mùi thường dùng làm tên gọi thuật toán. Các quy tắc thông dụng sẽ được nêu trong mục 2.2.2, đa số chúng đều có dạng:

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

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 mô tả trong hình 2.4.

```
Procedure Thuật toán ACO;
Begin
    Khởi tạo tham số, ma trận mùi, khởi tạo m con kiến;
    repeat
        for k = 1 to m do
            Kiến k xây dựng lời giải;
        end-for
        Cập nhật mùi;
        Cập nhật lời giải tốt nhất;
    until (Điều kiện kết thúc);
    Đưa ra lời giải tốt nhất;
End;
```

Hình 2.4: Thuật toán ACO

Nhận xét chung về các thuật toán ACO

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

1) Việc tìm kiếm ngẫu nhiên dựa trên các thông tin heuristic trở nên linh hoạt và mềm dẻo trên miền rộng hơn so với các phương pháp heuristic đã có. Do đó, cho ta lời giải tốt hơn và có thể tìm được lời giải tối ưu.

2) Học tăng cường thông qua thông tin về cường độ vết mùi cho phép 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 các bài toán cụ thể, ba yếu tố sau có ảnh hưởng quyết định đến hiệu quả thuật toán:

1) *Xây dựng đồ thị cấu trúc thích hợp*. Trong mục 2.2.1 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, không gian tìm kiếm quá rộng, đòi hỏi ta sử dụng các ràng buộc Ω một cách hợp lý để giảm miền tìm kiếm của kiến. Cách xử lý bài toán suy diễn haplotype ở chương 4 minh họa cho điều này.

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

3) *Chọn quy tắc cập nhật mùi*. Quy tắc cập nhật mùi thể hiện chiến lược học của thuật toán. Trong khi đồ thị cấu trúc và thông tin heuristic phụ thuộc vào bài toán cụ thể, quy tắc cập nhật mùi lại là yếu tố phổ dụng và thường dùng để đặt tên cho thuật toán. Có nhiều quy tắc cập nhật mùi đã được đề xuất, luận án này sẽ tìm quy tắc thích hợp cho hai loại bài toán, tùy theo thông tin heuristic ảnh hưởng nhiều hay ít tới thủ tục tìm kiếm lời giải.

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

2.3. Phương pháp ACO giải bài toán người chào hàng

Bài toán người chào hàng (*Traveling Salesman Problem - TSP*) có nhiều ứng dụng trong thực tế. Nó thuộc loại NP-khó (xem [29-31]) 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 thuật toán *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.

Ta trở lại với bài toán TSP đã phát biểu trong chương trước và mô hình toán học của nó.

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

Mục tiêu của bài toán TSP là tìm một chu trình Hamilton có độ dài ngắn nhất. Do đó, lời giải tối ưu của bài toán TSP là một hoán vị π của tập n đỉnh $\{1,2,..,n\}$ ($n = |V|$) sao cho hàm độ dài $f(\pi)$ là nhỏ nhất, trong đó $f(\pi)$ bằng: $f(\pi) = \sum_{i=1}^{n-1} d_{\pi(i)\pi(i+1)} + d_{\pi(n)\pi(1)}$.

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 gán trên các cạnh và do đó τ_{ij} dùng để chỉ thông tin học tăng cường phục vụ mở rộng tuần tự lời giải từ i đến j.

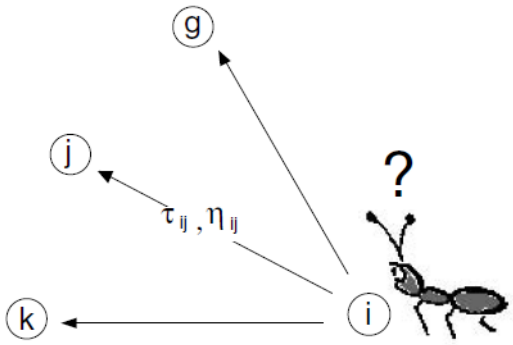
Khi đó, đồ thị cấu trúc của bài toán là đồ thị đầy $G = (V, E, H, \tau)$. Nếu $x_k =< u_0, \ldots, u_k >$ là đường đi mở rộng được, tức là các đỉnh u_i đều khác nhau và $k < n$) thì $J(x_k) = N_{u_k}$ là các đỉnh mà đường x_k chưa đi đến. Các thuật toán ACO cho bài toán TSP đều thực hiệu trên đồ thị cấu trúc này. Các kết quả thực nghiệm (xem [28-30]) đã chỉ ra hiệu quả nổi trội của các 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 $G = (V, E, H, \tau)$ có không gian tìm kiếm chính là tập các hành trình có thể. Ràng buộc Ω sẽ được thỏa mãn khi hành trình do kiến xây dựng là một hành trình đúng (là chu trình Hamilton), biểu diễn một hoán vị các chỉ số của các thành phố.

Quá trình kiến xây dựng lời giải theo thủ tục bước ngẫu nhiên được thực hiện như sau:

- Lựa chọn thành phố xuất phát (có thể theo một số tiêu chí nào đó).
- Thực hiện lặp bước mở rộng bằng cách thêm một thành phố kiến chưa đi qua (xem hình 2.5) cho đến khi tất cả các thành phố đều được thăm: tính xác suất lựa chọn đỉnh mới theo giá trị thông tin mùi và thông tin heuristic, chọn ngẫu nhiên đỉnh mới thêm vào theo phân bố xác suất ngẫu nhiên này.
- Quay trở lại thành phố xuất phát.



Hình 2.5: Lựa chọn đỉnh đi tiếp theo

Sau khi tất cả kiến xây dựng xong hành trình, các con kiến sẽ để lại vết mùi trên các cạnh kiến đi qua. Trong một số trường hợp, trước khi thêm mùi, các hành trình xây dựng được có thể được 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 mô tả trong hình 2.6 như sau:

Procedure Thuật toán ACOTSP;
Dữ liệu vào: $G = (V, E)$
Kết quả ra: Một chu trình và tổng độ dài của nó;
Begin
    Khởi tạo tham số, ma trận mùi, khởi tạo m con kiến;
    repeat
        for $k = 1$ **to** m **do**
            Kiến k xây dựng lời giải;
            Cải tiến lời giải do kiến k xây dựng bằng tìm kiếm cục bộ;
        end-for
        Cập nhật mùi;
        Cập nhật lời giải tốt nhất;
    until (Điều kiện kết thúc);
    Đưa ra lời giải tốt nhất;
End;

Hình 2.6: Thuật toán ACO giải bài toán TSP có sử dụng tìm kiếm cục bộ

Thuật toán ACO đầu tiên (thuật toán AS) giải bài toán TSP đã đạt được hiệu quả khả quan, 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 về sau này đều dựa trên thuật toán AS. Sự khác biệt chính giữa các thuật toán mở rộng của AS như: thuật toán elitist AS, rank-based AS và MAX-MIN,…là cách thức cập nhật 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 AS.

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)

Các thuật toán ACO thông dụng nhất hiện nay là MMAS và ACS. Tuy hiệu quả của chúng như nhau nhưng MMAS 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 được Dorigo đề xuất (xem [28,31]) là ant-density, ant-quantity và ant-cycle. Ở phiên bản ant-density và ant-quantity, 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 kiến đã xây dựng xong hành trình và lượng mùi được cập nhật phụ

thuộc vào độ dài hành trình mà kiến tìm được. Hai thuật toán ant-density và ant-quantity không hiệu quả so với thuật toán ant-cycle, nên khi nói tới thuật toán AS ta chỉ quan tâm đến phiên bản ant-cycle.

Hai bước chính trong thuật toán AS là xây dựng lời giải của kiến và cập nhật mùi. Trong AS, lời giải tìm được dựa trên phương pháp heuristic (chẳng hạn phương pháp tham ăn) khi xác định vết mùi khởi tạo. Giá trị vết mùi khởi tạo cho tất cả các cạnh là: $\tau_{ij} = \tau_0 = \frac{m}{C^{nn}}$, trong đó m là số kiến, C^{nn} là độ dài lời giải tìm được của thuật toán heuristic. Lý do lựa chọn này là nếu khởi tạo vết mùi τ_0 quá thấp thì quá trình tìm kiếm có khuynh hướng hội tụ về hành trình đầu tiên tìm được, dẫn đến việc tìm kiếm sa lầy vào vùng này, làm cho chất lượng lời giải không tốt. Nếu khởi tạo vết mùi quá cao thì thường phải mất nhiều vòng lặp bay hơi mùi trên các cạnh không tốt và cập nhật bổ sung thêm mùi cho các cạnh tốt mới để thể hướng việc tìm kiếm đến vào vùng không gian có chất lượng tốt.

Xây dựng lời giải

Trong thuật toán AS, m kiến đồng thời xây dựng lời giải. Ban đầu, các con kiến được đặt ngẫu nhiên tại các thành phố. Ở mỗi bước, kiến sử dụng xác suất theo phương thức tỉ lệ ngẫu nhiên (random proportional) để chọn đỉnh tiếp theo. Cụ thể, khi kiến k đang ở đỉnh i sẽ lựa chọn đỉnh j với xác suất:

$$p_{ij}^k = \begin{cases} \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in N_i^k} [\tau_{il}]^\alpha [\eta_{il}]^\beta}, & \text{nếu } j \in N_i^k \\ 0, & \text{nếu } j \overline{\in} N_i^k \end{cases} \quad (2.3)$$

trong đó $\eta_{ij} = \frac{1}{d_{ij}}$ là giá trị thông tin heuristic, 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 kiến k có thể đi đến (là tập các đỉnh kiến k chưa đến, xác suất 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 $\alpha = 0$ thì thành phố gần nhất sẽ ưu tiên được lựa chọn, khi đó thuật toán tương đương với thuật toán chọn ngẫu nhiên theo ưu tiên tỉ lệ nghịch đảo độ dài cạnh, không có học tăng cường. Nếu $\beta = 0$ thì chỉ có thông tin học tăng cường biểu thị qua vết mùi được sử dụng, không có thông tin heuristic. Nếu α lớn, thuật toán nhanh chóng bị tắc nghẽn (tất cả kiến sẽ lựa chọn cũng một hành trình) và lời giải tìm được hội tụ về lời giải tối ưu cục bộ [31].

Để cài đặt, kiến k sẽ duy trì một bộ nhớ M^k chứa thông tin lần lượt các thành phố đã đi qua. Thông tin trong bộ nhớ dùng để xác định các thành phố lân cận phù hợp N_i^k trong (2.3). Hơn nữa, thông tin trong bộ nhớ M^k giúp 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. Theo cách xây dựng song song, tại mỗi bước tất cả kiến sẽ di chuyển sang đỉnh tiếp theo. Trong cách xây dựng tuần tự, lần lượt từng kiến xây dựng lời giải (kiến này xây dựng xong rồi mới đến kiến tiếp theo). Trong AS, cả hai cách xây dựng này là như nhau, không có ảnh hưởng gì đến kết quả. 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 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 trên tất cả các cạnh được thực hiện như sau:

$$\tau_{ij} \leftarrow (1 - \rho)\tau_{ij} \tag{2.4}$$

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 thì vết mùi nhanh chóng bị giảm theo cấp số nhân. Sau khi bay hơi, tất cả kiến sẽ để lại vết mùi trên cạnh đi qua:

$$\tau_{ij} \leftarrow \tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \tag{2.5}$$

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

$$\Delta\tau_{ij}^{k} = \begin{cases} \frac{1}{C^k} & nếu\ cạnh\ (i,j)\ thuộc\ T^k \\ 0 & ngược\ lại \end{cases} \tag{2.6}$$

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. Như vậy, cạnh nào càng có nhiều kiến sử dụng và thuộc hành trình ngắn, sẽ càng được cập nhật mùi nhiều hơn. Do đó sẽ được kiến lựa chọn nhiều hơn trong các bước lặp sau.

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

2.3.2. Hệ đàn kiến ACS

Thuật toán ACS (Ant Colony System) do Dorigo và Gambardella đề xuất năm 1997 [30], khác với AS ở ba điểm chính sau đây:

- Thứ nhất, khai thác kinh nghiệm tìm kiếm mạnh hơn trong AS, thông qua việc sử dụng quy tắc lựa chọn dựa trên thông tin tích lũy nhiều hơn.
- Thứ hai, cơ chế bay hơi mùi và để lại mùi chỉ trên các cạnh thuộc vào lời giải tốt nhất đến lúc đó (*Global-best: G-best*).

- Thứ ba, tăng cường việc thăm do đường mới. Mỗi lần kiến đi qua cạnh (i,j) vết mùi sẽ bị giảm trên cạnh (i,j). Sau đây chúng ta sẽ tìm hiểu chi tiết những điểm khác đã nêu.

Xây dựng lời giải

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

$$j = \begin{cases} argmax_{l \in N_i^k}\{\tau_{il}[\eta_{il}]^{\beta}\}, nếu\ q \leq q_0 \\ J, \qquad\qquad\qquad ngược\ lại \end{cases}, \tag{2.7}$$

trong đó q là một biến ngẫu nhiên, phân bố đều trong [0,1], q_0 ($0 \leq q_0 \leq 1$) là một tham số cho trước và J là một biến ngẫu nhiên, lựa chọn theo phân bố xác suất như trong (2.3) với $\alpha = 1$. Nói cách khác, với xác suất q_0 kiến sẽ 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, ta nói kiến khai thác thông tin đã học). Ta cũng nói với xác suất $(1 - q_0)$ kiến thực hiện khám phá trên các cạnh. Điều chỉnh tham số q_0 sẽ cho phép thay đổi mức độ khai thác và lựa chọn tập trung tìm kiếm quanh lời giải G-best hoặc khám phá các hành trình khác.

Cập nhật mùi toàn cục

Trong ACS chỉ có duy nhất một kiến tìm được lời giải G-best được phép để lại mùi sau mỗi lần lặp. Cụ thể, với các cạnh(i,j) thuộc lời giải G-best được cập nhật như sau:

$$\tau_{ij} \leftarrow (1 - \rho)\tau_{ij} + \rho\Delta\tau_{ij}^{best} \tag{2.8}$$

trong đó $\Delta\tau_{ij}^{best} = \frac{1}{C^{G-best}}$, C^{G-best} là độ dài lời giải G-best. 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^{G-best} (chứ không phải cho tất cả các cạnh như trong AS). Tham số ρ là tham số bay hơi. Không giống như AS, trong (2.4) và (2.5) trong (2.8) vết mùi được để lại sẽ giảm theo tham số ρ. Kết quả của việc cập nhật này là vết mùi được thay đổi theo trung bình trọng số của vết mùi cũ và lượng mùi được để lại.

Trong thí nghiệm, người ta cũng sử dụng chọn lời giải tốt nhất trong bước lặp hiện tại (*Iteration-best: I-best*) để cập nhật mùi.

Cập nhật mùi cục bộ

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

$$\tau_{ij} \leftarrow (1-\xi)\tau_{ij} + \xi\tau_0 \tag{2.9}$$

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 thực 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 tham ăn. Hiệu quả của thuật toán cập nhật mùi cục bộ thể hiện ở chỗ mỗi khi kiến đi qua cạnh (i,j) thì vết mùi trên cạnh (i,j) bị giảm, làm cho kiến ít lựa chọn lại cạnh này. Nói cách khác, việc cập nhật mùi cục bộ làm tăng khả năng khám phá các cạnh chưa được sử dụng. Trên thực tế, hiệu quả của cách cập nhật mùi này chính là ở chỗ không bị tắc nghẽn (nghĩa là, các kiến không bị dồn vào một đường đi nào) như trong AS.

Đối với AS, kiến xây dựng hành trình song song hay tuần tự là không ảnh hưởng gì. Nhưng trong ACS, công việc này có ảnh hưởng, vì ACS sử dụng cơ chế cập nhật mùi cục bộ. Trong thực nghiệm, thuật toán ACS yêu cầu mọi kiến đồng thời xây dựng hành trình, mặc dù không có kết quả thực nghiệm chứng tỏ sự lựa chọn nào tốt hơn.

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. Theo cách này, danh sách ứng cử viên có thể được xây dựng trước khi bắt đầu tìm kiếm và 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 thăm. Trong trường hợp tất cả các thành phố trong danh sách ứng cử viên đều đã 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.

Chú ý. Quy tắc cập nhật mùi của ACS trong [30] khác quy tắc ở trên ở điểm là thực hiện bay hơi trong mỗi lần lặp như AS với các cạnh chưa có kiến đi qua. Theo cách cập nhật này, hiệu quả của MMAS dưới đây tốt hơn ACS.

2.3.2. Hệ kiến Max-Min

Thuật toán MMAS (Max-Min Ant System) do Stutzle và Hoos đề xuất năm 2000 [66] với bốn điểm thay đổi so với AS.

- Thứ nhất, để tăng cường khai thác lời giải tốt nhất tìm được: các cạnh thuộc vào lời giải I-best hoặc G-best được cập nhật mùi. Điều này có thể dẫn đến hiện tượng 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 đã đưa ra miền giới hạn cho vết mùi: vết mùi sẽ bị hạn chế trong khoảng $[\tau_{min}, \tau_{max}]$.
- 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 có hiện tượng tắc nghẽn hoặc không tìm được lời giải tốt hơn sau một số bước.

Cập nhật mùi

Sau khi tất cả kiến xây dựng lời giải, vết mùi được cập nhật bằng thủ tục bay hơi giống như AS (công thức 2.4), và được thêm một lượng mùi cho tất cả các cạnh thuộc lời giải tốt như sau:

$$\tau_{ij} \leftarrow \tau_{ij} + \Delta\tau_{ij}^{best} \tag{2.10}$$

trong đó $\Delta\tau_{ij}^{best} = \frac{1}{C^{G-best}}$ khi dùng G-best hoặc $\Delta\tau_{ij}^{best} = \frac{1}{C^{I-best}}$ khi dùng I-best để cập nhật mùi. Sau đó vết mùi sẽ bị giới hạn trong đoạn $[\tau_{min}, \tau_{max}]$ như sau:

$$\tau_{i,j} = \begin{cases} \tau_{max} & nếu\ \tau_{i,j} > \tau_{max} \\ \tau_{i,j} & nếu\ \tau_{i,j} \in [\tau_{min}, \tau_{max}] \\ \tau_{min} & nếu\ \tau_{i,j} < \tau_{min} \end{cases} \tag{2.11}$$

Nói chung, MMAS dùng cả I-best và G-best 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 thì việc tìm kiếm sẽ sớm định hướng quanh G-best còn khi cập nhật bằng I-best thì số lượng cạnh được cập nhật mùi nhiều do đó việc tìm kiếm giảm tính định hướng hơn.

Giới hạn vết mùi

Trong MMAS sử dụng giới hạn trên τ_{max} và giới hạn dưới τ_{min} đối với vết mùi của tất cả các cạnh để tránh tình trạng tắc nghẽn. Đặc biệt, việc giới hạn vết mùi sẽ có ảnh hưởng đến giới hạn xác suất p_{ij} trong đoạn $[p_{min}, p_{max}]$ phục vụ 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, trong một thời gian dài, 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 cập nhật lại cận trên $\tau_{max} = \frac{1}{\rho C^{G-best}}$ và cận dưới $\tau_{min} = \frac{\tau_{max}}{w}$ (w là một tham số) mỗi khi tìm được lời giải tốt hơn. Kết quả thực nghiệm chỉ ra rằng: để tránh tắc nghẽn, cận dưới τ_{min} đóng vai trò quan trọng hơn cận trên τ_{max}. Tuy nhiên, τ_{max} 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 trên tất cả các cạnh được thiết đặt bằng cận trên của vết mùi τ_{max}. Cách khởi tạo như vậy, kết hợp với tham số bay hơi nhỏ sẽ cho phép làm chậm sự khác biệt vết mùi giữa 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 mỗi khi gặp tình trạng tắc nghẽn (thuật toán kiểm tra tình trạng tắc nghẽn dựa trên sự thống kê vết mùi trên các cạnh) hoặc sau một số bước lặp mà vẫn không tìm được lời giải tốt hơn.

MMAS là thuật toán được nghiên cứu nhiều nhất trong số các thuật toán ACO và nó có rất nhiều mở rộng. Một trong các cải tiến là khi khởi tạo lại vết mùi, cập nhật mùi dựa trên lời giải tốt nhất tìm được tính từ khi khởi tạo lại vết mùi thay cho G-best. Một cải tiến khác là sử dụng luật di chuyển theo kiểu ACS.

2.4. Một số vấn đề liên quan

2.4.1. Đặc tính hội tụ

Gutjahr [36-38] là một trong những người đầu tiên nghiên cứu đặc tính hội tu của thuật toán MMAS, nhưng chưa xét đến yếu tố có sử dụng 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) $\lim_{t\to\infty} w(t) = w^*, \lim_{t\to\infty} P(t) = 1$ (2.12)

2) $\lim_{t\to\infty} \tau_{i,j} = \tau_{max}$ với mọi cạnh (i,j) thuộc lời giải tối ưu (2.13)

Mô hình này của Gutjahr không áp dụng được cho ACS. Trong trường hợp MMAS không sử dụng thông tin heuristic, Stützle và Dorigo [65] đã chứng minh rằng:

$$\forall \varepsilon > 0, \exists\, t \text{ đủ lớn } P(t) > 1 - \varepsilon, \quad (2.14)$$

do đó $$\lim_{t\to\infty} P(t) = 1. \quad (2.15)$$

Các tác giả cũng suy ra rằng kết quả này cũng đúng cho cả thuật toán ACS. Với giả thiết 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 sẽ hội tụ đến τ_{max} , còn vết mùi trên các cạnh không thuộc lời giải sẽ hội tụ về τ_{min} hoặc τ_0.

Plelegrini và Elloro [55] chỉ ra rằng sau một thời gian chạy, đa số vết mùi trên cạnh trở nên bé và chỉ có số ít cạnh có giá trị vết mùi là lớn vượt trội.

2.4.2. Thực hiện song song

Đặc tính tự nhiên của các thuật toán ACO cho phép thực hiện song song theo dữ liệu hoặc theo quần thể (xem [31]). Trên thực tế, có nhiều mô hình song song được sử dụng cho các thuật toán dựa trên quần thể, dễ dàng tương thích với ACO.

2.4.3. ACO kết hợp với tìm kiếm cục bộ

Nhiều tài liệu [10,31,63] chỉ ra rằng với các phương pháp metaheuristic, một cách tiếp cận đầy hứa hẹn cho phép nhận 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 (xem hình 2.6). 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 này. Kết hợp xây dựng lời giải với tìm kiếm cục bộ sẽ là một cách tiếp cận có triển vọng, là do trên thực tế, cách xây dựng lời giải của ACO có sử dụng lân cận khác với tìm kiếm cục bộ. Thực nghiệm cho thấy khả năng kết hợp tìm kiếm cục bộ cải tiến được lời giải là khá cao.

2.4.4. Thông tin heuristic

Ta biết rằng thuật toán ACO mà không sử dụng tìm kiếm cục bộ, thông tin heuristic sẽ rất cần thiết để có được lời giải tốt. Trên thực tế, ở giai đoạn đầu vết mùi được khởi tạo như nhau. Khi đó vết mùi không thể giúp kiến tìm đường đi dẫn tới các lời giải tốt, vì chưa khác nhau nhiều. Vai trò chính của thông tin heuristic là để khắc phục điều này, giúp kiến có thể xây dựng được các hành trình tốt ngay trong giai đoạn đầu. Trong nhiều trường hợp, nhờ sử dụng tìm kiếm cục bộ, kiến vẫn có thể tìm được lời giải tốt ngay trong giai đoạn đầu, không cần sử dụng thông tin heuristic nào cả, mặc dù có làm cho quá trình tìm kiếm chậm hơn.

2.4.5. 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 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 do đó 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 ở mỗi vòng lặp. Khi có sử dụng tìm kiếm cục bộ hoặc thông tin heuristic mạnh, sử dụng nhiều kiến là lãng phí.

2.4.6. Tham số bay hơi

Ở mỗi vòng lặp, khi xây dựng được lời giải tốt (sử dụng tìm kiếm cục bộ hoặc thông tin heuristic mạnh), tham số bay hơi sẽ được xác lập có giá trị lớn, điều này giúp kiến quên đi những lời giải đã xây dựng, tập trung công việc tìm kiếm xung quanh lời giải tốt mới được xây dựng. Trong trường hợp ngược lại, ở mỗi vòng lặp, khả năng kiến tìm được lời giải tốt không cao thì tham số bay hơi phải được thiết lập với giá trị nhỏ.

2.5. Kết luận chương

Phương pháp ACO là một phương pháp metaheuristic được sử dụng rộng rãi để giải các bài toán TƯTH khó, hiệu quả nổi trội của nó 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 kiến xây dựng nhờ thủ tục bước ngẫu nhiên trên đồ thị cấu trúc. Việc tìm kiếm đỉnh mới của đường đi dựa trên kết hợp thông tin heuristic và thông tin học tăng cường biểu thị bởi vết mùi.

Khi áp dụng phương pháp này, ba yếu tố sau đây có vai trò 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 được nghiên cứu sâu hơn nhằm đưa ra các cải tiến hiệu quả.

Chương 3. TÍNH BIẾN THIÊN CỦA VẾT MÙI VÀ CÁC THUẬT TOÁN MỚI

Như đã trình bày trong chương trước, Gutjahr [36-38], Stützle và Dorigo [65] đã xét tính hội tụ theo xác suất tới lời giải tối ưu của MMAS, ACS và sự hội tụ của cường độ vết mùi cho các biến thể của thuật toán MMAS. Các tác giả chưa chưa khảo sát sự hội tụ của cường độ vết mùi đối với ACS.

Tuy nhiên, trong các bài toán tối ưu tổ hợp do số phương án là hữu hạn, nên kết quả xác suất tìm thấy lời giải hội tụ về 1 khi số lần lặp dần ra vô hạn là không có nhiều ý nghĩa. Trong chương này, luận án phân tích chi tiết hơn về các đặc tính biến thiên của vết mùi trong các thuật toán ACO thông dụng và xem xét các quy tắc cập nhật mùi theo cách nhìn *học tăng cường* (reinforcement learning), trên cơ sở đó đề xuất các quy tắc cập nhật mùi mới. Kết quả thực nghiệm trên các bài toán TSP và UBQP cho thấy ưu điểm của các đề xuất này.

Trước khi phân tích toán học, ta biểu diễn lại thuật toán dưới dạng dễ khảo sát hơn.

3.1. Thuật toán tổng quát

Xét một bài toán TƯTH cực tiểu hoá (S, f, Ω) trong mục 2.2 với đồ thị cấu trúc: $G = (V, E, H, \tau)$, trong đó V là tập đỉnh, E là tập các cạnh, H là vectơ các trọng số heuristic của cạnh tương ứng, còn τ là vectơ vết mùi tích luỹ được (ban đầu được khởi tạo bằng τ_0>0), C_0 là tập đỉnh khởi tạo để xây dựng các lời giải chấp nhận được theo thủ tục bước ngẫu nhiên. Thuật toán sử dụng m kiến, thực hiện N_c bước lặp xây dựng lời giải nhờ thủ tục bước ngẫu nhiên như mô tả trong mục 2.2.

3.1.1. Quy tắc chuyển trạng thái

Giả sử kiến r đã xây dựng $x_k =< u_0, \dots, i >$ là mở rộng được, nó chọn đỉnh y thuộc $J(x_k)$ để mở rộng thành $x_{k+1} =< u_0, \dots, i, y >$ với xác suất được cho bởi công thức (3.1) sau đây:

$$P(y/\tau, x_k) = \begin{cases} \frac{\tau_{i,y}^{\alpha} h_{i,y}}{\sum_{j \in J(x_k)} \tau_{i,j}^{\alpha} h_{i,j}} & với\ y \in J(x_k) \\ 0 & với\ y \notin J(x_k) \end{cases} \quad (3.1)$$

Quá trình mở rộng tiếp tục cho tới khi kiến r tìm được lời giải chấp nhận được với độ dài không quá h.

Chú ý. Quy tắc này khác một chút so với quy tắc chuyển trạng thái của thuật toán ACS và công thức 2.1, nhưng không ảnh hưởng tới các kết quả phân tích toán học về sau.

Ký hiệu $w(t)$ là lời giải tốt nhất các kiến tìm được cho tới lần lặp thứ t và $w^i(t)$ là lời giải tốt nhất trong bước lặp thứ t. Nếu $w^i(t)$ không tốt hơn $w(t-1)$ ta có $w(t) = w(t-1)$. Ta sẽ quan tâm tới các lời giải gần đúng $w(t)$ này.

3.1.2. Cập nhật mùi

Ở đây luận án xét hai quy tắc điển hình, được sử dụng phổ biến nhất hiện nay xuất phát từ ACS và MMAS. Giả sử g là một hàm thực, xác định trên S sao cho $\forall s \in S\ 0 < g(s) < \infty$ và $g(s) > g(s')$ nếu $f(s) < f(s')$ (trong bài toán TSP, $g(s)$ chính là nghịch đảo của độ dài đường đi tương ứng), khi đó ở mỗi bước lặp cường độ vết mùi sẽ thay đổi theo một trong các quy tắc sau đây:

Quy tắc ACS. Quy tắc này phỏng theo ACS, bao gồm cả cập nhật địa phương và toàn cục.

Cập nhật mùi địa phương. Nếu kiến k thăm cạnh (i, j), tức là $(i, j) \in s(k)$ thì cạnh này sẽ thay đổi mùi theo công thức sau:

$$\tau_{i,j} \leftarrow (1 - \rho)\,\tau_{i,j} + \rho\tau_1 \tag{3.2}$$

Cập nhật mùi toàn cục. Cập nhật mùi toàn cục đối với các cạnh thuộc $w(t)$:

$$\tau_{i,j} \leftarrow (1-\rho)\,\tau_{i,j} + \rho g\big(w(t)\big) \tag{3.3}$$

Quy tắc MMAS. Quy tắc này giống như trong MMAS. Sau khi kiến xây dựng xong lời giải ở bước lặp nào đó, vết mùi được thay đổi theo công thức sau:

$$\tau_{i,j} \leftarrow (1-\rho)\,\tau_{i,j} + \Delta\tau_{i,j} \tag{3.4}$$

trong đó

$$\Delta\tau_{i,j} = \begin{cases} \rho g\big(w(t)\big) & (i,j) \in w(t) \\ \max\{\tau_1 - (1-\rho)\tau_{i,j}, 0\} & (i,j) \notin w(t) \end{cases} \tag{3.5}$$

ở đây τ_1>0 là tham số.

Chú ý:

1) Công thức (3.4) trở thành công thức (2.10) trong MMAS khi lấy $\tau_1 = \tau_{min}$ và giả thiết $\tau_{max} \geq g(w(t))$
2) Các quy tắc cập nhật mùi ở trên chính là quy tắc G-best. Nếu trong các công thức (3.3) và (3.5) thay $w(t)$ bởi $w^i(t)$, thì ta nói là quy tắc I-best. Trong mục sau ta chỉ xét cho quy tắc G-best.

3.2. Phân tích toán học về xu thế vết mùi

Dưới đây đưa ra nghiên cứu về tính hội tụ của các thuật toán ACS và MMAS. Sau khi ước lượng xác suất tìm thấy một phương án ở bước lặp t, luận án sẽ khảo sát sự thay đổi của vết mùi.

3.2.1. Ước lượng xác suất tìm thấy một phương án

Mệnh đề 3.1. Các khẳng định sau là đúng:

a) Bài toán tổng quát có lời giải tối ưu.

b) Với mỗi kết quả thực nghiệm, các giá trị $f(w(t))$ hội tụ cho mỗi lần chạy khi t dần ra vô hạn.

c) Ký hiệu tập lời giải của bài toán là S^* và giá trị tối ưu là g^* ($g^* = f(s): s \in S^*$) thì với mọi cạnh $(i,j) \in E$ ta có đánh giá sau:

$$0 < \tau_{min} = min\{\tau_0, \tau_1, g(w(1))\} \leq \tau_{i,j} \leq max\{\tau_0, \tau_1, g(w(1))\} = \tau_{max} \qquad (3.6)$$

Chứng minh

Ta thấy khẳng định a) là hiển nhiên vì tập S (và X^*) là tập hữu hạn nên tồn tại giá trị tối ưu là f^*.

Khẳng định b) suy từ tính đơn điệu giảm của dãy $f(w(t))$ và dãy này bị chặn bởi g^*.

Khẳng định c) dễ dàng nhận được nhờ chứng minh quy nạp theo t với lưu ý rằng ở mỗi lần cập nhật mùi, cường độ vết mùi của các cạnh (i,j) theo quy tắc ACS có dạng: $\tau_{i,j} \leftarrow (1-\rho)\tau_{i,j} + \rho\delta_{ij}$, trong đó $\tau_{min} \leq \tau_{i,j}, \delta_{i,j} \leq \tau_{max}$, còn cường độ vết mùi các cạnh cập nhật theo MMAS suy từ các công thức (3.4) và (3.5).

Về sau, ta sẽ giả thiết $\tau_1 < g(w(t))\ \forall t$ và như vậy, $\tau_{max} = g^*$.

Định nghĩa 3.1. Với mọi i thuộc V, đại lượng $k_*(i) = min\left\{\frac{h_{i,j}}{h_{i,k}}: j,k \in V\right\}$ được gọi là hệ số lệch heuristic của đỉnh i, còn đại lượng $k_* = min\{k_*(i): i \in V\}$ được gọi là hệ số lệch heuristic của bài toán.

Với mọi $s \in S$, ta ký hiệu $p_s(t)$ là xác suất để m con kiến tìm được s ở bước lặp t. Định lý sau đây cho ta một ước lượng cận dưới của nó.

Định lý 3.1.Với mọi $s \in S$ và với mọi t, ta luôn có:

$$p_s(t) \geq p_{min} > 0, \tag{3.7}$$

trong đó p_{min} xác định bởi công thức:

$$p_{min} = 1 - \exp\left(-\frac{mk_*^h \tau_{min}^{h\alpha}}{n^h \tau_{max}^{h\alpha}}\right) \tag{3.8}$$

Chứng minh. Giả sử s được xác định bởi $x\ (\in X^*) = < u_0, \ldots, u_k > (k \leq h)$ được mở rộng từ u_0. Với mỗi bước lặp t và $\forall i < h$, từ (3.1) và (3.3) ta có đánh giá sau đối với xác suất $p(x_{i+1}/x_i)$ để $x_{i+1} = < u_0, \ldots, u_i, u_{i+1} >$ được mở rộng từ $x_i = < u_0, \ldots, u_i >$ bởi kiến r là:

$$p(x_{i+1}/x_i) \geq \frac{\tau_{min}^{\alpha} h_{u_i u_{i+1}}}{\sum_{u \in J_r(u_i)} \tau_{max}^{\alpha} h_{u_i u_{i+1}}} \geq \frac{k_*(u_i) \tau_{min}^{\alpha}}{n \tau_{max}^{\alpha}} \tag{3.9}$$

Do đó, nếu gọi $p(r, s)$ là xác suất để kiến r tìm được s ta có:

$$p(r,s) = \prod_{i=1}^{k} p(x_i/x_{i-1}) \geq \frac{k_*^k \tau_{min}^{k\alpha}}{n^k \tau_{max}^{k\alpha}} \geq \frac{k_*^h \tau_{min}^{h\alpha}}{n^h \tau_{max}^{h\alpha}} > 0$$

Như vậy, ta có ước lượng xác suất tất cả m kiến không tìm thấy trong lần lặp này như sau:

$$p(\neg s) \leq \left(1 - \frac{k_*^h \tau_{min}^{h\alpha}}{n^h \tau_{max}^{h\alpha}}\right)^m \leq \exp\left(-\frac{mk_*^h \tau_{min}^{h\alpha}}{n^h \tau_{max}^{h\alpha}}\right)$$

Suy ra : $p_s(t) \geq 1 - \exp\left(-\frac{mk_*^h \tau_{min}^{h\alpha}}{n^h \tau_{max}^{h\alpha}}\right) = p_{min}$. Định lý được chứng minh.

Chú ý. Nếu dùng quy tắc chuyển trạng thái cho bởi (2.7), dễ dàng nhận được:

$$p_{min} = (1 - q_0)\left[1 - \exp\left(-\frac{mk_*^h \tau_{min}^{h\alpha}}{n^h \tau_{max}^{h\alpha}}\right)\right]$$

Kí hiệu $P(t)$ là xác suất tìm được lời giải trong t bước lặp (hay $w(t) \in S^*$). Định lý sau là mở rộng của định lý 4.1 trong [65], đảm bảo tính hội tụ của thuật toán.

Định lý 3.2. Với mọi $\varepsilon > 0$ bé tuỳ ý, tồn tại T sao cho với mọi $t > T$ ta đều có: $P(t) > 1 - \varepsilon$.

Chứng minh. Theo mệnh đề 3.1, luôn tồn tại lời giải tối ưu $s^* \in S^*$. Từ định lý 3.1, ta có ngay ước lượng:

$$P(t) \geq 1 - \prod_{i=1}^{t}[1 - p_{s^*}(i)] = 1 - (1 - p_{min})^t = 1 - \left(\exp\left(-\frac{mk_*^h \tau_{min}^{h\alpha}}{n^h \tau_{max}^{h\alpha}}\right)\right)^t$$

$$= 1 - \exp\left(-t\frac{mk_*^h \tau_{min}^{h\alpha}}{n^h \tau_{max}^{h\alpha}}\right) \qquad (3.10)$$

Biểu thức (3.10) cho ta kết luận của định lý khi chọn T đủ lớn.

3.2.2. Đặc tính của vết mùi

Ta thấy rằng trong thực tế, ở các bước lặp t đủ lớn, khả năng xảy ra $g(w(t)) > g(w(t+1))$ (và do đó, $w(t+1) \neq w(t)$) rất bé, nên có thể từ bước lặp t_0 có các cạnh (i,j) không bao giờ thuộc vào $w(t)$ $\forall t > t_0$ hoặc luôn thuộc vào nó. Ta sẽ khảo sát đặc điểm của $\tau_{i,j}$ trong các trường hợp này.

Định lý 3.3. Giả sử cạnh (i,j) thuộc vào lời giải chấp nhận được s nào đó và tồn tại T sao cho $(i,j) \notin w(t)$ với $t \geq T$ thì các khẳng định sau đúng:

a) $\tau_{i,j}(t)$ hội tụ theo xác suất tới τ_1 nếu dùng quy tắc cập nhật mùi ACS.

b) $\tau_{i,j}(t) = \tau_1$ với mọi $t > T + \frac{\ln\frac{\tau_1}{g^*}}{\ln(1-\rho)}$ nếu dùng quy tắc cập nhật mùi MMAS.

Chứng minh

a) $\forall \varepsilon > 0$ ta cần chứng minh $\lim_{t\to\infty} P\left(\left|\tau_{i,j}(t) - \tau_1\right| > \varepsilon\right) = 0$. Công thức cập nhật mùi đia phương (3.2) có thể viết như sau:

$$\tau_{i,j} \leftarrow (1-\rho)\tau_{i,j} + \rho\tau_1 = \tau_1 + (1-\rho)(\tau_{i,j} - \tau_1)$$

Nếu từ bước lặp thứ T cho đến bước lặp t, cạnh (i,j) được cập nhật cục bộ k lần thì:

$$\tau_{i,j}(t) = \tau_1 + (1-p)^k[\tau_{i,j}(T) - \tau_1].$$

Do đó, tồn tại k_0 đủ lớn để cho với mọi $k > k_0$, ta có:

$$\left|\tau_{i,j}(t) - \tau_1\right| = \left|(1-p)^k\left[\tau_{i,j}(T) - \tau_1\right]\right| \le \varepsilon.$$

Ta sẽ chứng minh $\lim_{t\to\infty} P(k \le k_0) = 0$.

Thực vậy, nếu đặt $p_0 = \frac{k_*^h \tau_{min}^{h\alpha}}{n^h \tau_{max}^{h\alpha}}$ ta có đánh giá sau đối với xác suất $P(r;(i,j))$ để cho kiến r cập nhật cục bộ đối với cạnh (i,j) ở mỗi bước lặp: $P(r;(i,j)) \ge p_0 > 0$. Do vậy xác suất $P(r;\neg(i,j))$ của kiến r không cập nhật địa phương cạnh (i,j) ở mỗi bước lặp sẽ thoả mãn:

$$P(r;\ \neg(i,j)) \le 1 - p_0 = q_0 < 1 \qquad (3.11)$$

Đặt $t = T + q$, từ lần lặp thứ T tới $t + q$, có thể xem như có mq con kiến cập nhật cục bộ ngẫu nhiên cạnh (i,j) với xác suất không cập nhật thoả mãn (3.11) trong bất cứ điều kiện nào. Lập luận tương tự như phép thử Béc_nu_li ta có đánh giá:

$$P(k \le k_0) \le \sum_{i=1}^{k_0} C_{mq}^i q_0^{mq-i} \qquad (3.12)$$

Do vậy $\lim_{t\to\infty} P\left(\left|\tau_{i,j}(t) - \tau_1\right| > \varepsilon\right) = \lim_{t\to\infty} P(k \le k_0)$

$$= \lim_{q\to\infty} \sum_{i=1}^{k_0} C_{mq}^i q_0^{mq-i} = 0 \text{ (đpcm)}$$

b) Với $t = T + q$, ta có:

$$\tau_1 \le \tau_{i,j}(t) = \max\left\{(1-\rho)^q \tau_{i,j}(T), \tau_1\right\} \le \max\{(1-\rho)^q g^*, \tau_1\}\ \forall q \ge \frac{ln\frac{\tau_1}{g^*}}{\ln(1-\rho)} \text{ (đpcm).}$$

Định lý 3.4. Giả sử cạnh $(i,j) \in w(t)\ \forall t \ge T$ thì các khẳng định sau đúng.

a) Nếu cập nhật mùi theo ACS thì:

$$\lim_{t\to\infty}\tau_{i,j}(t) \geq \tau_1 + \rho\frac{g(w(T))-\tau_1}{1-(1-\rho)^{m+1}} \quad (3.13)$$

b) Nếu cập nhật mùi theo MMAS thì:

$$\lim_{t\to\infty}\tau_{i,j}(t) \geq g(w(T)) \quad (3.14)$$

Chứng minh

a) Ký hiệu k_u là số lần cập nhật mùi cục bộ của cạnh (i,j) trong bước lặp u, khi đó ta có:

$\tau_{i,j}(u+1) = (1-\rho)^{k_u+1}\tau_{i,j}(u) + \sum_{v=1}^{k_u+1}(1-\rho)\rho\tau_1 + (1-\rho)\rho\left[g\left(w(T)\right)-\tau_1\right]$,

tức là có một lần cập nhật toàn cục trong mỗi lần lặp. Do $g(w(t))$ là hàm đơn điệu tăng theo t, nên ta suy ra:

$\tau_{i,j}(t) \geq (1-\rho)^{q_t}\tau_{i,j}(T) + \sum_{v=1}^{q_t}(1-\rho)^v\rho\tau_1 + \sum_{v=1}^{t-T}(1-\rho)^{h_v}\rho\left[g\left(w(T)\right)-\tau_1\right]$,

trong đó $q_t = \sum_{u=T+1}^{t}(k_u+1)$ và $h_1 = 1; h_{v+1} = h_v + k_{t-v+2} + 1$ với mọi $v \geq 2$

Bởi vì $k_u \leq m$ với mọi u nên t: $\tau_{i,j}(t) \geq (1-\rho)^{q_t}\tau_{i,j}(T) + \sum_{v=1}^{q_t}(1-\rho)^v\rho\tau_1 + \sum_{v=0}^{t-T-1}(1-\rho)^{v(m+1)}\rho\left[g\left(w(T)\right)-\tau_1\right]$

Lấy giới hạn biểu thức này khi t dần ra vô hạn, với lưu ý rằng $\lim_{t\to\infty} q_t = \infty$, ta có biểu thức (3.13). Do vậy khẳng định a) đúng.

Khẳng định b) được chứng minh tương tự, khi lấy giới hạn biểu thức:

$$\tau_{i,j}(t) \geq (1-\rho)^{t-T}\tau_{i,j}(T) + \sum_{v=0}^{t-T-1}(1-\rho)^v\rho g(w(T)) \quad (3.15)$$

3.3. Thảo luận

Trong các bài toán tối ưu tổ hợp, về mặt lý thuyết, ta có thể tìm lời giải tối ưu bằng cách vét cạn, nhưng thực tế điều này không khả thi, do không gian tìm kiếm quá

rộng. Các quy tắc heuristic cho phép ta dựa trên "*các kinh nghiệm"* có được để tìm nhanh các lời giải đủ tốt trong phạm vi tìm kiếm hẹp và chấp nhận loại bỏ những lời giải tốt hơn. Sự kết hợp học tăng cường thông qua thông tin về cường độ vết mùi cho phép ta từng bước thu hẹp miền tìm kiếm, mà vẫn không loại bỏ các lời giải tốt. Do đó, nâng cao chất lượng thuật toán.

Ta thấy chất lượng của thông tin heuristic tốt sẽ nâng cao hiệu quả thuật toán. Tuy nhiên, các quy tắc này không phải luôn có được và rất khó có thể can thiệp để thay đổi chất lượng. Do vậy, ta sẽ quan tâm tới cách cập nhật mùi để nâng cao chất lượng thuật toán. Dưới đây, sau khi nhận xét chung về đặc tính khai thác và khám phá của các thuật toán, luận án sẽ đưa ra nhận xét về các quy tắc cập nhật mùi đã nêu ở trên và đưa ra một số đề xuất.

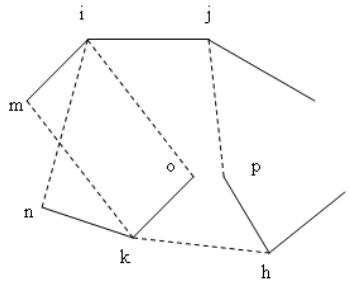
3.3.1. Tính khai thác và khám phá

Tính khai thác là việc tập trung tìm kiếm lời giải xung quanh phạm vi của các cạnh thuộc các lời giải tốt nhất đã được biết cho tới thời điểm đang xét, còn tính khám phá là tìm kiếm ở các phạm vi khác. Trong cách cập nhật mùi G-best, ta đã biết $w(t)$ nên việc tìm kiếm quanh nó sẽ hạn chế nhiều tính khám phá, còn khi cập nhật theo I-best sẽ mở rộng miền này hơn. Vì vậy, trong thực nghiệm cập nhật theo I-best sẽ cho kết quả tốt hơn G-best.

Trong các bài toán tối ưu tổ hợp, xác suất một phương án cho trước được kiến tìm ra ở mỗi lần lặp là rất bé. Vì vậy, có thể sau một số bước lặp, cường độ vết mùi trên mỗi cạnh không thuộc $w(t)$ sẽ bé, do đó làm giảm khả năng khám phá, mặc dù chúng có thể có triển vọng thuộc lời giải tốt. Với bài toán TSP, ta có mệnh đề sau.

Mệnh đề 3.2. Trong bài toán TSP với đồ thị không có hướng, mỗi chu trình Hamilton (đường liền) qua cạnh (i, j) và không qua cạnh (k, h) có thể đổi nhiều nhất 7 cạnh để có được chu trình đi qua cạnh (k, h) mà không qua (i, j).

Chứng minh. Kết luận của mệnh đề dễ dàng nhận được từ hình 3.1. Trong đó chu trình gồm các cạnh (đường liền) liền qua (i,j) và nhờ đổi đi 5 cạnh tương ứng $\{(m,i),(n,k),(i,j),(k,o),(p,h)\}$ thành $\{(m,k),(n,i),(i,o),(k,h),(p,j)\}$ (đường đứt đoạn) sẽ được hai chu trình độc lập, và đổi tiếp 2 cạnh để đưa 2 chu trình thành một chu trình đi qua cạnh (k,h) mà không qua (i,j).



Hình 3.1: Hai chu trình khác nhau 7 cạnh, đường liền qua cạnh (i,j) và đường đứt đoạn qua cạnh (k,h)

Các điểm hạn chế của ACO

1) Mệnh đề trên cho thấy khi thuật toán mới bắt đầu, các vết mùi được khởi tạo như nhau, một cạnh (k,h) "tốt hơn" cạnh (i,j), do nó thuộc chu trình dài hơn có thể bị đảo ngược một cách rất ngẫu nhiên. Khi một cạnh do ngẫu nhiên không được cập nhật mùi, sau một số bước cường độ mùi của nó nhanh chóng bị giảm xuống và do vậy khó được kiến chọn ở bước sau đó, mặc dù "chất lượng" của nó chưa chắc đã là "xấu".

2) Nếu khởi tạo mùi như nhau và không dùng thông tin heuristic, xác suất của mỗi cạnh được kiến đã cho sử dụng trong lần lặp đầu sẽ là $\frac{2}{n-1}$. Xác suất này rất bé khi n lớn. Như vậy, tùy theo từng loại bài toán mà tỷ lệ giữa τ_0 và τ_1 sẽ rất có ý nghĩa cho cân bằng giữa tính khám phá và tính khai thác của thuật toán.

3) Các lượng mùi cập nhật theo các công thức từ (3.2) đến (3.5) phụ thuộc vào giá trị hàm mục tiêu của lời giải được kiến xây dựng được trong các bước lặp. Việc xác định các giá trị τ_0 và τ_1 hay τ_{min} và τ_{max} cũng phụ thuộc vào tương quan với các giá

trị chưa được xác định trước này của từng bài toán, khi đó thuật toán mới tốt được. Tuy nhiên, điều này rất khó thực hiện.

Bây giờ ta sẽ bình luận cụ thể hơn về từng thuật toán đã nêu.

3.3.2. Các thuật toán cập nhật mùi theo quy tắc ACS

Như đã nói ở mục 3.2.2, cách cập nhật mùi toàn cục của ACS [30] thực hiện bay hơi đối với các cạnh không được kiến chọn. Vì vậy, không đảm bảo cường độ vết mùi thỏa mãn điều kiện thuộc khoảng $[\tau_{min}, \tau_{max}]$. Vết mùi của những cạnh, không được kiến sử dụng và không thuộc đường đi tốt, sẽ nhanh chóng dần về 0, làm cho các con kiến sau sẽ có bỏ qua các cạnh này. Tuy vậy, như đã chỉ ra ở trên, cạnh này vẫn có thể là cạnh tham gia vào lời giải "tốt", nhưng bị loại do rủi ro. Hiện tượng này làm giảm tính khám phá của ACS, vì thế hiệu quả của nó kém MMAS (xem [66])

Quy tắc cập nhật mùi toàn cục trong [31] (được giới thiệu ở chương trước) đã khắc phục được hạn chế này nên đến nay nó được thay cho cách cập nhật ở [30]. Tuy vậy, việc cập nhật địa phương chưa cho thấy rõ ý nghĩa của học tăng cường.

3.3.3. Các thuật toán cập nhật mùi theo quy tắc MMAS

Theo quy tắc này, việc tìm kiếm chỉ tập trung quanh lời giải tốt nhất, còn các cạnh không thuộc lời giải này sẽ có cường độ vết mùi nhanh chóng tụt về τ_{min} theo đánh giá b) của định lý 3.3. Vì vậy, khi τ_{min} nhỏ hơn nhiều so với τ_{max}, tính khám phá sẽ kém, còn nếu chọn τ_{min} gần với τ_{max} thì thuật toán chủ yếu là tìm kiếm ngẫu nhiên dựa theo thông tin heuristic.

3.4. Đề xuất các phương pháp cập nhật mùi mới

Dựa trên các phân tích trên, luận án đề xuất quy tắc cải tiến của ACS và MMAS như sau:

a) Phương pháp cập nhật mùi đa mức: MLAS (Multi-level Ant System)

Dựa vào nhận xét ở mục trước, thay cho việc bay hơi vết mùi ở các thành phần không thuộc các lời giải của mỗi con kiến trong mỗi lần cập nhật mùi ở mỗi bước lặp, ta cho τ_1 và τ_{max} tăng dần. Độ lệch giữa τ_1 và τ_{max} cho phép ta điều khiển tính hội tụ và khám phá. Nếu thấy lời giải tốt ít thay đổi thì cho τ_1 gần τ_{max} để tăng tính khám phá và ngược lại, cho τ_1 dịch xa τ_{max} để cho lời giải tập trung tìm kiếm quanh lời giải tốt nhất tìm được.

Quy tắc này đã thử nghiệm cho các bài toán TSP và JSS, cho kết quả khả quan so với MMAS. Tuy nhiên, việc điều khiển độ lệch giữa τ_1 và τ_{max} rất khó áp dụng cho các bài toán cụ thể, nên chúng tôi thay bởi phương pháp 3-LAS sẽ trình bày ở phần c) dưới đây.

b) Phương pháp Max-Min trơn: SMMAS (Smoothed Max Min Ant System)

Dựa vào nhận xét ở mục trên, ta thấy không nên giảm vết mùi ở các cạnh không thuộc lời giải tốt quá nhanh như trong quy tắc MMAS, mà nên dùng quy tắc Max-Min trơn như sau: $\tau_{i,j} \leftarrow (1-\rho)\tau_{i,j} + \Delta\tau_{i,j}$ với

$$\Delta\tau_{i,j} = \begin{cases} \rho\tau_{min} \; nếu \; (i,j) \notin w(t) \\ \rho\tau_{max} \; nếu \; (i,j) \in w(t) \end{cases} \tag{3.16}$$

Khi cài đặt, lấy $\tau_0 = \tau_{max}$.

c) Phương pháp 3-LAS (Three-level Ant System)

Đối với các bài toán sử dụng thông tin heuristic, ảnh hưởng nhiều tới chất lượng tìm kiếm lời giải, chẳng hạn như bài toán TSP, phương pháp 3-LAS tương tự ACS, nhưng dễ dùng hơn và hiệu quả tốt hơn. Phương pháp này sử dụng thêm tham số τ_{mid} thuộc khoảng (τ_{min}, τ_{max}) và cập nhật mùi tương tự SMMAS cho các cạnh có kiến sử dụng hoặc thuộc $w(t)$. Cụ thể là:

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

$$\Delta\tau_{i,j} = \begin{cases} \rho\tau_{max} & nếu\ (i,j) \in w(t) \\ \rho\tau_{mid} & nếu\ \forall(i,j) \overline{\in} w(t) \text{ và có kiến đi qua} \\ \rho\tau_{min} & \text{cho các cạnh còn lại} \end{cases} \quad (3.17)$$

3.5. Nhận xét về các thuật toán mới

Trong ba phương pháp cập nhật mùi ở trên, hai phương pháp SMMAS và 3-LAS đơn giản và dễ sử dụng hơn, nên luận án sẽ nêu ra các ưu điểm của hai thuật toán này khi sử dụng và nhận xét về tính bất biến của chúng.

3.5.1. Ưu điểm khi sử dụng SMMAS và 3-LAS

Ta thấy thuật toán SMMAS và 3-LAS có một số ưu điểm sau so với ACS và MMAS.

1) Với ACS và MMAS, để xác định τ_0 hay τ_{min} và τ_{max} người ta cần tìm một lời giải theo phương pháp heuristic và dựa vào giá trị hàm mục tiêu của nó. Vì giá trị hàm mục tiêu này nhận được ngẫu nhiên, nên khó xác định tốt tham số cho học tăng cường. Quy tắc cập nhật mới cho phép ta xác định các tham số này đơn giản và hợp lý hơn, cụ thể: trong SMMAS và 3-LAS ta không cần xác định chính xác giá trị τ_{min} , τ_{max} mà chỉ cần xác định tỉ lệ giữa τ_{min} , τ_{max}. Trong thực nghiệm, luận án luôn thiết đặt $\tau_{max} = 1.0$ và xác định τ_{min} qua tỉ lệ giữa τ_{min} , τ_{max}. Cần nhấn mạnh rằng, việc chỉ cần lựa chọn tỉ lệ giữa τ_{min} , τ_{max} đơn giản và mất ít thời gian thực nghiệm hơn rất nhiều so với việc lựa chọn cụ thể hai tham số τ_{min} , τ_{max}.

2) Việc thêm mùi cho các cạnh thuộc lời giải tốt ở mỗi bước lặp trong thuật toán ACS và MMAS, ta phải xây dựng hàm để tính lượng mùi được thêm dựa trên chất lượng lời giải do kiến xây dựng được. Ví dụ, trong bài toán TSP, ACS và MMAS sử dụng hàm nghịch đảo độ dài đường đi được kiến xác định. Điều này cũng là một trong

những khó khăn khi áp dụng ACS (hoặc MMAS) đối với một bài toán mới. Tuy nhiên, trong SMMAS và 3-LAS không cần phải xây dựng hàm này.

3) Dễ dàng kiểm tra được các thuật toán này có cùng độ phức tạp như MMAS và ACS, nhưng ít phép toán hơn MMAS vì không phải tính hàm mục tiêu ở lượng mùi cập nhật và không phải so sánh để giới hạn vết mùi trong khoảng τ_{min} , τ_{max}. Theo cách cập nhật của SMMAS và 3-LAS, vết mùi luôn trong khoảng τ_{min} , τ_{max}.

3.5.2. Tính bất biến

Một số tác giả đã xét tính bất biến của các quy tắc cập nhật mùi khi hàm mục tiêu được biến đổi tuyến tính đơn điệu [8,70]. Để khảo sát tính bất biến, ta cần khái niệm về các thể hiện của bài toán TƯTH và giả thiết về tính lặp của máy tạo số giả ngẫu nhiên.

Định nghĩa 3.2. (Thể hiện của bài toán)

Xét hai bài toán TƯTH (S, f, Ω) và (S, f', Ω). Ta sẽ gọi chúng là hai thể hiện I và I' tương ứng của một bài toán, nếu $f'(s) = g(f(s))$ với mọi s thuộc S trong đó g là hàm đơn điệu tăng chặt.

Thông thường ta sẽ xét g thuộc một nhóm các hàm G nào đó.

Định nghĩa 3.3. (Giả thiết về tính lặp của máy tạo số giả ngẫu nhiên)

Khi chạy một thuật toán tìm kiếm ngẫu nhiên cho hai thể hiện của một bài toán, các quyết định dựa trên các thí nghiệm ngẫu nhiên nhờ một máy tạo số giả ngẫu nhiên. Ta giả thiết các số này được tạo ra cùng một phương pháp (chẳng hạn cùng một giống:seed), như vậy dãy số ngẫu nhiên tạo ra khi giải hai thể hiện là như nhau và ta gọi nó là máy phát lặp.

Bây giờ ta định nghĩa tính bất biến của thuật toán.

Định nghĩa 3.4. (Tính bất biến của thuật toán)

Ta nói thuật toán A bất biến trên nhóm biến đổi đơn điệu G đối với bài toán (S, f, Ω) nếu khi sử dụng nó nhờ một máy phát lặp để giải hai thể hiện của bài toán vẫn cho ta cùng một dãy lời giải và vết mùi.

Birattari [8] và Zang [70] đã xét tính bất biến của một số thuật toán nhờ biến đổi tuyến tính đơn điệu tăng: $G = \{g(s) = af(s) + b\}$, trong đó $a > 0$.

Ta dễ dàng kiểm tra được các thuật toán SMMAS và 3-LAS bất biến đối với nhóm biến đổi đơn điệu tăng của bài toán (S, f, Ω). Định lý sau là đúng.

Định lý 3.5. Giả sử I và I' là hai thể hiện của một bài toán TƯTH tùy ý. Khi giải bằng một trong hai thuật toán SMMAS hoặc 3-LAS với cùng số lần lặp nhờ dùng một máy phát lặp sẽ cho kết quả cùng một dãy lời giải và các vectơ vết mùi.

Kết luận của định lý là hiển nhiên vì quyết định và lượng mùi thay đổi trong mỗi lần lặp sẽ như nhau ở mỗi lần lặp.

3.6. Kết quả thực nghiệm cho hai bài toán TSP và UBQP

Trong phần này luận án giới thiệu kết quả thực nghiệm đối với các thuật toán mới cho bài toán TSP và so sánh với MMAS. Ngoài ra, luận án cũng so sánh SMMAS với MMAS cho bài toán UBQP với lý do sẽ nêu ở mục 3.6.2.

3.6.1. Thực nghiệm trên bài toán TSP

Như đã nói trước đây, bài toán TSP được xem là bài toán chuẩn để thử nghiệm các thuật toán giải các bài toán TƯTH mới. Trong [20], đã giới thiệu một thuật toán loại MLAS cho bài toán TSP và cho thấy phương pháp này tốt hơn MMAS và ACS. Trong [21], kết quả thực nghiệm của một thuật toán SMMAS và biến thể khác của

MLAS đối với bài toán JSS cho thấy thuật toán với các quy tắc cập nhật mùi này tốt hơn ACS và MMAS.

Dưới đây trình bày kết quả thực nghiệm so sánh ba phương pháp MMAS, SMMAS và MLAS với 5 bộ test eil51, kroA100, d198, lin318 và rat783 với số đỉnh tương ứng là 51, 100, 198, 318 và 783 đỉnh. Đây là các bộ test chuẩn của bài toán TSP được Dorigo và Stützle dùng để thực nghiệm và công bố [30,66] (các bộ test lấy từ địa chỉ [77]).

Kết quả của MMAS trong bảng 3.1 là kết quả chạy chương trình của chính tác giả Stützle, lấy từ địa chỉ [76]. Kết quả chạy theo chương trình này tốt hơn kết quả đã công bố trong bài báo [66]. MMAS [66] khi công bố kết quả thực nghiệm cho thấy tốt hơn ACS, sau đó không thấy có tài liệu nào cho rằng ACS tốt hơn. Hơn nữa MMAS đơn giản hơn nên nhiều người ưa dùng. Phần mềm MMAS này là phiên bản mới nhất của Thomas Stutzle [76] được biết, nên được dùng để so sánh thuật toán mới.

Trong MLAS, SMMAS và 3-LAS, các tham số được thiết đặt như sau: $m = \frac{N}{2}$;$\alpha = 1$; $\beta = 2$; $\rho = 0.05$. Ngoài ra, trong SMMAS và 3-LAS tỉ lệ $\frac{\tau_{max}}{\tau_{min}}$ đặt bằng $N.k$, với $k = \begin{cases} \frac{N+50}{100} & nếu\ N \geq 50 \\ 1 & nếu\ N < 50 \end{cases}$, tỉ lệ $\frac{\tau_{max}}{\tau_{mid}} = k$. Việc điều khiển τ_{min}, τ_{max} trong MLAS thực hiện như trong [20]. Thực nghiệm thực hiện với số lần chạy và số lời giải như Stützle đã làm trong [66]. Cụ thể, với các test eil51, kroA100 và d198, tất cả phương pháp cho chạy 25 lần, mỗi lần cùng chạy với số lời giải $S = 10000 \times N$ (N là số đỉnh), hai test lin318 và rat783 chạy 10 lần, mỗi lần cùng chạy với số lời giải $S = 10^6$, sau đó so sánh kết quả trung bình và kết quả tốt nhất của các lần chạy.

Kết quả của các phương pháp với từng bộ test nằm trong ô giao giữa cột và dòng tương ứng, trong đó số ở trên biểu thị kết quả trung bình và số trong ngoặc biểu thị độ lệch của kết quả trung bình với kết quả tối ưu của test đó, hai số ở dưới lần lượt

là kết quả tốt nhất và kết quả tồi nhất trong các lần chạy, số ở dòng thứ ba là độ lệch chuẩn. Trong phương pháp ACO, thực nghiệm so sánh các thuật toán được thực hiện trên nhiều bộ dữ liệu và thống kê theo đa số dựa trên kết quả trung bình. Kết quả trung bình phản ánh chất lượng của thuật toán, còn các kết quả tốt nhất và tồi nhất để tham khảo về tính khám phá của nó, độ lệch chuẩn để tham khảo tính ổn định của thuật toán. Trong các bảng kết quả dưới đây, kết quả được tô đậm là kết quả tốt nhất trong các phương pháp.

Bảng 3.1: Kết quả thực nghiệm so sánh bốn phương pháp MMAS, SMMAS, MLAS và 3-LAS.

test	MMAS		SMMAS		MLAS		3-LAS	
eil51	426.44(0.10%)		426.08(0.02%)		**426.00(0.00%)**		426.2(0.05%)	
	426	428	**426**	427	**426**	426	**426**	428
	0.77		0.28		0.00		0.50	
kroA100	21304.40(0.11%)		21305.76(0.11%)		21289.70(0.04%)		**21287.96(0.03%)**	
	21282	21378	**21282**	21379	**21282**	21319	**21282**	21379
	42.86		41.92		10.17		19.41	
d198	**15950.96(1.08%)**		15990.44(1.33%)		15970.60 (1.21%)		15952.80(1.1%)	
	15875	16034	**15828**	16116	15902	16040	15854	16006
	31.42		44.05		27.24		25.30	
lin318	43106.30(2.56%)		42445.80(0.99%)		**42438.10(0.97%)**		42441.9(0.98%)	
	42859	43376	42201	42873	42257	42639	**42175**	42918
	288.50		264.86		201.33		276.95	
rat783	8951.5(1.65%)		8938.80(1.51%)		8933.00(1.44%)		**8924.50(1.35%)**	
	8920	8986	**8868**	8989	8904	8959	8873	8984
	34.78		60.59		26.56		45.84	

Nhận xét:

1) *So sánh kết quả trung bình*. Trong 5 test được thực nghiệm, 3-LAS và MLAS cho kết quả trung bình tốt nhất và cả ba thuật toán SMMAS, MLAS và 3-LAS đều tốt hơn MMAS (trừ trường hợp d198).

2) *Kết quả tốt nhất*. Trong 5 test được thực nghiệm, có 2 test với số đỉnh nhỏ (eil51 - 51 đỉnh, kroA100 - 100 đỉnh) cả bốn thuật toán đều tìm được lời giải tối ưu, các test sau thì SMMAS, MLAS, 3-LAS đều cho kết quả tốt hơn MMAS.

3) Hầu như SMMAS đều cho kết quả tốt nhất trong các phương pháp nhưng kết quả trung bình các lần chạy thì kém MLAS và 3-LAS. Điều này được giải thích như sau: SMMAS không có cơ chế điều chỉnh khai thác và khám phá linh hoạt như MLAS và 3-LAS. Để khắc phục nhược điểm này, luận án sử dụng khởi tạo lại vết mùi trong SMMAS như trong MMAS. Kết quả thực nghiệm của SMMAS khi có sử dụng khởi tạo lại vết mùi trong bảng 3.2.

Bảng 3.2: Kết quả thực nghiệm so sánh các phương pháp MMAS, SMMAS, SMMAS^{RS} (SMMAS có khởi tạo lại vết mùi), MLAS và 3-LAS.

test	MMAS		SMMAS		SMMAS^{RS}		MLAS		3-LAS	
eil51	426.44(0.10%)		426.08(0.02%)		**426.00(0.00%)**		**426.00(0.00%)**		426.2(0.05%)	
	426	428	**426**	427	**426**	426	**426**	426	**426**	428
	0.77		0.28		0.00		0.00		0.50	
kroA100	21304.40(0.11%)		21305.76(0.11%)		**21282.96(0.00%)**		21289.70(0.04%)		21287.96(0.03%)	
	21282	21378	**21282**	21379	**21282**	21296	**21282**	21319	**21282**	21379
	42.86		41.92		3.37		10.17		19.41	
d198	**15950.96(1.08%)**		15990.44(1.33%)		15966.88(1.18%)		15970.60 (1.21%)		15952.80(1.1%)	
	15875	16034	**15828**	16116	**15828**	16045	15902	16040	15854	16006
	31.42		44.05		37.41		27.24		25.30	
lin318	43106.30(2.56%)		42445.80(0.99%)		42442.50(0.98%)		**42438.10(0.97%)**		42441.9(0.98%)	
	42859	43376	42201	42873	42201	42886	42257	42639	**42175**	42918
	288.50		264.86		267.22		201.33		276.95	
rat783	8951.5(1.65%)		8938.80(1.51%)		8930.90(1.42%)		8933.00(1.44%)		**8924.50(1.35%)**	
	8920	8986	**8868**	8989	8878	8963	8904	8959	8873	8984
	34.78		60.59		43.56		26.56		45.84	

Kết quả thực nghiệm cho thấy, SMMAS^{RS} điều chỉnh khai thác và khám phá bằng cơ chế khởi tạo lại vết mùi khi tắc nghẽn (giống MMAS) đã đạt hiệu quả tốt hơn.

Vì vậy, trong các thuật toán sử dụng cách cập nhật mùi SMMAS về sau đều sử dụng cơ chế khởi tạo lại vết mùi. Như vậy, cách cập nhật mùi SMMAS đơn giản, dễ sử dụng và tốt như MLAS và 3-LAS. Khi ứng dụng phương pháp ACO, ta thường phải xác định nhiều tham số và các tham số ảnh hưởng nhiều đến hiệu quả thuật toán nên SMMAS là thích hợp nhất khi giải một bài toán mới. Trong phần sau, luận án tiếp tục khảo sát tính hiệu quả của SMMAS trên bài toán quy hoạch toàn phương nhị phân không ràng buộc.

3.6.2. Thực nghiệm trên bài toán quy hoạch toàn phương nhị phân không ràng buộc

Trở lại với bài toán qui hoạch toàn phương nhị phân không ràng buộc:

Cho ma trận $Q = (q_{ij})$ là ma trận đối xứng kích thước $n \times n$. Tìm vector x là vector nhị phân gồm n thành phần $x = (x_1, x_2, .., x_n)$, trong đó $x_i = 0$ hoặc 1 sao cho:

$f(x) = x^t Qx = \sum_{i=1}^{n} \sum_{j=1}^{n} q_{ij} x_i x_j$ đạt giá trị lớn nhất.

Lý do chọn bài toán thực nghiệm

Ngoài lý do bài toán UBQP có nhiều ứng dụng, còn có những lý do luận án lựa chọn bài toán UBQP như sau:

- Trong [10], tác giả Blum và Dorigo đã đề xuất thuật toán MMAS_HCF (*Hyper-Cube Framework - HCP*) giải bài toán UBQP. Kết quả thực nghiệm trong [10] cho thấy MMAS_HCF tốt hơn các phương pháp GLS (*Genetic Local Search*), STS (*Simple Tabu Search*) và SSA (*Simple Simulated Annealing*).
- MMAS_HCF giải bài toán UBQP không sử dụng dụng thông tin heuristic.
- Vết mùi được lưu trên đỉnh chứ không theo cách thức truyền thống (vết mùi lưu trên cạnh).

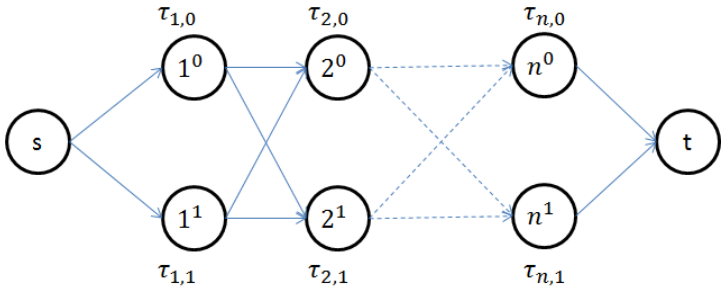
Luận án tiến hành áp dụng SMMAS theo cùng đồ thị cấu trúc, cách lưu trữ mùi và không sử dụng dụng thông tin heuristic như trong [10] đã được Blum và Dorigo dùng giải bài toán UBQP để thấy hiệu quả của cách cập nhật mùi do luận án đề xuất.

Phương pháp tối ưu đàn kiến giải bài toán UBQP

1) Đồ thị cấu trúc và vết mùi

Đồ thị cấu trúc gồm $2 \times n + 2$ đỉnh, trong đó có đỉnh s (xuất phát), t (kết thúc) và n đỉnh i^0 và n đỉnh $i^1 (i = 1,2,..,n)$. Việc xây dựng vectơ nhị phân $x = (x_1, x_2, .., x_n)$ tương ứng với việc kiến xây dựng hành trình từ s đến t. Nếu ở bước thứ i kiến chọn đỉnh i^0 thì $x_i = 0$ còn kiến chọn đỉnh i^1 thì $x_i = 1$.

Vết mùi $\tau_{i,v}$ được gán trên các đỉnh i^v $(i = 1,2,..,n; v = 0,1)$ thể hiện sự ưu tiên lựa chọn giá trị v của x_i.



Hình 3.2: Đồ thị cấu trúc giải bài toán UBQP

2) Xây dựng lời giải

Xuất phát từ đỉnh s, kiến di chuyển qua n bước rồi kết thúc tại đỉnh t. Tại bước thứ i, kiến lựa chọn đỉnh i^0 hoặc i^1 theo xác suất:

$$p_{i,v}^k = \frac{\tau_{i,v}}{\tau_{i,0} + \tau_{i,1}} \quad (3.18)$$

Nếu trong hành trình, ở bước thứ i kiến lựa chọn đỉnh i^0 thì thành phần thứ i của vector x là $x_i = 0$, còn nếu ở bước thứ i kiến lựa chọn đỉnh i^1 thì thành phần thứ i của vector x là $x_i = 1$.

3) Cập nhật mùi theo SMMAS

Vết mùi sẽ được cập nhật theo quy tắc:

$$\tau_{i,v} \leftarrow (1-\rho)\tau_{i,v} + \Delta\tau_{i,v} \tag{3.19}$$

$$\text{với } \Delta\tau_{i,v} = \begin{cases} \rho\tau_{min} \text{ nếu } (i,v) \notin w(t) \\ \rho\tau_{max} \text{ nếu } (i,v) \in w(t) \end{cases} \tag{3.20}$$

Kết quả thực nghiệm

Luận án tiến hành thực nghiệm trên bài toán UBQP để so sánh SMMAS với MMAS_HCF trong hai trường hợp có dùng tìm kiếm cục bộ và không dùng tìm kiếm cục bộ. Luận án tiến hành thực nghiệm cho tất cả các bộ test chuẩn của bài toán UBQP [5]. Mỗi thuật toán đều được chạy 20 lần độc lập và so sánh kết quả tốt nhất và kết quả trung bình của 20 lần chạy.

1) Kết quả thực nghiệm không sử dụng tìm kiếm cục bộ

Với bộ dữ liệu nhỏ (n=200): Trong 5 test được thực nghiệm, cả 5 test giá trị trung bình của SMMAS đều tốt hơn MMAS_HCF. SMMAS tìm được kết quả tối ưu của cả 5 test, còn MMAS_HCF chỉ tìm được 3 test. (Bảng 3.3)

Bộ dữ liệu trung bình (n=500): Trong bộ dữ liệu này SMMAS tốt hơn hẳn MMAS_HCF, trong 10 test được thực nghiệm cả 10 test giá trị trung bình của SMMAS đều tốt hơn MMAS_HCF và 9/10 test có kết quả tốt hơn kết quả tốt nhất của MMAS_HCF. (Bảng 3.4)

Bảng 3.3: So sánh SMMAS với MMAS_HCF với bộ dữ liệu n=200 (không dùng tìm kiếm cục bộ)

thời gian chạy 100 giây (1)		best (2)	%best (3)	avg-best (4)	%avg (5)	std (6)	suc(/20) (7)
200-1e	SMMAS	**16464**	0.00	**16437.80**	0.16	14.60	3
16464	MMAS_HCF	**16464**	0.00	16416.10	0.29	26.35	2
200-2e	SMMAS	**23395**	0.00	**23368.45**	0.11	21.62	1
23395	MMAS_HCF	23381	0.06	23315.30	0.34	83.25	0
200-3e	SMMAS	**25243**	0.00	**25207.75**	0.14	94.91	14
25243	MMAS_HCF	**25243**	0.00	24925.35	1.26	336.30	6
200-4e	SMMAS	**35594**	0.00	**35572.50**	0.06	15.32	2
35594	MMAS_HCF	**35594**	0.00	35511.50	0.23	81.36	3
200-5e	SMMAS	**35154**	0.00	**35058.35**	0.27	51.41	1
35154	MMAS_HCF	35064	0.26	34823.45	0.94	217.09	0

Ghi chú: Cột 1 là thông tin về test, tên và kết quả tối ưu của test đó (số ở dưới tên test), tên phương pháp giải bài toán UBQP. Cột 2, best là kết quả tốt nhất trong 20 lần chạy. Cột 3, %best là độ lệch kết quả tốt nhất với kết quả tối ưu. Cột 4, avg-best là kết quả trung bình của 20 lần chạy. Cột 5, %avg-best là độ lệch kết quả trung bình với kết quả tối ưu. Cột 6, std là độ lệch chuẩn. Cột 7, suc(/20) là số lần chạy cho kết quả tối ưu.

Bảng 3.4: So sánh SMMAS với MMAS_HCF với bộ dữ liệu *n*=500 (không dùng tìm kiếm cục bộ)

thời gian chạy 500 giây (1)		best (2)	%best (3)	avg-best (4)	%avg (5)	std (6)	suc(/20) (7)
500-1	SMMAS	**115501**	0.93	**114499.75**	1.79	516.61	0
116586	MMAS_HCF	114802	1.53	113593.05	2.57	968.93	0
500-2	SMMAS	**127965**	0.29	**127525.65**	0.63	406.66	0
128339	MMAS_HCF	127686	0.51	126581.55	1.37	895.32	0
500-3	SMMAS	**130382**	0.33	**129683.65**	0.86	374.86	0
130812	MMAS_HCF	130207	0.46	128838.60	1.51	849.21	0
500-4	SMMAS	**129651**	0.34	**128972.50**	0.86	397.70	0
130097	MMAS_HCF	129513	0.45	128322.60	1.36	782.87	0
500-5	SMMAS	**124847**	0.51	**124050.80**	1.14	451.85	0
125487	MMAS_HCF	124477	0.80	123110.20	1.89	850.45	0
500-6	SMMAS	**120119**	1.36	**119460.00**	1.90	583.41	0
121772	MMAS_HCF	119945	1.50	118787.05	2.45	899.41	0
500-7	SMMAS	121174	0.84	**120272.20**	1.58	442.30	0
122201	MMAS_HCF	**121241**	0.79	119629.50	2.10	915.37	0
500-8	SMMAS	**122892**	0.54	**121995.25**	1.27	368.53	0
123559	MMAS_HCF	122086	1.19	121202.05	1.91	654.56	0
500-9	SMMAS	**120134**	0.55	**118934.50**	1.54	596.42	0
120798	MMAS_HCF	118752	1.69	117526.80	2.71	853.72	0
500-10	SMMAS	**129861**	0.58	**129357.80**	0.97	294.51	0
130619	MMAS_HCF	129840	0.60	128709.60	1.46	747.64	0

2) Kết quả thực nghiệm có sử dụng tìm kiếm cục bộ

Việc thực nghiệm các thuật toán không dùng tìm kiếm cục bộ nhằm mục tiêu so sánh tính hiệu quả của cách cập nhật mùi mới. Ở đây, luận án thực nghiệm so sánh khi kết hợp với tìm kiếm cục bộ, để khảo sát khả năng tích hợp với tìm kiếm cục bộ của thuật toán mới. Thuật toán tìm kiếm cục bộ được sử như trong [10] đã được Blum và Dorigo dùng.

Bộ dữ liệu trung bình (n=500 và n=1000): Khi áp có sử dụng tìm kiếm cục bộ cả SMMAS và MMAS_HCF đều cho kết quả tốt hơn hẳn so với không sử dụng tìm kiếm cục bộ. Cả SMMAS và MMAS_HCF đều tìm được tối ưu của cả 10 test (n=500), 10 test (n=1000). So sánh theo kết quả trung bình, SMMAS tốt hơn MMAS_HCF. Có 3/10 test của bộ n=500 và tốt hơn 7/10 test của bộ n=1000 (Bảng 3.5 và Bảng 3.6).

Bảng 3.5: So sánh SMMAS với MMAS_HCF với bộ dữ liệu n=500 (có sử dụng tìm kiếm cục bộ)

thời gian chạy 100 giây (1)		best (2)	%best (3)	avg-best (4)	%avg (5)	std (6)	suc(/20) (7)
500-1	SMMAS	**116586**	0.000	**116586.00**	0.000	0.00	20
116586	MMAS_HCF	**116586**	0.000	**116586.00**	0.000	0.00	20
500-2	SMMAS	**128339**	0.000	**128339.00**	0.000	0.00	20
128339	MMAS_HCF	**128339**	0.000	**128339.00**	0.000	0.00	20
500-3	SMMAS	**130812**	0.000	**130812.00**	0.000	0.00	20
130812	MMAS_HCF	**130812**	0.000	**130812.00**	0.000	0.00	20
500-4	SMMAS	**130097**	0.000	**130096.00**	0.001	4.47	19
130097	MMAS_HCF	**130097**	0.000	130084.00	0.010	9.79	7
500-5	SMMAS	**125487**	0.000	**125487.00**	0.000	0.00	20
125487	MMAS_HCF	**125487**	0.000	**125487.00**	0.000	0.00	20
500-6	SMMAS	**121772**	0.000	**121766.70**	0.004	16.31	18
121772	MMAS_HCF	**121772**	0.000	121761.40	0.009	21.75	16
500-7	SMMAS	**122201**	0.000	**122201.00**	0.000	0.00	20
122201	MMAS_HCF	**122201**	0.000	**122201.00**	0.000	0.00	20
500-8	SMMAS	**123559**	0.000	**123559.00**	0.000	0.00	20
123559	MMAS_HCF	**123559**	0.000	123554.45	0.004	11.14	17
500-9	SMMAS	**120798**	0.000	**120798.00**	0.000	0.00	20
120798	MMAS_HCF	**120798**	0.000	**120798.00**	0.000	0.00	20
500-10	SMMAS	**130619**	0.000	**130619.00**	0.000	0.00	20
130619	MMAS_HCF	**130619**	0.000	**130619.00**	0.000	0.00	20

Bảng 3.6: So sánh SMMAS với MMAS_HCF với bộ dữ liệu n=1000 (có sử dụng tìm kiếm cục bộ)

thời gian chạy 500 giây (1)		best (2)	%best (3)	avg-best (4)	%avg (5)	std (6)	suc(/20) (7)
1000-1	SMMAS	**371438**	0.000	**371422.50**	0.004	47.71	18
371438	MMAS_HCF	**371438**	0.000	371418.20	0.005	53.26	16
1000-2	SMMAS	**354932**	0.000	**354932.00**	0.000	0.00	20
354932	MMAS_HCF	**354932**	0.000	**354932.00**	0.000	0.00	20
1000-3	SMMAS	**371236**	0.000	**371236.00**	0.000	0.00	20
371236	MMAS_HCF	**371236**	0.000	**371236.00**	0.000	0.00	20
1000-4	SMMAS	**370675**	0.000	**370675.00**	0.000	0.00	20
370675	MMAS_HCF	**370675**	0.000	370638.15	0.010	47.12	11
1000-5	SMMAS	**352760**	0.000	**352748.00**	0.003	15.08	12
352760	MMAS_HCF	**352760**	0.000	352740.50	0.006	14.68	7
1000-6	SMMAS	**359629**	0.000	**359601.20**	0.008	68.06	17
359629	MMAS_HCF	**359629**	0.000	359574.65	0.015	85.34	14
1000-7	SMMAS	**371193**	0.000	**371193.00**	0.000	0.00	20
371193	MMAS_HCF	**371193**	0.000	**371193.00**	0.000	0.00	20
1000-8	SMMAS	**351994**	0.000	**351970.90**	0.007	32.74	13
351994	MMAS_HCF	**351994**	0.000	351894.40	0.028	38.73	2
1000-9	SMMAS	**349337**	0.000	**349308.10**	0.008	111.26	18
349337	MMAS_HCF	**349337**	0.000	349266.45	0.020	30.41	3
1000-10	SMMAS	**351415**	0.000	**351415.00**	0.000	0.00	20
351415	MMAS_HCF	**351415**	0.000	351308.25	0.030	135.49	10

Bộ dữ liệu lớn (n=2500): Trong bộ dữ liệu lớn này cho thấy hiệu của của SMMAS khi kết hợp với tìm kiếm cục bộ so với MMAS_HCF kết hợp với tìm kiếm

cục bộ. 7/10 test SMMAS cho kết quả tốt nhất tốt hơn MMAS_HCF, 9/10 test SMMAS cho kết quả trung bình tốt hơn MMAS_HCF. (Bảng 3.7)

Bảng 3.7: So sánh SMMAS với MMAS_HCF với bộ dữ liệu *n*=2500 (có sử dụng tìm kiếm cục bộ)

thời gian chạy 1000 giây (1)		best (2)	%best (3)	avg-best (4)	%avg (5)	std (6)	suc(/20) (7)
2500-1	SMMAS	**1515944**	0.000	**1515788.65**	0.010	144.49	6
1515944	MMAS_HCF	1515913	0.002	1515570.65	0.025	197.86	0
2500-2	SMMAS	**1471083**	0.021	**1470622.45**	0.052	314.13	0
1471392	MMAS_HCF	1470885	0.034	1470497.55	0.061	217.29	0
2500-3	SMMAS	**1414192**	0.000	**1413688.40**	0.036	348.38	4
1414192	MMAS_HCF	1414075	0.008	1413602.70	0.042	296.41	0
2500-4	SMMAS	1507701	0.000	1507700.20	0.000	2.462	18
1507701	MMAS_HCF	1507701	0.000	**1507701.00**	0.000	0.00	20
2500-5	SMMAS	1491816	0.000	**1491768.40**	0.003	74.76	4
1491816	MMAS_HCF	1491816	0.000	1491752.10	0.004	88.24	2
2500-6	SMMAS	**1469162**	0.000	**1468847.45**	0.021	264.49	3
1469162	MMAS_HCF	1469022	0.010	1468567.25	0.040	325.92	0
2500-7	SMMAS	**1479040**	0.000	**1478567.90**	0.032	409.96	2
1479040	MMAS_HCF	1478730	0.021	1478369.50	0.045	333.38	0
2500-8	SMMAS	1484199	0.000	**1483821.50**	0.025	249.66	3
1484199	MMAS_HCF	1484199	0.000	1483760.40	0.030	261.98	3
2500-9	SMMAS	**1482413**	0.000	**1482346.90**	0.004	61.80	1
1482413	MMAS_HCF	1482399	0.001	1482297.10	0.008	108.49	0
2500-10	SMMAS	**1483178**	0.012	**1482629.90**	0.049	336.41	0
1483355	MMAS_HCF	1482861	0.033	1482309.85	0.070	281.61	0

Kết luận: Khi không sử dụng tìm kiếm cục bộ với bộ dữ liệu nhỏ (n=200), SMMAS chưa tốt hơn nhiều so với MMAS_HCF. Nhưng với bộ dữ liệu trung bình (n=500) SMMAS tốt hơn hẳn MMAS_HCF. Tương tự như vậy, khi áp dụng tìm kiếm cục bộ, SMMAS chưa tốt hơn nhiều so với MMAS_HCF trên bộ dữ liệu trung bình (n=500, n=1000), nhưng với bộ dữ liệu lớn (n=2500), SMMAS tốt hơn hẳn MMAS_HCF. Như vậy, khi giải bài toán không có thông tin heuristic và có (hoặc không) sử dụng tìm kiếm cục bộ thì SMMAS vẫn tốt hơn MMAS.

3.7. Kết luận chương

Dựa trên phân tích tính biến thiên của vết mùi, luận án đề xuất các quy tắc cập nhật mùi mới: SMMAS, MLAS và 3-LAS. Các thuật toán này sẽ bất biến đối với phép biến đổi đơn điệu hàm mục tiêu. Thực nghiệm ở bài toán TSP cho thấy ba quy tắc này tốt như nhau và tốt hơn quy tắc MMAS thông dụng nhất hiện hành. *Tuy nhiên, trong các thuật toán này, SMMAS đơn giản và dễ sử dụng hơn nhiều. Vì vậy khi áp dụng cho các bài toán mới thì nên dùng SMMAS này, đặc biệt với các bài toán lớn đòi hỏi nhiều thời gian thí nghiệm để xác định các tham số.*

Thuật toán MLAS cho phép điều tiết linh hoạt khả năng khám phá và tăng cường của thuật toán theo từng thời điểm nhưng khó áp dụng hơn mặc dù thực nghiệm cho kết quả hứa hẹn.

Thuật toán 3-LAS thích hợp với các bài toán có thông tin heuristic tốt, khi sử dụng chúng ảnh hưởng nhiều tới chất lượng của kết quả tìm kiếm, chẳng hạn như bài toán TSP.

Trong các chương sau, các bài toán Tin-Sinh học đòi hỏi thời gian chạy lớn và thông tin heuristic không mạnh nên luận án chỉ sử dụng quy tắc SMMAS.

Chương 4. THUẬT TOÁN ACOHAP GIẢI BÀI TOÁN SUY DIỄN HAPLOTYPE

Suy diễn haplotype giúp ta hiểu được cấu trúc di truyền của quần thể dựa trên dữ liệu kiểu gen (genotype) của các tổ chức lưỡng bội. Theo tiêu chuẩn tìm tập haplotype nhỏ nhất (*pure parsimony*), bài toán suy diễn haplotype trở thành bài toán tối ưu tổ hợp thuộc lớp NP-khó. Chương này, luận án đề xuất một thuật toán hiệu quả có tên là ACOHAP [24] giải bài toán suy diễn haplotype theo tiêu chuẩn pure parsimony. Thực nghiệm trên dữ liệu chuẩn và dữ liệu thực cho thấy ưu điểm nổi trội của nó so với các phương pháp tốt nhất hiện thời.

4.1. Bài toán suy diễn haplotype và tiêu chuẩn pure parsimony

Trong các tổ chức lưỡng bội, hầu hết các nhiễm sắc thể có hai "*bản sao*" không giống nhau. Một haplotype là một *bản sao* của một genotype trong một tổ chức lưỡng bội, mang các thông tin cho phép nghiên cứu các triệu chứng và tác nhân gây bệnh di truyền. Tuy nhiên, do các hạn chế về công nghệ, người ta không thể thu thập được các haplotype nhờ thực nghiệm, mà chỉ thu thập được các genotype được kết hợp từ các cặp haplotype.

Bài toán suy diễn haplotype là từ một tập n genotype có độ dài m, xác định tập haplotype sao cho các cặp kết hợp từ chúng tạo nên được tập genotype đang xét. Hiện nay, bài toán suy diễn haplotpe là thách thức quan trọng trong nghiên cứu di truyền của các sinh vật lưỡng bội nói chung và con người nói riêng [32-35]. Trước khi phát biểu bài toán suy diễn haplotype (*Haplotype Inference - HI*), ta cần đến khái niệm giải thích genotype.

4.1.1. Giải thích genotype

Trong biểu diễn dạng toán học của bài toán suy diễn haplotype, mỗi genotype được biểu diễu bằng một xâu độ dài m các ký tự thuộc tập {0, 1, 2}. Các ký tự 0 và 1 thể hiện allene của genotype ở vị trí tương ứng là đồng hợp tử, ký tự 0 biểu thị allen dạng tự nhiên (wild type) và ký tự 1 biểu thị allen dạng biến dị (mutant), còn ký tự 2

biểu thị cặp allen ở vị trí tương ứng là dị hợp tử. Mỗi haplotype là một xâu độ dài m các ký tự thuộc tập {0,1}. Tại vị trí dị hợp tử, genotype được kết hợp từ hai haplotype mà ở vị trí này một có dạng tự nhiên và một có dạng biến dị.

Với một genotype, ta cần tìm một cặp không thứ tự của haplotype có thể giải thích theo định nghĩa sau:

Định nghĩa 4.1. (Giải thích genotype)

Cho một genotype g, ta nói rằng cặp haplotype không thứ tự $< h^a, h^b >$ giải thích g (hay g được giải thích bởi $< h^a, h^b >$) và ký hiệu là $< h^a, h^b > \rhd g$ nếu chúng thỏa mãn điều kiện sau với mọi vị trí $i = 1,2, \dots, m$:

$$g_i = 0 \text{ thì } h_i^a = h_i^b = 0, \tag{4.1}$$

$$g_i = 1 \text{ thì } h_i^a = h_i^b = 1, \tag{4.2}$$

$$g_i = 2 \text{ thì } (h_i^a = 0 \wedge h_i^b = 1) \text{ hoặc } (h_i^a = 1 \wedge h_i^b = 0) \tag{4.3}$$

Biểu thức (4.1) và (4.2) nghĩa là cả hai haplotype phải có cùng giá trị tại tất các vị trí đồng hợp tử, còn điều kiện (4.3) có nghĩa là tại các vị trí dị hợp tử giá trị của hai haplotype là khác nhau. Với một genotype, ký tự trên cặp haplotype ở vị trí các đồng hợp tử hoàn toàn xác định, còn ký tự ở vị trí dị hợp tử có hai khả năng nhận giá trị. Nếu trong genotype có l vị trí là dị hợp tử thì sẽ có 2^{l-1} cặp không thứ tự haplotype giải thích nó.

Ví dụ, Với $g = 2021$, có 2 cặp haplotype giải thích g là $< 0001, 1011 >$ và $< 0011, 1001 >$

Với một danh sách n genotype, $G = (g^1, \dots, g^n)$ có độ dài m đã cho, trong đó $g^s = (g_1^s, \dots, g_m^s)$ và $g_i^s \in \{0,1,2\}$ với mọi $s \leq n$ và $i \leq m$, ta định nghĩa các haplotype giải thích nó như sau.

Định nghĩa 4.2. (Giải thích tập genotype)

Cho một danh sách n genotype $G = (g^1, \dots, g^n)$ có độ dài m. Ta nói một danh sách $2n$ haplotype $H = (h^{1a}, h^{1b}, h^{2a}, h^{2b}, \dots, h^{na}, h^{nb})$ là một giải thích HI của G nếu g^s được giải thích bởi cặp haplotype $< h^{sa}, h^{sb} >$ với mọi $s \leq n$.

4.2.2. Suy diễn haplotype theo tiêu chuẩn pure parsimony

Như vậy, với một danh sách n genotype, $G = (g^1, \dots, g^n)$ có độ dài m đã cho bài toán HI là tìm danh sách $2n$ haplotype $H = (h^{1a}, h^{1b}, h^{2a}, h^{2b}, \dots, h^{na}, h^{nb})$ giải thích *hợp lý* các genotype này.

Hiện nay có hai cách tiếp cận chính cho bài toán này là: Phương pháp tổ hợp và phương pháp thống kê (xem [33-35,58]). Lời giải cho bài toán tùy thuộc vào mô hình di truyền là tiêu chuẩn cho xác định tập haplotype. Theo phương pháp tổ hợp, tiêu chuẩn *pure parsimony* nhằm tìm tập haplotype nhỏ nhất, giải thích G do Gusfield [34, 35] đề xuất theo gợi ý của Hubbel (xem [34]) đang được nhiều người sử dụng. Lời giải cho bài toán suy diễn haplotype theo tiêu chuẩn pure parsimony (*Haplotype Inference by Pure Parsimony - HIPP*) được phát biểu như sau.

Định nghĩa 4.3. (Lời giải theo tiêu chuẩn pure parsimony)

Nếu danh sách H là một lời giải HI của G có số lượng haplotype khác nhau của H (như là tập hợp) ít nhất, ta nói nó là lời giải của bài toán HI theo tiêu chuẩn pure parsimony (viết tắt là HIPP) hay H suy diễn từ G.

Ví dụ: Với danh sách gồm 3 genotype G=(121, 002, 221) thì H=(**101, 111, 000, 001, 011**, 101) là một lời giải gồm 5 haplotype khác nhau. Tuy nhiên, lời giải H=(**101, 111, 000, 001,**001, 111) gồm 4 haplotype khác nhau mới là lời giải tối ưu.

Bài toán HIPP thuộc loại NP-khó [33,34] và đã có nhiều thuật toán bao gồm heuristic tham ăn, giải đúng và xấp xỉ theo hướng này được công bố (xem [7,11,32-35,43,46,48,58,68,69]). Trong số các thuật toán này, Benedettini và các cộng sự [7] đã

thử nghiệm một thuật toán ACO hai mức nhưng hiệu quả không cao khi so với RPoly [32]. Trong các chương trình hiện hành giải HIPP [32], RPoly là phương pháp giải đúng tốt nhất, còn CollHap [68] là phương pháp giải gần đúng tốt nhất hiện nay. Thuật toán ACOHAP mà luận án đề xuất được trình bày dưới đây sẽ được so sánh với các chương trình này.

4.2. Thuật toán ACOHAP

Trong các thuật toán ACO truyền thống, kiến xây dựng lời giải theo thủ tục bước ngẫu nhiên trên đường đi liên tục. Ở thuật toán này đồ thị cấu trúc là đồ thị con của cây nhị phân có độ sâu m. Chúng được xác định động theo mỗi con kiến ở từng bước lặp. Mỗi mức của đồ thị biểu thị cho một vị trí trên haplotype mà kiến xây dựng lời giải.

4.2.1. Mô tả thuật toán

Không giảm tính tổng quát, ta giả thiết các genotype trong G đều khác nhau. Với số vòng lặp N_C hoặc thời gian chạy xác định trước, thuật toán ACOHAP được mô tả như trong hình 4.1.

Procedure ACOHAP;
Dữ liệu vào: n genotype
Kết quả ra: n cặp haplotype suy diễn n genotype và số lượng haplotype khác nhau
Begin
  Khởi tạo tập A gồm N_a kiến, ma trận mùi, các tham số $\rho, \alpha, \beta, \tau_{\min}, \tau_{\max}$
  while (chưa kết thúc) **do**
    for each a $\in$ A **do**
      Kiến a xây dựng lời giải HI; {gồm $2n$ đường đi từ gốc đến lá trên đồ thị cấu trúc;}
    end-for
    Sử dụng tìm kiến cục bộ cải tiến lời giải;
    Cập nhật mùi;
    Cập nhật lời giải tốt nhất;
  end-while
  Đưa ra lời giải tốt nhất;
End;

Hình 4.1: Thuật toán ACOHAP

4.2.2. Đồ thị cấu trúc

Về hình thức, đồ thị cấu trúc là cây nhị phân đầy đủ có độ sâu m. Tuy nhiên, để tránh bùng nổ tổ hợp khi m lớn, đối với mỗi kiến ở mỗi bước ta chỉ hiện thị một cây con T của cây nhị phân đầy đủ, được trích từ quá trình xây dựng lời giải của nó với nút gốc ở mức 0 và các nút lá ở mức m. Các cây này biểu thị khác nhau (một cách động) phù hợp với quá trình xây dựng lời giải của mỗi con kiến trong mỗi lần lặp và có các đặc điểm sau.

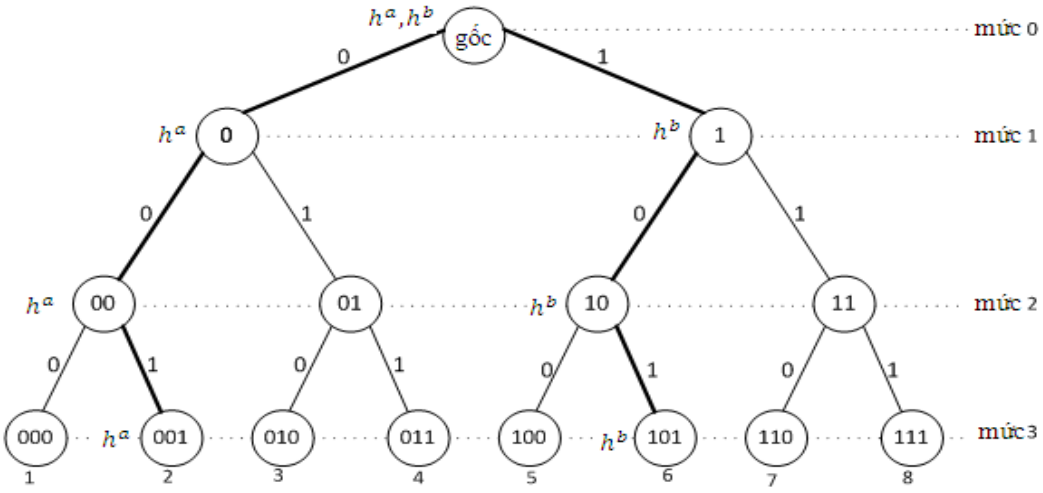
- Mỗi nút trong X ở mức i có hai nút con tại mức $i+1$. Nhánh từ X sang con bên trái có nhãn là 0 (gọi là nhánh 0). Tương tự, nhánh từ X sang con bên phải có nhãn là 1 (gọi là nhánh 1).
- Nhãn của nhánh trên đường đi từ nút gốc đến nút X tạo thành nhãn của nút X. Nhãn của nút X tại mức i là i ký tự đầu tiên của haplotype (nhãn của nút lá sẽ là một haplotype độ dài m)
- Mỗi nút có một danh sách kết hợp chỉ các haplotype được xây dựng nhờ đường đi đến nút này. Như vậy nút gốc luôn có danh sách kết hợp là $h_1^a, h_1^b, h_2^a, h_2^b, .., h_n^a, h_n^b$, các nút trên đường đi từ gốc đến lá có danh sách tương ứng giảm dần.
- Mỗi đường đi từ gốc đến lá xác định haplotype có trong danh sách tương ứng ở nút lá và nhãn của nút xác định nội dung của haplotype.

Như vậy, đồ thị T này có nhiều nhất $2n$ nút lá biểu thị các haplotype cần tìm chứ không phải có 2^m nút như cây nhị phân đầy đủ. Đồ thị này không xác định ngay từ đầu mà được hiển thị dần theo quá trình xây dựng lời giải được nói rõ hơn ở phần dưới.

Hình 4.2 mô tả cây độ sâu bằng 3 giúp xây dựng cặp haplotype $< h^a, h^b >$ giải thích genotype g =(201). Kiến sẽ xây dựng hành trình xuất phát từ nút gốc đến nút lá để xác định haplotype h^a cho genotype g. Hai haplotype h^a=(001) và h^b=(101) (đường đậm trong hình 4.2) là cặp haplotype giải thích g=(201). Đồ thị T chỉ gồm các

nút và cạnh của cây nhị phân nằm trên đường in đậm. Các nút còn lại là các nút ảo, chúng thuộc cây nhị phân đầy đủ nhưng không đưa vào T để tránh bùng nổ tổ hợp khi m lớn.

Thủ tục xây dựng lời giải của mỗi con kiến dưới đây sẽ giúp hiểu rõ hơn tính mềm dẻo của đồ thị cấu trúc và cách xây dựng.



Hình 4.2: Đồ thị cấu trúc (phần thuộc đường đậm) xác định giải thích genotype 201

4.2.3. Thủ tục xây dựng lời giải của mỗi con kiến

Trong phần này luận án sẽ trình bày thuật toán xây dựng đồng thời $2n$ haplotype của mỗi kiến lần lượt theo từng vị trí để suy diễn cả n genotype của G. Để thực hiện xây dựng lời giải, mỗi nút của cây sẽ có một danh sách haplotype kết hợp có ý nghĩa các haplotype trong danh sách sẽ nhận giá trị là nhãn của nút đó cho các vị trí từ đấy về trước.

Ban đầu, nút gốc được khởi tạo có một danh sách kết hợp gồm $2n$ haplotype $(h^{1a}, h^{1b}, .., h^{na}, h^{nb})$ rồi thực hiện m lần lặp, trong đó lần lặp thứ i sẽ xác định giá trị ở vị trí thứ i cho tất cả các haplotype và tạo danh sách kết hợp cho các nút ở mức i

(trước đó danh sách này rỗng). Mỗi lần lặp, kiến thực hiện lần lượt hai bước: bước thứ nhất xử lý đồng hợp tử và bước thứ hai xử lý dị hợp tử.

Bước thứ nhất: Xử lý đồng hợp tử. Với các genotype có vị trí thứ i là đồng hợp tử thì các cặp haplotype tương ứng ở vị trí thứ i sẽ nhận giá trị bằng giá trị vị trí thứ i trên genotype. Cụ thể, nếu $g_i^s = 0/1$ thì h_i^{sa}, h_i^{sb} nhận giá trị 0/1. Khi đó, h^{sa}, h^{sb} được thêm vào danh sách nút con theo nhánh 0/1 tương ứng.

Bước thứ hai: Xử lý dị hợp tử. Với các genotype có vị trí thứ i là dị hợp tử thì giá trị hai haplotype tương ứng ở vị trí thứ i sẽ có giá trị khác nhau. Như vậy nếu xác định được giá trị thứ i của haplotype thứ nhất sẽ tính được giá trị thứ i của haplotype thứ hai. Cụ thể, nếu $g_i^s = 2$ thì h_i^{sa} sẽ lựa chọn 0 hoặc 1, h_i^{sb} sẽ bằng $1 - h_i^{sa}$. Nếu ở danh sách của nút mức $(i-1)$ chứa h^{sa} thì kiến lựa chọn ngẫu nhiên h_i^{sa} theo xác suất như sau:

$$P_i^s(v) = \frac{(\tau_{i,v}^s)^\alpha (\eta_{i,v}^s)^\beta}{(\tau_{i,0}^s)^\alpha (\eta_{i,0}^s)^\beta + (\tau_{i,1}^s)^\alpha (\eta_{i,1}^s)^\beta} \quad (4.4)$$

trong đó α và β là hai tham số dương cho trước điều khiển ảnh hưởng giữa thông tin vết mùi và thông tin heuristic (sẽ trình bày cách xác định ở mục 4.2.4). Giả sử $h_i^{sa} = 0$ thì $h_i^{sb} = 1$, khi đó h^{sa} được thêm vào danh sách nút con bên trái (theo nhánh 0), còn h^{sb} được thêm vào danh sách nút con bên phải (theo nhánh 1) của nút mức $(i-1)$ có h^{sb} trong danh sách kết hợp.

Chú ý rằng, tại mỗi vòng lặp, bước xử lý đồng hợp tử luôn được xử lý trước bước dị hợp tử để tạo thông tin heuristic cho việc lựa chọn giá trị ở bước xử lý dị hợp tử. Việc tính thông tin heuristic sẽ được trình bày trong phần 4.2.4.

Sau khi thực hiện xong m lần lặp (cũng là xây dựng xong m vị trí cho tất cả các haplotype), số lượng nút lá (nút ở mức m) có danh sách kết hợp khác rỗng chính là số

lượng haplotype khác nhau cần dùng để suy diễn n genotype. Thủ tục xây dựng lời giải được mô tả trong hình 4.3.

Procedure Xây dựng lời giải;
Dữ liệu vào: n genotype
Kết quả ra: n cặp haplotype suy diễn n genotype và số lượng haplotype khác nhau
Begin
  Khởi tạo danh sách kết hợp ở nút gốc $(h^{1a}, h^{1b}, .., h^{na}, h^{nb})$
  for i=1 **to** m **do**
    {Bước xử lý đồng hợp tử}
    for s=1 **to** n **do**
      if $(g_i^s = 0) or (g_i^s = 1)$ **then**
        $h_i^{sa} = h_i^{sb} = g_i^s$;
        thêm h_i^{sa}, h_i^{sb} vào danh sách kết hợp ở mức i tương ứng;
      end-if
    end-for

    {Bước xử lý dị hợp tử}
    for s=1 **to** n **do**
      if $(g_i^s = 2)$ **then**
        Xác định h_i^{sa} theo công thức 4.4;
        $h_i^{sb} = 1 - h_i^{sa}$;
        thêm h_i^{sa}, h_i^{sb} vào danh sách kết hợp ở mức i tương ứng;
      end-if
    end-for
  end-for
End;

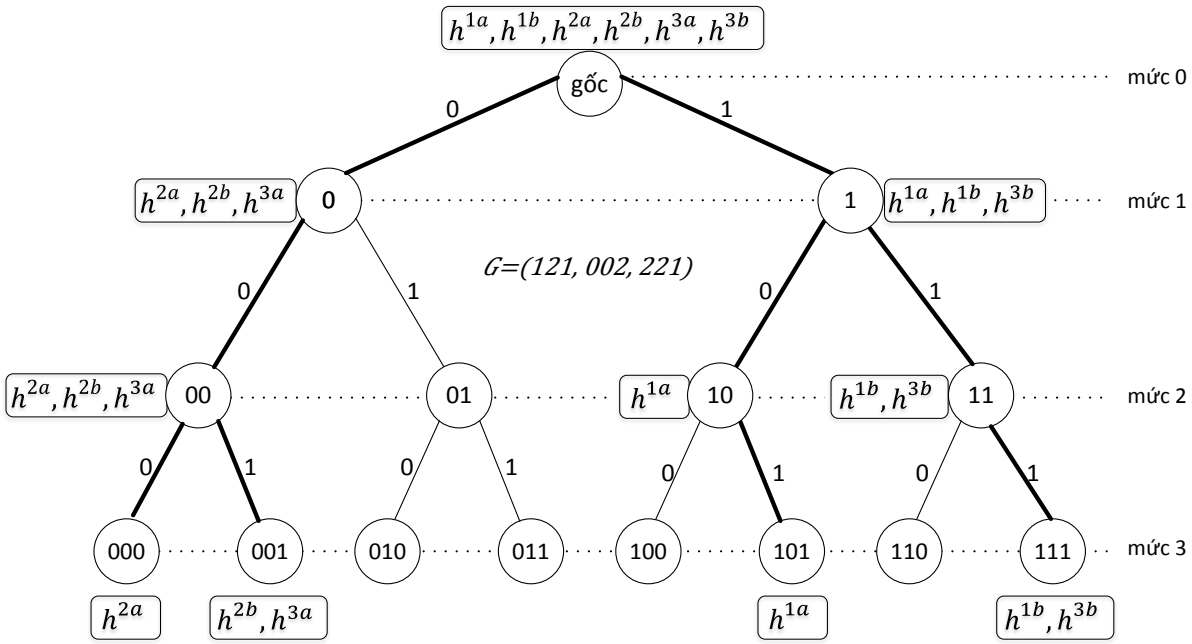
Hình 4.3: Thủ tục tìm lời giải của mỗi con kiến

Ví dụ minh họa

Xét bài toán HIPP với G gồm 3 genotype: g^1=121, g^2=002, g^3=221. Lời giải của một con kiến được minh họa trong hình 4.4 với các haplotype là:

$$h^{2a} = 000, h^{2b} = h^{3a} = 001, h^{1a} = 101, h^{1b} = h^{3b} = 111.$$

Đồ thị cấu trúc động gồm các nút và cạnh thuộc đường in đậm. Như vậy, $H = \{000,001,101,111\}$



Hình 4.4: Lời giải HI của một con kiến với g^1=121, g^2=002, g^3=221

4.2.4. Thông tin heuristic

Như đã nói ở trên, các thông tin heuristic $\eta_{i,0}^s$ và $\eta_{i,1}^s$ sẽ cho phép định hướng xác định giá trị tương ứng (0 hoặc 1) của h_i^{sa} ở các vị trí dị hợp tử i của genotype g^s. Số lượng nút lá (nút ở mức m) có danh sách kết hợp khác rỗng chính là số lượng haplotype khác nhau cần dùng để suy diễn n genotype. Vì vậy, ý tưởng chính để xác định thông tin heuristic cho nút đang xét là ước lượng số nút có danh sách kết hợp khác rỗng ở mức m sẽ tương thích với haplotype đang xét. Để xác định các thông tin này, ta cần định nghĩa về tính tương thích [7] của các haplotype và genotype cùng độ dài.

Định nghĩa 4.4. (Tính tương thích)

i) Ta nói haplotype h tương thích với genotype g nếu với mọi vị trí i mà g_i=0 hoặc g_i=1 thì đều có $h_i = g_i$, khi đó ký hiệu $h \mapsto g$.

ii) Hai genotype g^s và g^t được gọi là tương thích với nhau nếu tồn tại h sao cho $h \mapsto g^s$ và $h \mapsto g^t$, hơn nữa, ký hiệu $h \mapsto \{g^1, g^2, .., g^n\}$ nếu $h \mapsto g^s$ với mọi $s = 1,2,..,n$.

Nhờ định nghĩa này, với mỗi tập G gồm n genotype độ dài m, ta xác định ma trận quan hệ tương thích $C(G) = (c_{s,t})$, trong đó $c_{s,t} = 1$ nếu g^s tương thích với g^t và $c_{s,t} = 0$ nếu g^s không tương thích với g^t.

Nhận xét sau là cơ sở để ta xác định thông tin heuristic khi kiến xây dựng lời giải.

Cho hai genotype tương thích g^s, g^t và h^s là haplotype giải thích cho genotype g^s, h^t là haplotype giải thích cho genotype g^t, h^s và h^t đang trong cùng danh sách kết hợp của một nút. Khi đó h^s và h^t có thể cùng danh sách kết hợp ở nút lá (có nghĩa là h^s và h^t có thể giống nhau).

Gọi c_0^{sa} là số lượng haplotype tương thích với h^{sa} đang nằm trong danh sách kết hợp ở con bên trái, còn c_1^{sa} là số lượng haplotype tương thích với h^{sa} nằm trong danh sách kết hợp ở con bên phải. Tương tự, gọi c_0^{sb} là số lượng haplotype tương thích với h^{sb} nằm trong danh sách kết hợp ở con bên trái còn c_1^{sb} là số lượng haplotype tương thích với h^{sb} nằm trong danh sách kết hợp ở con bên phải.

Quan hệ giữa các cặp đại lượng (c_0^{sa}, c_1^{sb}) và (c_1^{sa}, c_0^{sb}) cho ta thông tin định hướng chọn giá trị 0 hoặc 1 đối với h_i^{sa}. Gọi w_0^s và w_1^s tương ứng là số giá trị bằng 0 của cặp (c_0^a, c_1^b) và (c_1^a, c_0^b). Như vậy, w_0^s và w_1^s nhận các giá trị trong tập {0,1,2}. Khi đó $\eta_{i,0}^s$ và $\eta_{i,1}^s$ được xác định theo 3 khả năng như sau:

1) Nếu $w_0^s = w_1^s$ thì $\eta_{i,0}^s = (c_0^a + c_1^b + 1) \times m$; $\eta_{i,1}^s = (c_1^a + c_0^b + 1) \times m$ (4.5)

2) Nếu $w_0^s > w_1^s$ thì $\eta_{i,1}^s = \frac{\tau_{max}}{\tau_{min}}$; $\eta_{i,0}^s = 1.0$ (4.6)

3) Nếu $w_0^s < w_1^s$ thì $\eta_{i,0}^s = \frac{\tau_{max}}{\tau_{min}}$; $\eta_{i,1}^s = 1.0$ (4.7)

Dễ dàng kiểm tra được độ phức tạp tính toán để xác định $C(G)$ và sử dụng thông tin heuristic đều là $O(mn^2)$.

Chú ý. Thuật toán heuristics PTG của Li đề xuất [48] cũng dựa trên cây nhị phân và cũng có độ phức tạp là $O(n^2m)$ nhưng đánh giá thông tin heuristic không tốt bằng phương pháp được đề xuất trong luận án. Thực nghiệm trong luận án cho thấy lời giải của một kiến xây dựng theo cách mô tả trên cho kết quả tốt hơn lời giải của PTG tìm được.

4.2.5. Cập nhật vết mùi

Tương tư như thông tin heuristic, ta chỉ quan tâm tới vết mùi ở các vị trí dị hợp tử của genotype, tức là các vị trí của genotype g^s có $g_i^s = 2$. Các vết mùi được khởi tạo $\tau_{i,0}^s = \tau_{i,1}^s = \tau_{max}$ cho trước. Sau mỗi vòng lặp, kiến cập nhật mùi theo quy tắc SMMAS. Kiến trao đổi thông tin với nhau để xác định lời giải xấp xỉ tốt nhất của bài toán HIPP trong vòng lặp (có số haplotype cực tiểu). Dựa vào bộ haplotype $h^{1a}, h^{1b}, .., h^{na}, h^{nb}$ của lời giải này, ma trận vết mùi $\tau_{i,v}^s$ được cập nhật theo công thức (4.8):

$$\tau_{i,v}^s \leftarrow (1-\rho)\tau_{i,v}^s + \Delta\tau_{i,v}^s \text{ với } \Delta\tau_{i,v}^s = \begin{cases} \rho\tau_{min} \text{ nếu } h_i^{sa} \neq v \\ \rho\tau_{max} \text{ nếu } h_i^{sa} = v \end{cases} \quad (4.8)$$

4.2.6. Hoán vị thứ tự xử lý các vị trí trong bộ genotype

Trong thủ tục xây dựng lời giải của kiến ở trên, việc xử lý tuần tự theo thứ tự tăng dần của genotype và vị trí của gen tạo nên sự phụ thuộc có định hướng vào các thứ tự này. Thực ra, thì vai trò của mỗi genotype và các vị trí là bình đẳng và có thể chọn thứ tự xử lý tùy ý để đảm bảo tính ngẫu nhiên. Vì vậy, ở mỗi lần xây dựng lời giải của mỗi kiến, ta hoán vị ngẫu nhiên các cột và các hàng của ma trận G để xây

dựng lời giải, sau đó khôi phục lại theo hoán vị đã xét. Điều này giúp cho ta phần nào khắc phục được tối ưu cục bộ. Công việc này dễ dàng thực hiện được mà không làm tăng độ phức tạp tính toán, nhưng cải thiện đáng kể chất lượng thuật toán.

Chú ý. Nhắc lại rằng so với các thuật toán ACO truyền thống [31], kiến xây dựng lời giải ở đây không đi theo đường đi liên tục. Các thông tin heuristic và học tăng cường (vết mùi lưu trữ riêng chứ không gắn liền với cạnh hoặc đỉnh). Ngoài ra, thay vì xét đồ thị cấu trúc cây nhị phân đầy đủ, ta xét đồ thị con động của nó đối với mỗi kiến ở từng bước lặp.

4.2.7. Sử dụng tìm kiếm cục bộ

Để tăng hiệu quả thuật toán, trong mỗi lần lặp luận án sử dụng thuật toán tìm kiếm cục bộ cho lời giải tìm được theo chiến lược cải tiến *tốt hơn* trong lân cận khoảng cách 1-Hamming do Gaspero và Roli đề xuất trong [19]. Độ phức tạp khi áp dụng kỹ thuật tìm kiếm cục bộ này là $O(n^2m)$.

Cách áp dụng tìm kiếm cục bộ như sau: chọn k ($k \leq$ số kiến) lời giải tốt nhất tìm được trong mỗi lần lặp để cải tiến bằng tìm kiếm cục bộ, rồi chọn lời giải tốt nhất trong k lời giải đó để cập nhật mùi.

4.2.8. Độ phức tạp thuật toán

Thuật toán ACOHAP thực hiện N_c vòng lặp, mỗi vòng lặp có N_a kiến tìm lời giải. Độ phức tạp của thủ tục xây dựng một lời giải là $O(n^2m)$, còn độ phức tạp của thủ tục cục bộ để cải tiến một lời giải là $O(n^2m)$. Như vậy, thuật toán sử dụng tìm kiếm cục bộ cho k ($k \leq N_a$) lời giải tốt nhất trong N_a kiến tham gia xây dựng trong một lần lặp mất: $(N_a \times n^2m + k \times n^2m)$. Do đó, thuật toán có độ phức tạp là: $O(N_c \times N_a \times n^2m)$.

4.3. Kết quả thực nghiệm

Luận án tiến hành thực nghiệm trên hai loại bộ dữ liệu: bộ dữ liệu chuẩn và dữ liệu thực để so sánh với phương pháp RPoly [32] và phương pháp CollHap [68]. Bộ dữ liệu chuẩn gồm các tập SU1, SU2, SU3, SU-100kb được lấy từ địa chỉ [79]. Còn bộ bộ dữ liệu thực là nhiễm sắc thể 20 của người da trắng châu Âu tại Utah [58].

Bảng 4.1: Thiết đặt tham số cho ACOHAP

Tham số	**Giá trị**	**Mô tả**
N_a	30	Số kiến
τ_{max}	1.0	Cận trên của vết mùi
τ_{min}	$\frac{\tau_{max}}{2mn}$	Cận dưới của vết mùi
α	1	Tham số điều khiển ảnh hưởng của vết mùi
β	1	Tham số điều khiển ảnh hưởng của thông tin heuristic
ρ	0.5	Tham số bay hơi
k	N_a	Số lời giải được áp dụng tìm kiếm cục bộ trong mỗi vòng lặp

Tất cả các chương trình đều được chạy trên cùng một máy trong khoảng thời gian 1000 giây. Tham số thiết đặt cho ACOHAP như trong bảng 4.1. Phương pháp RPoly là phương pháp chính xác tốt nhất hiện nay, phương pháp này cho kết quả tối ưu, nhưng đòi hỏi thời gian tính toán lớn và chỉ phù hợp với bộ dữ liệu nhỏ hoặc đơn giản (mối quan hệ giữa các genotype ít). Còn phương pháp CollHap là phương pháp xấp xỉ tốt nhất hiện nay. Phương pháp heuristic PTG [48] nhanh, tuy nhiên kết quả không tốt bằng các phương pháp khác. Luận án không so sánh với ACO_HI [7], vì kết quả của phương pháp này rất hạn chế và chương trình hiện không chạy được.

4.3.1. Thực nghiệm trên bộ dữ liệu chuẩn

Bộ dữ liệu chuẩn gồm các tập SU1, SU2, SU3, SU-100kb được lấy từ địa chỉ [79] đã được Marchini [52] dùng để làm dữ liệu chuẩn, so sánh với các phương pháp khác. Sau đây là bảng tóm tắt thông tin về các bộ dữ liệu này.

Bảng 4.2: Bảng tóm tắt thông tin về các bộ dữ liệu chuẩn

Dữ liệu chuẩn	Số lượng test	Số lượng genotype	Độ dài genotype
SU-100kb	29	90	18
SU1	100	90	179
SU2	100	90	171
SU3	100	90	187

Với bộ dữ liệu chuẩn, luận án không so sánh với PTG và RPoly, vì giao diện của PTG không thuận lợi để chạy nhiều bộ dữ liệu, còn RPoly chỉ phù hợp với dữ liệu có kích thước nhỏ hoặc dữ liệu đơn giản. Luận án cho chương trình ACOHAP và CollHap chạy trên cùng một máy trong khoảng thời gian 1000 giây với từng dữ liệu.

Bảng 4.3: Kết quả thực nghiệm so sánh ACOHAP với CollHap với dữ liệu chuẩn

Bộ dữ liệu	Số lượng test ACOHAP tốt hơn CollHap	Số lượng test ACOHAP bằng CollHap	Số lượng test ACOHAP kém hơn CollHap
SU-100kb	7%	86%	7%
SU1	76%	24%	0%
SU2	40%	55%	5%
SU3	66%	34%	0%

Kết quả thực nghiệm cho thấy, với bộ dữ liệu nhỏ SU-100kb, ACOHAP tốt như CollHap. Còn với các bộ dữ liệu lớn SU1, SU2 và SU3, ACOHAP nổi trội hơn hẳn CollHap.

4.3.2. Thử nghiệm trên dữ liệu thực

Trong mục này, luận án thực nghiệm trên bộ dữ liệu thực cho cả 4 phương pháp. Dữ liệu haplotype được thu thập từ HapMap giai đoạn III, nhiễm sắc thể 20 của người da trắng châu Âu tại Utah (*Caucasian European in Utah - CEU*). Thông tin về các bộ dữ liệu thực trong bảng 4.4.

Bảng 4.4: Thông tin dữ liệu thực

Data set	#Test	#genotype	Độ dài genotype
CEU-100	3	88	100
CEU-200	3	88	200
CEU-400	3	88	400
CEU-800	3	88	800
CEU-1600	1	88	1600

Kết quả thực nghiệm trong bảng 4.5 cho thấy, PTG hiệu quả kém nhất trong số 4 phương pháp. ACOHAP tốt hơn CollHap trong 8/13 test, còn tốt như CollHap 5/13 test.

RPoly có thể tìm được tối ưu trong 10/13 test. Trong 10 test chỉ có 2 test ACOHAP tìm được kết quả gần tối ưu (chỉ lệch 1 đến 2 haplotype so với kết quả tối ưu). Còn 3/13 test RPoly không chạy được, ACOHAP chạy được và cho kết quả tốt hơn CollHap và PTG. Như vậy, phương pháp ACOHAP là phương pháp xấp xỉ, nhưng cũng đã tìm được kết quả tối ưu trong rất nhiều trường hợp. So với CollHap, PTG thì ACOHAP hiệu quả nổi trội hơn hẳn.

Bảng 4.5: Kết quả thực nghiệm với dữ liệu thực

Dữ liệu thực	RPoly	ACOHAP	CollHap	PTG
$A1(n = 16; m = 100)$	**12**	**12**	**12**	**12**
$A2(n = 83; m = 100)$	-	**80**	81	88
$A3(n = 88; m = 100)$	-	**135**	139	169
$B1(n = 66; m = 200)$	**38**	**38**	39	57
$B2(n = 88; m = 200)$	**120**	122	125	161
$B3(n = 88; m = 200)$	**140**	141	145	169
$C1(n = 86; m = 400)$	-	85	87	88
$C2(n = 88; m = 400)$	**143**	**143**	146	172
$C3(n = 88; m = 400)$	**170**	**170**	**170**	175
$D1(n = 88; m = 800)$	**176**	**176**	**176**	**176**
$D2(n = 88; m = 800)$	**162**	**162**	164	175
$D3(n = 88; m = 800)$	**175**	**175**	**175**	**175**
$E(n = 88; m = 1600)$	**176**	**176**	**176**	**176**

Ghi chú: Cột dữ liệu thực là thông tin về test (số lượng genotype khác nhau và độ dài genotype), cột RPoly, ACOHAP, CollHap và PTG tương ứng là kết quả số lượng haplotype tìm được của các phương pháp RPoly, ACOHAP, CollHap và PTG. Kết quả tốt nhất trong các phương pháp được tô đậm.

4.4. Kết luận chương

Trên đây luận án đã đề xuất một thuật toán ACO cho bài toán HIPP với đồ thị đồ thị cấu trúc là một cây nhị phân động cho phép làm việc với các bài toán có độ dài genotype lớn, thông tin heuristic hợp lý và quy tắc cập nhật mùi dễ sử dụng. Thủ tục xây dựng lời giải và cập nhật thông tin mùi không gắn trên đồ thị cấu trúc như các thuật ACO thông dụng [31]. Kết quả thực nghiệm so sánh với các phương pháp RPoly (phương pháp đúng tốt nhất hiện nay) và CollHap (phương pháp xấp xỉ tốt nhất hiện nay) cho thấy hiệu quả của thuật toán đề xuất trên dữ liệu chuẩn và dữ liệu thực.

Tuy nhiên, việc áp dụng tìm kiếm cục bộ còn rập khuôn theo [19], chưa có cải tiến thích hợp. Trong thời gian tới có thể nghiên cứu cải tiến kỹ thuật tìm kiếm cục bộ để tăng hiệu quả thuật toán.

Chương 5. THUẬT TOÁN AcoSeeD TÌM TẬP HẠT GIỐNG CÓ CÁCH TỐI ƯU

Tìm kiếm các đoạn tương tự trong các chuỗi sinh học là một trong những công việc thường gặp và quan trọng nhất trong tin sinh học. Để nâng cao chất lượng tìm kiếm, hiện nay người ta sử dụng tập hạt giống có cách để sắp hàng địa phương và sau đó thác triển ra để tìm các đoạn tương đồng. Tuy nhiên, tìm tập hạt giống có cách tối ưu là bài toán thuộc lớp NP-khó. Chương này, luận án đề xuất một thuật toán sử dụng phương pháp ACO cho bài toán này có tên là AcoSeeD [22]. Thuật toán AcoSeeD có đồ thị cấu trúc hợp lý, dùng quy tắc cập nhật mùi SMMAS và kỹ thuật tìm kiếm cục bộ được định hướng bằng một hàm mục tiêu xấp xỉ nhanh thay cho hàm mục tiêu chính trong phương pháp ACO. Kết quả thực nghiệm cho thấy AcoSeeD đã cải thiện đáng kể hiệu quả so với thuật toán tốt nhất hiện nay: SpEEDfast [50,51].

5.1. Bài toán tìm tập hạt giống có cách tối ưu và một số vấn đề liên quan

5.1.1. Bài toán tìm tập hạt giống tối ưu

Bài toán gióng hàng địa phương để tìm các đoạn con tương đồng giữa hai hay nhiều chuỗi sinh học là quan trọng và thường gặp nhất trong sinh học phân tử. Với công nghệ giải mã gen ngày càng phát triển như hiện nay, dữ liệu chuỗi sinh học ngày càng nhiều nên rất cần các thuật toán tìm kiếm nhanh, hiệu quả.

Thuật toán đầu tiên để tìm đoạn tương đồng giữa hai chuỗi được Smith và Waterman đề xuất là một thuật toán quy hoạch động [62] để tìm lời giải chính xác. Thuật toán này có nhược điểm là thời gian chạy lớn nên không dùng được cho các chuỗi có độ dài lớn. Vì vậy, người ta phát triển các thuật toán heuristic tìm lời giải ít chính xác hơn nhưng thời gian chạy nhanh hơn (xem [13,50,11]), trong đó chương trình thông dụng nhất là BLAST [4]. Các phương pháp này dựa trên ý tưởng *hợp và*

mở rộng, trong đó việc so khớp gần đúng giữa hai chuỗi sinh học thực hiện nhờ các đoạn có độ dài cho trước gọi hạt giống (seed) để tìm kiếm và mở rộng ra.

Để tăng chất lượng lời giải, Li và các cộng sự (xem [47]) đề xuất dùng các hạt giống này không đòi hỏi so khớp chính xác gọi là hạt giống có cách (spaced seed). Về sau ta chỉ làm việc với hạt giống có cách nên ta sẽ gọi gọn là hạt giống. Đến nay, có nhiều thuật toán được đề xuất theo cách tiếp cận này (xem [13,17,41,50,51]). Chất lượng tìm kiếm ảnh hưởng rất nhiều từ tập hạt giống được sử dụng trong tìm kiếm. Dưới đây, luận án phát biểu bài toán tìm tập giống tối ưu theo mô hình Bec_nu_li được đề xuất trong [47] và đã được nhiều tác giả sử dụng (xem [49-51]).

Việc so khớp địa phương hai hay nhiều chuỗi sinh học được đưa về xét bài toán trên một miền tương đồng (homologous region) biểu diễn bằng xâu nhị phân R có độ dài N, ký tự 0 ở mỗi vị trí i của R biểu thị không khớp (mismatch) còn ký tự 1 biểu thị khớp (match). Chuỗi R sẽ gọi là chuỗi so khớp. Ta xét các hạt giống được biểu diễn bởi các ký tự 1 hoặc *, ký tự 1 biểu thị khớp còn ký tự * biểu thị khớp hoặc không khớp ở vị trí tướng ứng của hạt giống khi đối sánh với R.

Định nghĩa 5.1. (Tính hợp đúng được của hạt giống)

i. Với miền tương đồng biểu thị bởi chuỗi so khớp $R = r_1 r_2 .. r_N$ đã cho, hạt giống $s = s_1 s_2 .. s_l$ (l là độ dài hạt giống) được gọi là *hợp đúng được (hit)* R nếu tồn tại vị trí v của R sao cho với mọi $i = 1,2,..,l$ ta đều có:

$$r_{v+i-1} = \begin{cases} 1 & nếu\ s_i = 1 \\ 0\ hoặc\ 1 & nếu\ s_i = * \end{cases} \quad (5.1)$$

Số lượng ký tự 1 trong hạt giống s gọi là *trọng số* của nó.

ii. Một tập hạt giống S gồm k hạt giống có cùng trọng số được gọi là *hợp đúng được* R nếu tồn tại một hạt giống hợp đúng được R.

Ví dụ: Tập hạt giống {11*1, 1*111} có thể hợp đúng được các chuỗi {100**1101**00001, 10000**1011**1001, 10000**11111**001, **1101**00**10111**001}.

Bây giờ ta xét chuỗi so khớp của hai chuỗi sinh học có xác suất khớp ở mỗi vị trí của chuỗi như nhau và bằng p, tức là các ký tự r_i ở mỗi vị trí i của chuỗi $R = r_1 r_2 .. r_N$ đều nhận giá trị 1 với xác suất p:

$$\mathrm{P}(r_i = 1) = p \; \forall i \leq N, \tag{5.2}$$

khi đó p được gọi là mức tương tự (similarity level) của R.

Bài toán tìm tập giống tối ưu như sau: Với chuỗi so khớp R có mức tương tự p đã cho, tìm một tập S gồm k hạt giống có cùng trọng số w sao cho xác suất mà tập S hợp đúng được chuỗi này lớn nhất.

Chú ý rằng, các hạt giống trong tập S nói chung có độ dài khác nhau nhưng có cùng trọng số. Xác suất để tập hạt giống S hợp đúng được chuỗi R gọi là độ nhạy của S. Bài toán tìm tập hạt giống tối ưu được xét trong hai trường hợp: độ dài các hạt giống đã biết hoặc chưa biết. Trong cả hai trường hợp, Li và các cộng sự [47] đã chứng minh các bài toán đều thuộc lớp NP-khó, đặc biệt, ngay cả việc tính độ nhạy (hàm mục tiêu) cũng thuộc lớp NP-khó. Trong [47] cũng đề xuất một thuật toán để tính độ nhạy cảm cho tập hạt giống, nó được dùng trong SpEEDfast và AcoSeeD.

5.1.2. Các cách tiếp cận hiện nay

Bài toán tìm tập hạt giống tối ưu đã có nhiều thuật toán giải được công bố. Trong đó phải kể đến thuật toán heuristic tham ăn do Li và cộng sự đề xuất năm 2004 [47]. Ý tưởng thuật toán là xuất phát từ tập rỗng, mỗi lần chọn một hạt giống cho vào tập sao cho tập hạt giống hiện tại có độ nhạy lớn nhất, thực hiện cho tới khi chọn đủ k hạt giống. Thuật toán leo đồi do Sun và Buhler đề xuất năm 2005 [67] thực hiện bằng cách xuất phát từ một tập hạt giống ngẫu nhiên, lặp lại bước cải tiến lời giải nhờ tráo

đổi 2 vị trí trong một hạt giống của tập. Tuy nhiên, do đặc điểm của bài toán là thời gian tính độ nhạy của tập hạt giống lớn [47] nên Ilie và các cộng sự [49] đã đề xuất thuật toán leo đồi sử dụng cách tính hàm mục tiêu xấp xỉ nhanh OC (Overlap Complexity) [49] thay cho tính độ nhạy trong mỗi bước tính toán. Năm 2011, Ilie và cộng sự công bố phần mềm SpEED [50] và được cho là phần mềm thể hiện thuật toán tìm tập hạt giống tốt nhất hiện nay. Phiên bản mới của phần mềm này là SpEEDfast được công bố năm 2012 [51]. Thuật toán leo đồi dùng trong SpEED và SpEEDfast được mô tả trong hình 5.1. Nhược điểm chính của thuật toán SpEED (hay SpEEDfast) là sử dụng thuật toán leo đồi đơn giản và chưa tận dụng được thông tin từ hàm mục tiêu chính (hàm tính độ nhạy) trong tìm kiếm. Trong AcoSeeD, sử dụng kết hợp hai hàm, hàm OC sẽ được dùng để làm hàm mục tiêu khi cải tiến lời giải bằng tìm kiếm cục bộ, hàm tính độ nhạy được dùng để lựa chọn lời giải khi cập nhật mùi.

Procedure Thuật toán leo đồi tìm tập hạt giống tối ưu;
Dữ liệu vào: w, k, p, N, độ dài các hạt giống nếu đã biết.
Kết quả ra: Tập hạt giống và độ nhạy
Begin
    while (chưa kết thúc) **do**
        $S \leftarrow$ Lời giải ngẫu nhiên;
        Cải tiến lời giải S bằng tìm kiếm cục bộ nhờ hàm mục tiêu OC;
        Tính độ nhạy của S và cập nhật lời giải tốt nhất;
    end-while
    Đưa ra lời giải tốt nhất và độ nhạy;
End;

Hình 5.1: Thuật toán leo đồi dùng trong SpEED và SpEEDfast

5.2. Thuật toán AcoSeeD giải bài toán tìm tập hạt giống tối ưu

5.2.1. Mô tả thuật toán

Thuật toán AcoSeeD áp dụng phương pháp ACO theo lược đồ có sử dụng tìm kiếm cục bộ cho mỗi lời giải tìm được ở mỗi bước lặp. Vì thuật toán tính độ nhạy cảm tốn nhiều thời gian chạy nên nó chỉ dùng để đánh giá chất lượng các lời giải sau khi đã áp dụng tìm kiếm cục bộ, còn trong quá trình tìm kiếm cục bộ thì hàm OC được áp dụng để định hướng tìm kiếm (xem mục 5.2.4).

Thuật toán AcoSeeD giải bài toán tìm tập hạt giống tối ưu được xét trong hai trường hợp: độ dài các hạt giống đã biết hoặc chưa biết. Khi chưa biết độ dài của các hạt giống, ở mỗi bước lặp, mỗi kiến cần thực hiện thuật toán xác định chúng như mô tả trong mục 5.2.2 trước khi xây dựng lời giải trên đồ thị cấu trúc được trình bày trong mục 5.2.3.

Procedure AcoSeeD;
Dữ liệu vào: w, k, p, N, độ dài các hạt giống nếu đã biết.
Kết quả ra: tập hạt giống và độ nhạy;
Begin
  Khởi tạo tập A gồm N_a kiến, ma trận mùi, các tham số $\rho, \tau_{max}, \tau_{min}$
  while (chưa kết thúc) **do**
    for i =1 **to** N_a **do**
      Kiến thứ i **x**ác định độ dài các hạt giống; {mục 5.2.2}
      Kiến thứ i xây dựng tập hạt giống; {mục 5.2.3}
      Cải tiến lời giải bằng tìm kiến cục bộ nhờ hàm mục tiêu OC;{mục 5.2.4}
      Tính độ nhạy của tập hạt giống do kiến i xây dựng;
    end-for
    Cập nhật mùi dựa trên lời giải có độ nhạy lớn nhất tìm được;{mục 5.2.5}
    Cập nhật lời giải tốt nhất;
  end-while
  Đưa ra lời giải tốt nhất;
End;

Hình 5.2: Thuật toán AcoSeeD

Với các tham số w, k, p, N đã cho và số vòng lặp N_c hoặc thời gian chạy xác định trước, thuật toán AcoSeeD xác định tập hạt giống tối ưu cho trường hợp chưa biết

độ dài được mô tả trong hình 5.2. Trong trường hợp độ dài các hạt giống đã xác định thì thủ tục xác định độ dài các hạt giống được bỏ qua.

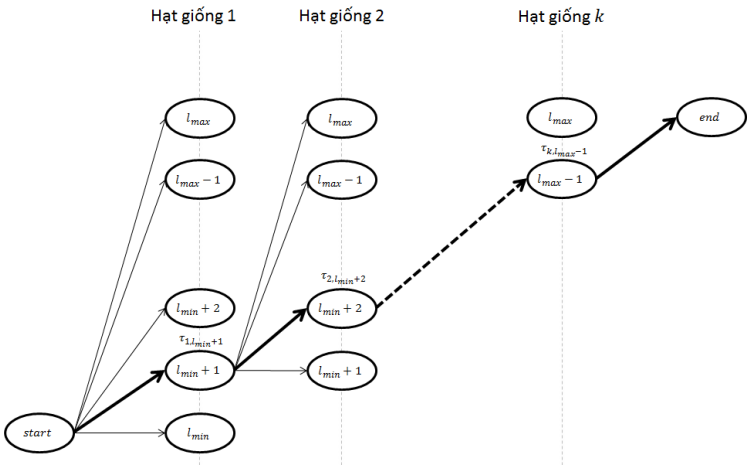
5.2.2. Thuật toán xác định độ dài các hạt giống

Trong trường hợp, độ dài các hạt giống chưa biết nhưng thuộc khoảng $[l_{min}, l_{max}]$ đã cho thì kiến phải thực hiện thủ tục xác định độ dài từng hạt giống trong tập nhờ đồ thị cấu trúc được mô tả trong hình 5.3. Ngoài hai đỉnh $start$ và end, đồ thị gồm k cột xếp từ phải sang trái, mỗi cột có $l_{max} - l_{min} + 1$ nút được gán nhãn từ l_{min} đến l_{max} biểu thị cho độ dài của hạt giống có thứ tự của cột tương ứng. Như vậy, các nút này được xếp thành $(l_{max} - l_{min} + 1)$ hàng và k cột. Ta xếp các hạt giống theo thứ tự tăng dần của độ dài, khi đó một đường xuất phát từ đỉnh $start$, đi qua k cột (chỉ sang ngang hoặc lên đỉnh trên ở cột tiếp theo) và kết thúc ở đỉnh end sẽ cho một phương án xác định độ dài của tập giống.

Thủ tục kiến xác định độ dài cho tập hạt giống thực hiện như sau. Tại đỉnh $start$ kiến lựa chọn ngẫu nhiên một trong các đỉnh $(1, l_{min}), (1, l_{min} + 1), .., (1, l_{max})$ với xác suất tương ứng theo công thức (5.3) với $i = 1$ để di chuyển. Ta dùng ký hiệu $length_i$ để chỉ nhãn của đỉnh tìm được ở bước i và ký hiệu (i, l) là nút ở cột i có nhãn l. Để các hạt giống có độ dài không giảm, tại bước thứ $i = 2, ... k$, kiến sẽ lựa chọn ngẫu nhiên các đỉnh ở cột tiếp theo có nhãn bằng hoặc lớn hơn nhãn nó đã chọn ở cột $(i - 1)$ tức là thuộc tập đỉnh $\{(i, length_{i-1}, .., (i, l_{max})\}$ để xác định độ dài cho hạt giống thứ i với xác suất cho bởi công thức (5.3):

$$p_i(l) = \frac{\tau_{i,l}\eta_{i,l}}{\sum_{h=v}^{l_{max}} \tau_{i,h}\eta_{i,h}} \tag{5.3}$$

trong đó $v = \begin{cases} l_{min} & nếu\ i = 1 \\ length_{i-1} & nếu\ i > 1 \end{cases}$, thông tin mùi $\tau_{i,l}$ thể hiện hạt giống thứ i ưu tiên lựa chọn độ dài l, còn thông tin heuristic $\eta_{i,h}$ được tính theo công thức (5.4).



Hình 5.3: Đồ thị cấu trúc để xác định độ dài các hạt giống

$$\eta_{i,h} = \begin{cases} 0.5 \; nếu \; (i > 1) và (h = length_{i-1}) \\ 1.0 \; nếu \; (i = 1) \; hoặc \; ((h > length_{i-1}) và (h < l_{max} - (k - i))) \\ 0.1 \; các \; trường \; hợp \; khác \end{cases} \quad (5.4)$$

Công thức (5.4) chỉ ra rằng, thông tin heuristic của việc lựa chọn độ dài cho hạt giống thứ i bằng độ dài hạt giống thứ $(i-1)$ có đánh giá thông tin heuristic bằng 0.5, vì độ dài các hạt giống là không giảm nên thông tin heuristic cho việc lựa chọn độ dài hạt giống thứ i lớn hơn độ dài hạt giống thứ $(i-1)$ và nhỏ hơn $l_{max} - (k-i)$ nên trường hợp này được đánh giá bằng 1.0 ($k-i$ hạt giống còn lại vẫn có cơ hội chọn độ dài khác nhau), các trường hợp khác thông tin heuristic được đánh giá bằng 0.1.

5.2.3. Thuật toán xây dựng các hạt giống

Đồ thị cấu trúc để xây dựng lời giải tìm tập giống được mô tả trong hình 5.4.A, gồm k hình chữ nhật kích thước $w \times (l_{max} - w)$. Kiến xây dựng lần lượt k hạt giống bằng cách xuất phát từ đỉnh start (đỉnh trái dưới của hình chữ nhật thứ nhất) có toạ độ (1,0,0), trong đó chỉ số thứ nhất là chỉ số thứ tự hình chữ nhật, chỉ số thứ hai là chỉ số

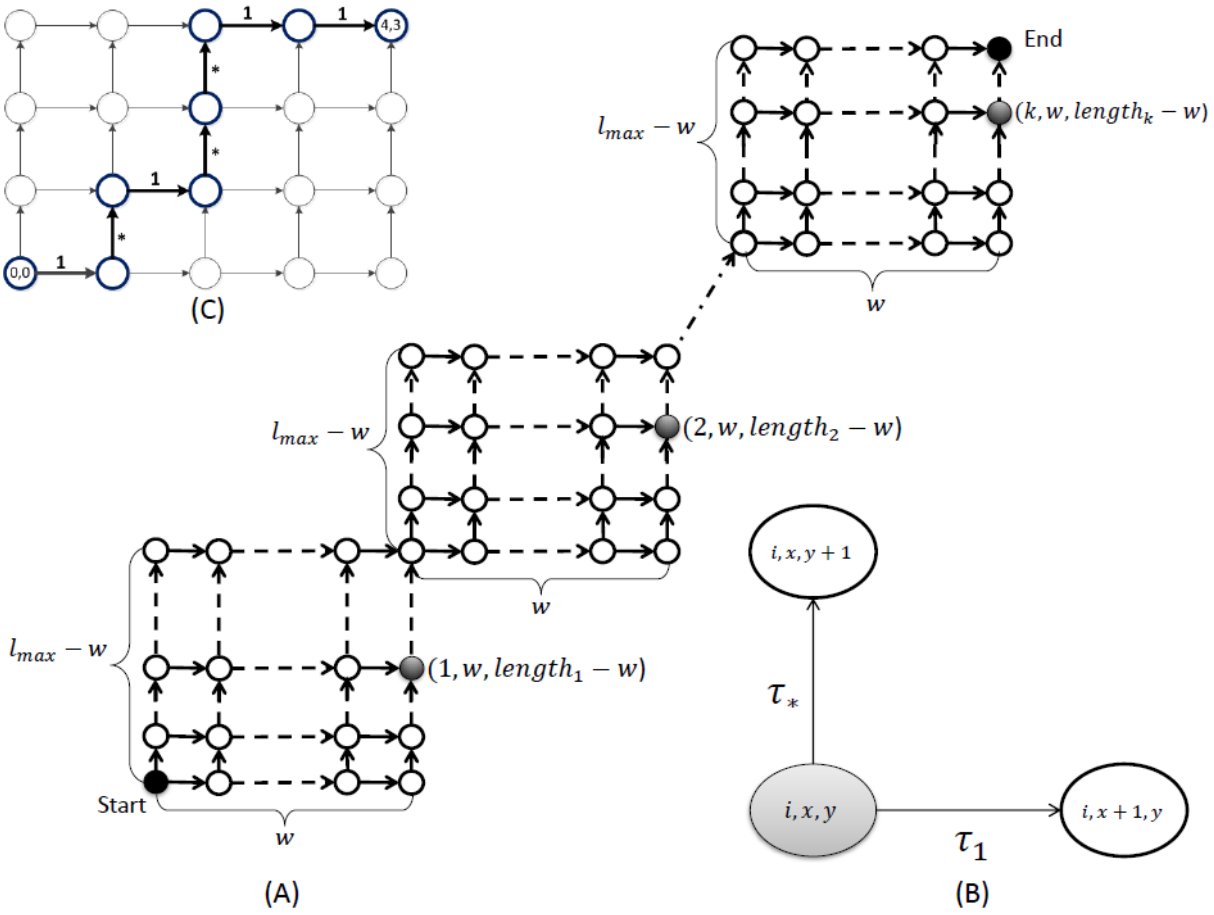
cột trên hình chữ nhật (đánh số từ trái qua phải), chỉ số thứ ba là chỉ số hàng trên hình chữ nhật (đánh số từ dưới lên trên) lần lượt di chuyển qua phải hoặc lên trên đến đỉnh End có toạ độ $(k, w, l_{max} - w)$ (đỉnh phải trên của hình chữ nhật thứ thứ k).

Hình 5.4.B cho thấy rằng khi kiến ở tọa độ (i, x, y) nó chỉ có thể di chuyển lên (chọn hướng τ_*) trên hoặc sang phải (chọn hướng τ_1). Trong hình chữ nhật thứ i kiến sẽ phải đi qua đỉnh $(i, w, length_i - w)$, trong đó $length_i$ là độ dài của hạt giống thứ i đã xác định trước hoặc là được xác định ở mục 5.2.2, sau đó di chuyển đến đỉnh đầu của hình chữ nhật tiếp theo. Với đồ thị cấu trúc và cách đi mô tả như trên, khi di chuyển qua hình chữ nhật thứ i thì kiến xây dựng được hạt giống thứ i có độ dài bằng $length_i$ và trọng số là w. Khi đi đến đỉnh End, kiến đã xây dựng xong được một lời giải gồm k hạt giống có trọng số w.

Vết mùi $\tau^i_{x,y,v}$ thể hiện thông tin học tăng cường cho sự ưu tiên lựa chọn đi theo hướng v khi kiến đang ở toạ độ (i, x, y). Khi đó, kiến chọn hướng di chuyển tiếp theo chỉ dựa trên thông tin này với xác suất cho bởi công thức (5.5):

$$P^i_{x,y}(v) = \frac{\tau^i_{x,y,v}}{\tau^i_{x,y,*} + \tau^i_{x,y,1}}, v \in \{*, 1\} \quad (5.5)$$

Hình 5.4.C minh họa một đường đi của kiến xây dựng một hạt giống có độ dài 7 và trọng số 4.



Hình 5.4: Đồ thị cấu trúc xây dựng các hạt giống.

Hình (A) Đồ thị cấu trúc xây dựng *k* hạt giống có trọng số *w*. Hình (B) Hướng kiến di chuyển tại mỗi đỉnh. (C) Ví dụ xây dựng một hạt giống trọng số 4 và độ dài 7.

5.2.4. Tìm kiếm cục bộ

Sau khi mỗi kiến xây dựng xong lời giải trong mỗi bước lặp. AcoSeeD dùng kỹ thuật tìm kiếm cục bộ để cải tiến lời giải. Như đã nói ở trên, do hàm tính độ nhạy của tập hạt giống có độ phức tạp lớn [47], nên trong quá trình tìm kiếm cục bộ, thuật toán AcoSeeD sử dụng kỹ thuật tìm kiếm cục bộ với hàm mục tiêu OC như Ilie đã thực hiện trong [49-51]. Cụ thể, từ tập hạt giống mà kiến xây dựng được, lặp lại bước cải tiến lời

giải nhờ tráo đổi 2 vị trí trong một hạt giống của tập, nếu hàm OC được cải thiện thì hạt giống này được thay bằng hạt giống mới.

5.2.5. Cập nhật mùi

Sau khi tất cả các kiến xây dựng xong lời giải và các lời giải được áp dụng kỹ thuật tìm kiếm cục bộ sử dụng hàm mục tiêu OC thì lời giải có độ nhạy lớn nhất sẽ được dùng để cập nhật mùi cho cả hai giai đoạn: giai đoạn xác định độ dài các hạt giống và giai đoạn xây dựng các hạt giống. AcoSeeD sử dụng cách cập nhật mùi mới SMMAS, cụ thể là:

Giai đoạn xác định độ dài các hạt giống được cập nhật mùi theo hai cận τ_{max}^1 và τ_{min}^1 (cận trên và cận dưới của vết mùi $\tau_{i,l}$) như sau:

$$\tau_{i,l} \leftarrow (1-\rho)\tau_{i,l} + \Delta_\tau \text{ trong đó } \Delta_\tau = \begin{cases} \rho\tau_{max}^1 \; nếu \; length_i = l \\ \rho\tau_{min}^1 \; nếu \; length_i \neq l \end{cases} \quad (5.6)$$

Giai đoạn xây dựng các hạt giống được cập nhật mùi theo hai cận τ_{max}^2 và τ_{min}^2 (cận trên và cận dưới của vết mùi $\tau_{x,y,v}^i$) như sau:

$$\tau_{x,y,v}^i \leftarrow (1-\rho)\tau_{x,y,v}^i + \Delta_\tau,$$

$$\text{trong đó } \Delta_\tau = \begin{cases} \rho\tau_{max}^2 \; nếu \; vị \; trí \; thứ \; (x+y+1) \; của \; hạt \; giống \; thứ \; i = v \\ \rho\tau_{min}^2 \; nếu \; vị \; trí \; thứ \; (x+y+1) \; của \; hạt \; giống \; thứ \; i \neq v \end{cases} \quad (5.7)$$

5.3. Kết quả thực nghiệm

Hiệu quả của AcoSeeD được so sánh bằng thực nghiệm với hai phương pháp tốt nhất hiện nay là SpEED [50] và SpEEDfast [51]. Để khách quan với SpEED và SpEEDfast, thực nghiệm so sánh trên các bộ dữ liệu đã công bố trong [50,51]. Trong các bài báo [50,51] khuyên rằng SpEED và SpEEDfast nên chạy với 5000 vòng lặp, tương ứng tạo ra 5000 lời giải. Do đó, trong AcoSeeD thiết đặt số kiến N_a=50 và số vòng lặp N_c=100 để cũng tạo ra 5000 lời giải. Như vậy, độ phức tạp của AcoSeeD có

cùng độ phức tạp với SpEED và SpEEDfast. Ngoài hai tham số N_a, N_c, thì AcoSeeD thiết đặt tham số khác như sau: $\rho = 0.3$, $\tau^1_{max} = 1.0$, $\tau^1_{min} = \frac{\tau^1_{max}}{2\times w\times k}$, $\tau^2_{max} = 1.0$, $\tau^2_{min} = \frac{\tau^2_{max}}{2\times w\times w\times k}$. Việc lựa chọn, tỉ lệ mùi cận trên và cận dưới của mỗi giai đoạn được chọn tỉ lệ với số đỉnh của đồ thị cấu trúc trong giai đoạn tương ứng.

Vì SpEEDfast [51] chỉ thực hiện với các bộ dữ liệu lớn nên với các bộ dữ liệu nhỏ và trung bình luận án chỉ so sánh với kết quả của SpEED [50]. Ngoài hai phương pháp SpEED và SpEEDfast, luận án cũng dẫn ra số liệu về kết quả chạy của hai phương pháp Mandala, Iedera đã công bố trong [50] để tham khảo.

5.3.1. Dữ liệu thực nghiệm

Thực nghiệm thực hiện trên các bộ dữ liệu mà SpEED [50] và SpEEDfast [51] đã chạy khi được công bố bao gồm các bộ:

- Bộ nhỏ với độ dài các hạt giống đã được xác định gồm 3 test;
- Bộ dữ liệu trung bình SHRiMP có số lượng hạt giống là 4 gồm 15 test;
- Bộ dữ liệu lớn gồm 3 bộ: bộ PatternHunterII có số lượng hạt giống là 16 gồm 3 test, bộ BFAST có số lượng hạt giống là 10 gồm 3 test và bộ MegaBLAST gồm 15 test.

5.3.2. Kết quả thực nghiệm trên bộ dữ liệu nhỏ với độ dài các hạt giống đã xác định

Kết quả thực nghiệm trên 3 test của bộ dữ liệu nhỏ được cho ở bảng 5.1. Cột k, w, N, p biểu thị thông tin về tham số của test tương ứng. Cột $length_1$, $length_2, .., length_k$ biểu thị thông tin độ dài các hạt giống. Cột Optimal biểu thị độ nhạy tối ưu, cột SpEED và AcoSeeD biểu thị kết quả tương ứng của phương pháp SpEED và AcoSeeD. Kết quả trong các phương pháp in đậm là kết quả ra

được tối ưu. Kết quả của SpEED là kết quả được công bố trong [50]. Ở đây không có kết quả của SpEEDfast vì trong [51] không công bố kết quả này.

Bảng 5.1: Kết quả thực nghiệm SpEED với AcoSeeD trên bộ dữ liệu nhỏ

k	w	N	p	$length_1, length_2, .., length_k$	Optimal	SpEED	AcoSeeD
2	14	35	0.88	17, 21	0.82991	0.82886	0.82886
3	10	35	0.78	12, 14, 16	0.81833	**0.81833**	**0.81833**
4	6	35	0.6	8,9,10,11	0.85026	0.84879	**0.85026**

Nhận xét: Kết quả thực nghiệm trong bảng 5.1 cho thấy SpEED có thể tìm được tối ưu 1 test trong 3 test, còn AcoSeeD có thể tìm tối ưu 2 test trong 3 test. Điều này được giải thích như sau, AcoSeeD đã kết hợp được cả hàm mục tiêu chính và hàm mục tiêu xấp xỉ OC. Trường hợp AcoSeeD không tìm được tối ưu là vì test quá nhỏ nên lời giải luôn hội tụ theo hàm mục tiêu OC.

5.3.3. Kết quả thực nghiệm trên bộ dữ liệu trung bình

Kết quả thực nghiệm trên bộ dữ liệu trung bình cho ở bảng 5.2. Cột w, p biểu thị thông tin về bộ test SHRiMP có $k = 4$, $N = 50$, còn các cột Mandala, Iedera, SpEED tương ứng biểu thị kết quả của phương pháp Mandala, Iedera, SpEED. Các kết quả này lấy trong [50]. Ở đây không có kết quả của SpEEDfast vì trong [51] không công bố kết quả này. Cột AcoSeeD gồm 3 cột nhỏ best, worst, avg tương ứng biểu thị kết quả tốt nhất, xấu nhất, trung bình trong 10 lần chạy. Kết quả tốt nhất trong các phương pháp được in đậm.

Nhận xét: Kết quả thực nghiệm trong bảng 5.2, cho thấy AcoSeeD tốt hơn Mandala, Iedera và SpEED trong tất cả các test. Trong thực nghiệm này, AcoSeeD chỉ chạy 10 lần chạy để lấy kết quả tốt nhất, xấu nhất, trung bình. Cần nhấn mạnh rằng, kết quả xấu nhất trong 10 lần chạy của AcoSeeD cũng đã tốt hơn các thuật toán khác.

Bảng 5.2: So sánh AcoSeeD với các phương pháp khác trên bộ dữ liệu trung bình

w	p	Mandala	Iedera	SpEED	AcoSeeD		
					best	worst	avg
10	0.75	90.6608	90.6802	*90.91*	**90.9757**	90.9104	90.9513
	0.8	97.7316	97.7586	*97.834*	**97.8584**	97.8467	97.8521
	0.85	99.7283	99.7437	*99.757*	**99.7624**	99.7599	99.7614
11	0.75	83.0512	83.2413	*83.379*	**83.5349**	83.4207	83.4728
	0.8	94.7845	94.935	*94.986*	**95.0636**	95.0144	95.037
	0.85	99.1929	99.2189	*99.243*	**99.2498**	99.2451	99.2478
12	0.8	90.258	90.3934	*90.575*	**90.6576**	90.6147	90.6328
	0.85	98.0786	98.0781	*98.159*	**98.1786**	98.1682	98.1766
	0.9	99.8633	99.8773	*99.882*	**99.8866**	99.8845	99.8853
16	0.85	84.3838	84.5795	*84.821*	**85.0328**	84.915	84.9829
	0.9	97.3023	97.2806	*97.432*	**97.483**	97.464	97.4712
	0.95	99.9287	99.9331	*99.939*	**99.9429**	99.9414	99.9419
18	0.85	72.1954	72.1695	*73.166*	**73.3357**	73.2432	73.27
	0.9	93.0855	93.0442	*93.712*	**93.7912**	93.7597	93.7778
	0.95	99.6603	99.669	*99.75*	**99.7617**	99.7575	99.7599

5.3.4. Kết quả thực nghiệm trên bộ dữ liệu lớn

Kết quả thực nghiệm trên hai bộ dữ liệu lớn PatternHunterII và BFAST được cho trong bảng 5.3, còn kết quả trên bộ dữ liệu MegaBFAST được cho trong bảng 5.4. Ý nghĩa biểu thị trong các cột như đã nêu ở trên. Các kết quả của SpEED và SpEEDfast được lấy trong [51].

Bảng 5.3: Kết quả thực nghiệm so sánh AcoSeeD với các phương pháp trên bộ dữ liệu lớn PatternHunterII và BFAST

w	p	Mandala	Iedera	SpEED	SpEEDfast	AcoSeeD		
						best	worst	avg
PatternHunterII: k =16 hạt giống, N = 64								
11	0.7	92.3811	92.0708	93.2526	*93.3406*	**93.3608**	93.3154	93.3576
	0.75	98.432	98.3391	98.6882	*98.7156*	**98.7346**	98.7101	98.7205
	0.8	99.8448	99.8366	99.882	*99.8859*	**99.8875**	99.884	99.8852
BFAST: k =10 hạt giống, N = 50								
22	0.85	Không chạy được	60.1535	60.8127	*60.9329*	**61.0258**	60.927	60.9797
	0.9	Không chạy được	87.9894	88.5969	*88.7120*	**88.8504**	88.8102	88.8287
	0.95	Không chạy được	99.2196	99.3659	*99.3959*	**99.4046**	99.3988	99.4025

Bảng 5.4: So sánh AcoSeeD với SpEEDfast trên bộ dữ liệu lớn MegaBFAST

w	N	p	Method	$k = 1$	$k = 2$	$k = 4$	$k = 8$	$k = 16$
28	100	0.9	SpEEDfast	69.3241	79.6629	87.5674	92.7762	95.9170
			AcoSeeD	**69.4522**	**80.6561**	**88.5098**	**93.5635**	**96.5560**
28	150	0.9	SpEEDfast	87.6426	93.4308	97.0118	98.7430	99.5137
			AcoSeeD	**87.6571**	**94.0766**	**97.5303**	**99.0520**	**99.6605**
28	200	0.9	SpEEDfast	**94.9876**	97.8936	99.2937	99.7877	99.9409
			AcoSeeD	94.9606	**98.2302**	**99.4766**	**99.8588**	**99.9648**

Nhận xét: Kết quả thực nghiệm trong các bảng trên cho thấy AcoSeeD tốt hơn SpEED và SpEEDfast trong hầu hết các trường hợp (trừ một test $w = 28, N = 200,\ p = 0.9$,

$k = 1$). AcoSeeD đã tìm được các tập hạt giống mới có độ nhạy cao hơn SpEEDfast tìm được.

5.4. Kết luận chương

Trên đây, luận án đề xuất thuật toán AcoSeeD tìm tập hạt giống tối ưu dùng trong tìm kiếm tương đồng của các chuỗi sinh học. Kết quả thực nghiệm cho thấy AcoSeeD tốt hơn phương pháp tốt nhất hiện nay SpEED, SpEEDfast. Nói riêng AcoSeeD đã tìm được các tập hạt giống mới có độ nhạy cao hơn hẳn tập hạt giống mà SpEEDfast tìm được công bố năm 2012. Thông qua thuật toán AcoSeeD, chúng tôi cũng giới thiệu cách sử dụng tìm kiếm cục bộ bằng hàm mục tiêu xấp xỉ nhanh thay cho hàm mục tiêu chính trong ACO. Nhờ cách kết hợp linh hoạt này mà thời gian chạy cho tìm kiếm cục bộ giảm đáng kể mà vẫn cho kết quả tốt.

Chương 6. ỨNG DỤNG PHƯƠNG PHÁP ACO CẢI TIẾN HIỆU QUẢ DỰ ĐOÁN HOẠT ĐỘNG ĐIỀU TIẾT GEN

Kể từ khi Watson và Crick phát hiện cấu trúc của DNA (DeDeoxyribonucleic Acid), đến nay, người ta đã có những thành tựu quan trọng trong di truyền học. Tuy nhiên, việc giải mã hoạt động điều tiết gen vẫn còn nhiều thách thức. Trong chương này, luận án đề xuất thuật toán ACO [23] và thuật toán di truyền [26] để tìm tham số cho SVM (*Support Vector Machine - SVM*) được dùng để dự đoán điều tiết gen từ mối liên kết các yếu tố phiên mã (*Transcriptional Factors - TFs*). Thực nghiệm trên dữ liệu thực chỉ ra rằng các tham số cho SVM tìm được bằng thuật toán ACO này đã cải tiến được 10% khả năng dự đoán so với cách tiếp cận tìm kiếm dựa trên lưới truyền thống và tốt hơn thuật toán di truyền.

6.1. Bài toán dự đoán hoạt động điều tiết gen

Hiểu cơ chế điều chỉnh biểu hiện gen qua các yếu tố phiên mã (*Transcription Factors-TFs*) là nhiệm vụ trung tâm của sinh học phân tử. Người ta biết rằng các trạng thái biểu hiện gen được thành lập thông qua sự tích hợp của mạng tín hiệu và phiên mã hội tụ trên các thành phần tăng cường, còn được gọi là mô-đun điều tiết (Cis-Regulatory Module - CRM)[64]. Các mô-đun điều tiết này là các đoạn DNA, nó liên kết các yếu tố phiên mã để điều tiết biểu diễn gen liên quan. Mỗi mô-đun có thể điều tiết một hoặc nhiều gen.

Ở mức độ bộ gen hoặc các mô-đun điều tiết gen cụ thể, người ta đã có phương pháp nhận dạng các hoạt động điều tiết này. Tuy nhiên, ở mức độ toàn cảnh, hiện tại vẫn là bài toán mở. Gần đây, Zinzen và các cộng sự [71] đã giới thiệu một mô hình dự báo điều tiết trên ruồi dấm Drosophila.

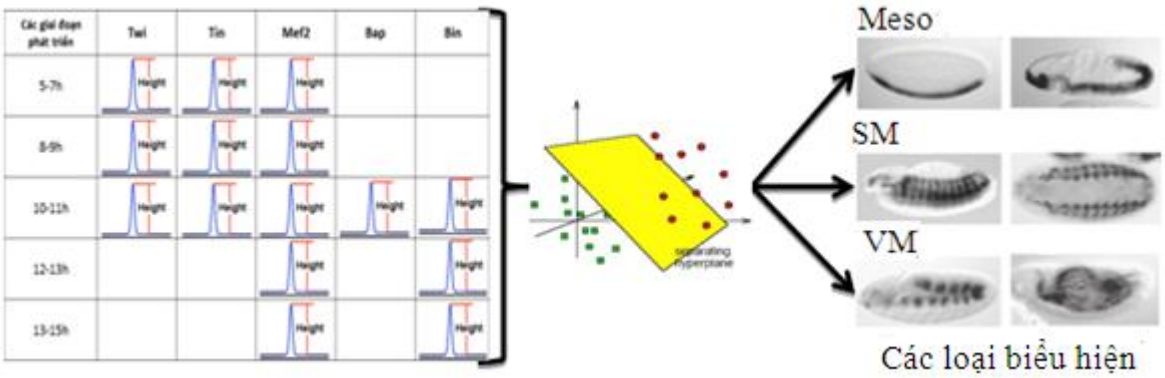
6.1.1. Mối liên kết yếu tố phiên mã trong phát triển phôi của ruồi giấm Drosophila

Ruồi giấm Drosophila là một mẫu sinh vật được dùng để nghiên cứu sự phát triển của phôi thai trong sinh học. Zinzen và các cộng sự [71] đề xuất sử dụng phương pháp ChIP (*Chromatin Immunoprecipitation*) để thu được dữ liệu của yếu tố phiên mã quan trọng của ruồi giấm Drosophila (Twist, TinMan, Mef2, Bagpipe và Biniou) tại 5 thời điểm trong quá trình phát triển phôi. Các dữ liệu được chuẩn hóa và trích được 15 đặc trưng biểu thị tính tích cực của CRM về điều chỉnh gen như được minh họa trong hình 5.1. Sau đó Zinzen dùng cơ sở dữ liệu gồm 310 CRM lấy từ cơ sở dữ liệu REDFly [39] để nghiên cứu sự ảnh hưởng của mối liên kết yếu tố phiên mã trên biểu hiện gen.

Mỗi CRM được xếp vào một nhóm biểu hiện hoạt động: Mesoderm – trung bì, Somatic muscle – cơ soma, Visceral muscle – cơ nội tạng, lần lượt gọi là Meso, SM và VM). Ngoài ra, có một số CRM thuộc loại hỗn hợp như trung bì và cơ soma (gọi là Meso_SM) hoặc cơ soma và cơ nội tạng (gọi là SM_VM). Như vậy, mỗi CMR có thể được xếp vào một trong năm nhóm biểu hiện:

- Meso
- SM
- VM
- Meso_SM
- SM_VM.

Cơ sở dữ liệu đã nêu được dùng để huấn luyện bộ nhận dạng theo phương pháp SVM để dự đoán hoạt động điều tiết gen thông qua xác định nhãn cho các CMR dựa trên các đặc trưng đã biết.



Hình 6.1: Dự đoán hoạt động điều tiết gen dựa trên liên kết phiên mã

Để rõ hơn cách tiếp cận mới, luận án giới thiệu tóm tắt phương pháp của Zinzen và cộng sự đã sử dụng SVM cho bài toán dự đoán điều tiết này.

6.1.2. Dự đoán hoạt động điều tiết gen bằng phương pháp học máy SVM

Phương pháp học máy SVM là phương pháp học có giám sát để nhận dạng mẫu (xem [3,6]). Trong phương pháp này, dựa trên tập dữ liệu huấn luyện đã có:

$$D = \{x^k, y^k\}_{k=1}^m, \tag{6.1}$$

trong đó $x^k = (x_1^k, \dots, x_n^k)$ là các đối tượng có n đặc trưng và y^k là nhãn lớp của nó. Người ta sẽ phân lớp theo từng loại nhãn bằng hàm phân biệt tuyến tính:

$$d(x) = \sum_{i=1}^n w_i x_i + w_0 \tag{6.2}$$

sao cho nó xác định lề cực đại.

Khi tập mẫu không tách được tuyến tính, người ta dùng biên mềm. Các hệ số w và w_0 của hàm phân biệt có thể xác định nhờ giải bài toán quy hoạch:

Cực tiểu hàm:

$$J(w, w_0, \xi) = \frac{\|w\|^2}{2} + C \sum_t \xi^t \tag{6.3a}$$

với các ràng buộc:

$$y^t(w^T x^t + w_0) \geq 1 - \xi^t \; \forall (x^t, y^t) \in D, \tag{6.3b}$$

$$\xi^t \geq 0 \; \forall t = 1, .., m, \tag{6.3c}$$

trong đó C là hằng số dương biểu thị mức phạt các điểm phân lớp sai, trường hợp tách được tuyến tính ứng với $C \to \infty$.

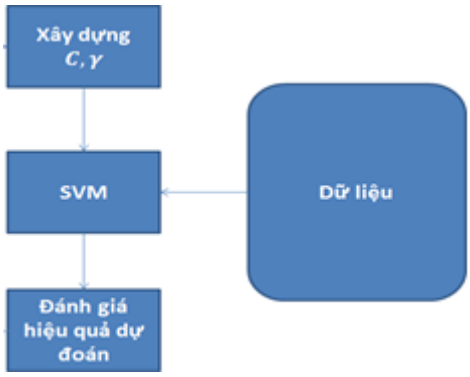
Để tăng chất lượng nhận dạng, người ta dùng ánh xạ ϕ: $R^n \to R^N$ nhúng không gian đặc trưng lên không gian có số chiều lớn hơn. Ánh xạ $\phi = (\phi_1, \ldots, \phi_N)$ thường được xác định qua hàm nhân:

$$K(u, v) = e^{-\gamma \|u - v\|^2} \tag{6.4a}$$

và: $$\phi_i(x) = K(u, v^i) \tag{6.4b}$$

trong đó v^i được chọn trước.

Zinzen và các cộng sự [71] chọn các hằng số C, γ trên một lưới $k \times k$ sao cho kết quả nhận dạng các lớp có sai số nhỏ nhất. Với mỗi cặp giá trị (C, γ) trên lưới được chọn, tập huấn luyện được dùng để huấn luyện các bộ phân lớp (dùng SVM ở [73]) theo phương pháp $k - folds$ với $k = 1$ (còn gọi là phương pháp Leave One Out). Với mỗi đối tượng bỏ ra, người ta huấn luyện bộ phân lớp dựa trên tập còn lại và nhận dạng cho đối tượng này để kiểm tra. Sau khi xoay vòng hết thì người ta đánh giá tỷ lệ sai để xác định sai số cho cặp giá trị tham số (C, γ) tương ứng. Cặp giá trị với sai số nhỏ nhất được dùng để huấn luyện bộ nhận dạng. Phần mềm dự đoán theo phương pháp này có ở [78] và được dùng để so sánh với phương pháp mới. Sơ đồ đánh giá hiệu quả dự đoán của tham số cho SVM này được minh họa trong hình 6.2.



Hình 6.2: Sơ đồ đánh giá hiệu quả tham số SVM

Mặc dù phương pháp tìm kiếm tham số trên lưới là thông dụng trong Y-Sinh nhưng nhược điểm cơ bản của nó là không thể tìm kiếm trên lưới dày (bước lưới nhỏ) vì vậy, khi đó không cải thiện được lời giải. Để tăng chất lượng dự đoán, luận án đề xuất ứng dụng thuật toán di truyền [26] và phương pháp ACO [23] để xác định tham số SVM.

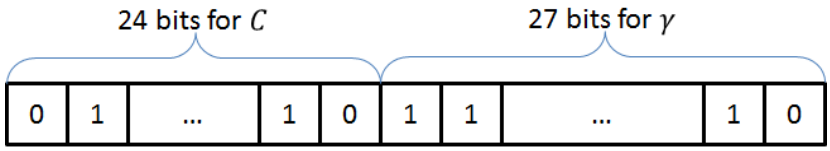
6.2. Thuật toán di truyền tìm tham số cho SVM dùng trong dự đoán hoạt động điều tiết gen

Luận án sử dụng phần mềm SVM [73] để phân biệt lần lượt từng lớp một, như cách làm của Zinzen [71]. Như vậy, nhãn của các mẫu dữ liệu có dạng nhị phân, thuộc lớp thì có nhãn bằng 1, ngược lại nhãn bằng -1.

Trong [26] luận án đã đề xuất thuật toán di truyền để tìm tham số cho SVM dùng trong bài toán dự đoán hoạt động điều tiết gen. Thuật toán di truyền đã được nói rõ trong [57]. Luận án xác định hàm mục tiêu, mã hóa tham số cần tìm, xác định các toán tử đột biến và tương giao chéo rồi dùng gói phần mềm dựa trên ngôn ngữ R ở địa chỉ [74] để tìm tham số tốt nhất.

6.2.1. Mã hóa các tham số cần tìm

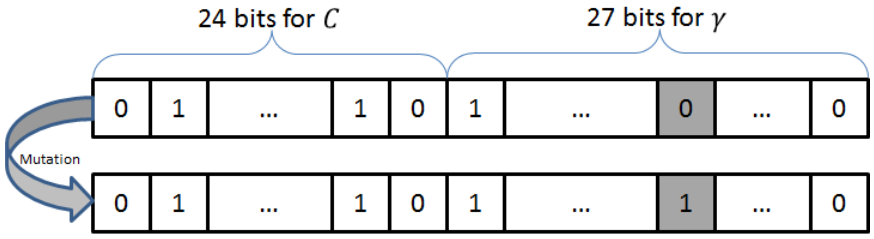
Luận án đề xuất cách mã hoá nhị phân 51 bit để biểu diễn hai tham số C và γ. Tham số C nhận giá trị từ 10^{-2} đến 10^{5} được biểu diễn bằng một dãy 24 bit, và γ nhận giá trị 10^{-6} đến 10^{2} được biểu diễn bằng một dãy 27 bit. Như vậy, việc tìm C và γ tương ứng với việc tìm một dãy 51 bit, trong đó 24 bit đầu tiên là mô tả cho C, 27 bit tiếp theo mô tả cho γ như mô tả trong hình 6.3.



Hình 6.3: Một nhiễm sắc thể biểu diễn C và γ

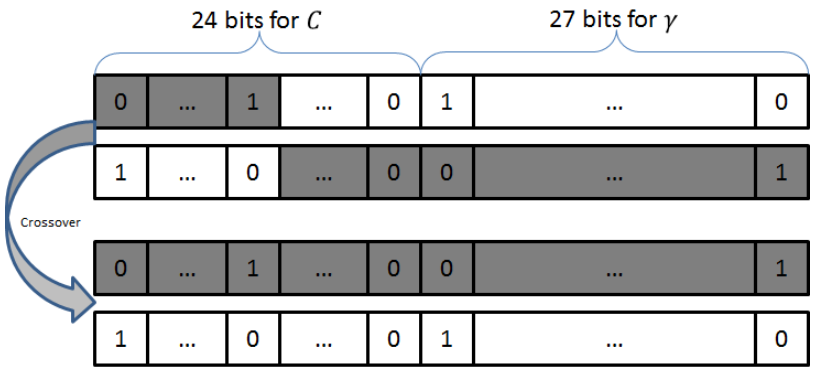
6.2.2. Các phép toán di truyền

Phép toán đột biến: Thực hiện đột biến theo di truyền cổ điển, với mỗi gen đột biến sẽ chuyển từ 0 thành 1 hoặc ngược lại như minh họa trong hình 6.4



Hình 6.4: Gen ở vị trí in đậm đột biến 0 thành 1 hoặc ngược lại

Phép toán tương giao chéo: Như trong thuật toán di truyền cổ điển, phép tương giáo chéo chọn ngẫu nhiên một vị trí của cặp nhiễm sắc thể rồi trao đổi các thành phần cho nhau để được cặp nhiễm sắc thể mới như minh họa trong hình 6.5.



Hình 6.5: Minh họa phép toán tương giao chéo của cặp nhiễm sắc thể

6.2.3. Lược đồ thuật toán di truyền

Hàm mục tiêu cho thuật toán là cực tiểu sai số nhận dạng bằng SVM theo tham số C, γ như đã nói ở mục 6.1. Khi đó, thủ tục của thuật toán di truyền tìm các tham số cho SVM (GASVM) cho dự đoán hoạt động điều tiết gen được đặc tả trong hình 6.6. Với cách mã hóa tham số, hàm mục tiêu và các phép toán di truyền đã nêu, phần mềm dựa trên ngôn ngữ R ở địa chỉ [74] được dùng để tìm tham số tốt nhất với xác suất đột biến và tương giao chéo lấy ngầm định theo gói phần mềm này. Kết quả thực nghiệm của thuật toán và so sánh với phương pháp của Zinzen sẽ được trình bày trong mục 6.4.

```
Procedure GASVM;
Dữ liệu vào: Dữ liệu huấn luyện và dữ liệu test;
Kết quả ra: Tham số C và γ;
Begin
        Khởi tạo quần thể P;
        while (chưa kết thúc) do
                Chọn lọc Q từ P;
                Tạo P từ Q bằng các phép toán di truyền;
                Đánh giá P;
                Cập nhật lời giải tốt nhất;
        end-while;
        Đưa ra lời giải tốt nhất;
End;
```

Hình 6.6: Thuật toán GASVM

6.3. Thuật toán tối ưu đàn kiến tìm tham số cho SVM dùng trong dự đoán hoạt động điều tiết gen

Mặc dù thuật toán di truyền có hiệu quả hơn phương pháp tìm kiếm lưới, tuy nhiên nó có nhược điểm là các lời giải tạo nên ở các bước lặp sau trùng lặp với lời giải ở bước trước khá nhiều. Vì vậy, luận án thử nghiệm xây dựng thuật toán ACOSVM cho bài toán dự đoán này.

Lược đồ của thuật toán tuân theo thủ tục đã mô tả trong chương 2, cụ thể được mô tả trong hình 6.7.

```
Procedure ACOSVM;
Dữ liệu vào: Dữ liệu huấn luyện và dữ liệu test;
Kết quả ra: Tham số C và γ;
Begin
    Khởi tạo tập A gồm N_a kiến, ma trận mùi, các tham số;
    while (chưa kết thúc) do
        for each a ∈ A do
            Kiến a xây dựng lời giải;{là một xâu nhị phân gồm 51 bit}
        end-for
        Cập nhật mùi;
        Cập nhật lời giải tốt nhất;
    end-while
    Ghi lời giải tốt nhất;
End;
```

Hình 6.7: Thuật toán ACOSVM

Dưới đây mô tả rõ đồ thị cấu trúc, ma trận mùi và thủ tục xây dựng lời giải.

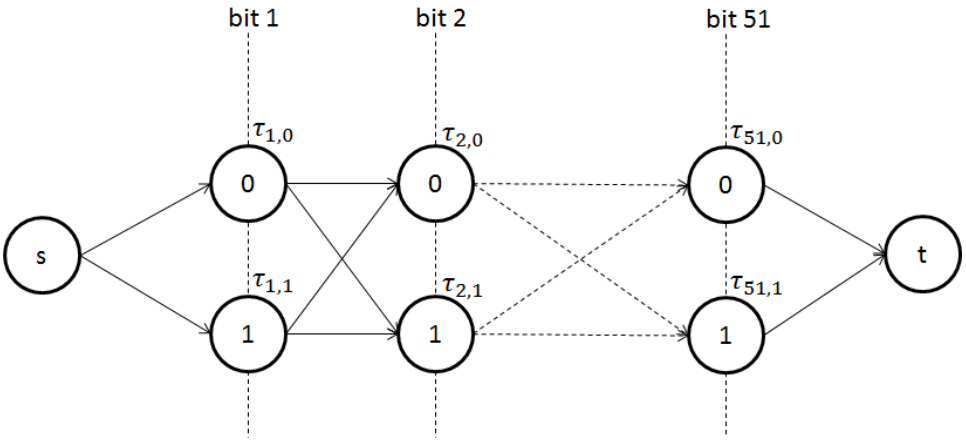
6.3.1. Đồ thị cấu trúc và ma trận mùi

Tương tự như trong thuật toán di truyền, một lời giải do kiến xây dựng cũng là một xâu nhị phân gồm 51 bit. Cụ thể là:

- Tham số C có giá trị từ 10^{-2} đến 10^{5} được biểu diễn bằng một dãy 24 bit
- Tham số γ có giá trị 10^{-6} đến 10^{2} được biểu diễn bằng một dãy 27 bit.

Như vậy việc tìm hai tham số C và γ tương ứng với việc tìm một dãy 51 bit, trong đó 24 bit đầu tiên là mô tả cho C, 27 bit tiếp theo mô tả cho γ.

Đồ thị cấu trúc có đỉnh xuất phát s, đỉnh kết thúc t và 51 tầng mỗi tầng gồm 2 đỉnh có nhãn 0 hoặc 1 như trong hình 6.8. Vết mùi được đặt trên đỉnh của đồ thị, cụ thể: mùi $\eta_{i,0}$ và $\eta_{i,1}$ tương ứng thể hiện sự ưa thích của kiến ở bit thứ i chọn giá trị 0 hoặc 1, còn thông tin heuristic như nhau và đều bằng 1. Để xây dựng lời giải là xâu nhị phân gồm 51 bit, kiến xây dựng hành trình trên đồ thị từ đỉnh xuất phát s qua 51 đỉnh được chọn ở 51 tầng và đến đỉnh kết thúc t.



Hình 6.8: Đồ thị cấu trúc

6.3.2. Thủ tục xây dựng lời giải của kiến và cập nhật mùi

Mỗi kiến sẽ xuất phát ở đỉnh s và kết thúc tại đỉnh t để xây dựng một cặp tham số C và γ theo quá trình sau. Nếu bước thứ i kiến lựa chọn đỉnh ở hàng trên tức là bit thứ i chọn giá trị 0, còn nếu kiến lựa chọn đỉnh hàng dưới tức là bit thứ i chọn giá trị 1. Kiến k ở bước thứ i sẽ lựa chọn đỉnh tiếp theo có giá trị nhãn là v theo xác suất:

$$p_i^k(v) = \frac{\tau_{i,v}}{\tau_{i,0}+\tau_{i,1}} \quad (6.5)$$

Sau khi kiến xây dựng xong lời giải là một dãy nhị phân gồm 51 bit. Ta tiến hành giải mã xác định C và γ. Tiếp theo gọi SVM sử dụng tham số C và γ với giá trị vừa tìm được. Hiệu quả của dự đoán sẽ đánh giá độ tốt của lời giải do kiến xây dựng. Lời giải của kiến tốt nhất sẽ được dùng để cập nhật mùi theo SMMAS như sau:

$$\tau_{i,v} = (1-\rho) + \begin{cases} \rho\tau_{max} \; nếu \; (i,v) \in s^{i-best} \\ \rho\tau_{min} \; nếu \; (i,v) \notin s^{i-best} \end{cases} \quad (6.6)$$

Hiệu quả của thuật toán này được so sánh với các thuật toán di truyền và phương pháp lưới bằng thực nghiệm.

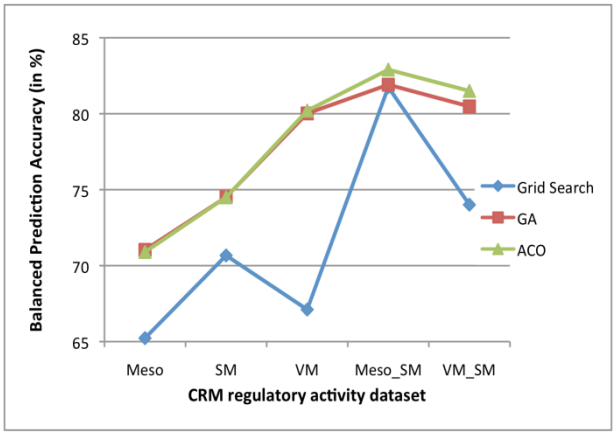
6.4. Kết quả thực nghiệm

Luận án thực nghiệm trên tập dữ liệu bao gồm 310 CRM đã biết hoạt động điều tiết lấy từ cơ sở dữ liệu (REDFly, [39]). Đối với mỗi dữ liệu, tính chính xác của dự báo được đo dựa trên tỷ lệ dự đoán đúng.

Thực nghiệm chạy cho cả ba phương pháp: Zinzen, GASVM và ACOSVM trên cùng một máy và cùng dùng gói phần mềm dựa trên ngôn ngữ R cho SVM. Các thuật toán GASVM và ACOSVM được chạy 10 lần để lấy kết quả trung bình của 10 lần chạy cho so sánh, phần mềm dự đoán theo phương pháp Zinzen lấy ở [78]. Kết quả thực nghiệm cho ở bảng 6.1 và thể hiện trực quan qua biểu đồ ở hình 6.9.

Bảng 6.1: Kết quả thực nghiệm so sánh 3 phương pháp

Dữ liệu	Grid Search	GA	ACO
Meso	65.23	**71.04**	70.9
SM	70.67	**74.51**	74.5
VM	67.11	80.02	**80.2**
Meso_SM	81.75	81.91	**82.9**
VM_SM	74.01	80.48	**81.5**



Hình 6.9: Biểu đồ so sánh kết quả dự đoán đúng giữa ba phương pháp

Kết quả thực nghiệm cho thấy cả hai phương pháp tiếp cận metaheuristic mới đề xuất (GASVM và ACOSVM) tốt hơn các kết quả của phương pháp tìm kiếm dựa trên lưới của Zinzen trong [71] về độ chính xác. Hầu hết các trường hợp kết quả đạt được độ chính xác tốt hơn 5-10%, ngoại trừ Meso_SM chỉ có tốt hơn 1%. Cả hai GA và ACO đã đạt được kết quả rất giống nhau trong 3 trên 5 bộ dữ liệu loại biểu hiện là duy nhất (Meso, SM hoặc VM). Trong hai trường hợp hỗn hợp, ACO tốt hơn so với GA.

6.5. Kết luận chương

Dự đoán hoạt động điều tiết gen là một trong các bước quan trọng để hiểu các yếu tố ảnh hưởng tới điều tiết gen trong sinh học. Các công nghệ giải mã hiện nay cho phép chúng ta giải quyết vấn đề này một cách hiệu quả cho từng bộ gen hoặc các gen riêng rẽ nhưng một bức tranh toàn cảnh vẫn còn là thách thức. Zinzen và cộng sự đã đề xuất sử dụng phương pháp ChIP để nghiên cứu các yếu tố phiên mã quan trọng của ruồi giấm Drosophila. Phương pháp này áp dụng tìm kiếm trên lưới để xác định tham số cho bộ nhận dạng SVM cho kết quả hứa hẹn.

Tuy nhiên, việc tìm kiếm lưới bị hạn chế do bùng nổ không gian tìm kiếm khi lấy lưới dày. Hai thuật toán GASVM và ACOSVM mới đề xuất cải thiện đáng kể hiệu quả dự đoán hoạt động điều tiết gen dựa trên SVM đã nêu của Zinzen và cộng sự .

KẾT LUẬN

Các bài toán TƯTH khó có nhiều ứng dụng quan trọng trong thực tiễn, đặc biệt là trong các bài toán sinh học. Phương pháp ACO kết hợp thông tin heuristic và thông tin học tăng cường nhờ mô phỏng hoạt động của đàn kiến có các ưu điểm nổi trội sau:

1) Việc tìm kiếm ngẫu nhiên dựa trên các thông tin heuristic 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.

Thực nghiệm đã chứng tỏ khả năng nổi trội của phương pháp ACO trong ứng dụng cho nhiều bài toán và phương pháp này đang được sử dụng rộng rãi.

Khi dùng phương pháp ACO, quy tắc cập nhật mùi đóng vai trò quan trọng, quyết định hiệu quả thuật toán được dùng. Luận án đề xuất các quy tắc cập nhật mùi mới: SMMAS, MLAS và 3-LAS. Các thuật toán này bất biến đối với phép biến đổi đơn điệu hàm mục tiêu, thực nghiệm trên các bài toán cơ bản như TSP, UBQP, lập lịch sản xuất với dữ liệu chuẩn cho thấy các thuật toán đề xuất có hiệu quả và dễ sử dụng hơn so với các thuật toán thông dụng nhất hiện nay như ACS và MMAS.

Trong các thuật toán này, *SMMAS đơn giản, dễ sử dụng hơn nên có thể dùng rộng rãi*. Thuật toán MLAS cho phép điều tiết linh hoạt khả năng khám phá và tăng cường của thuật toán theo từng thời điểm. Tuy thực nghiệm trên bài toán TSP cho kết quả hứa hẹn nhưng khó áp dụng hơn. Thuật toán 3-LAS thích hợp với các bài toán có thông tin heuristic tốt, khi sử dụng chúng ảnh hưởng nhiều tới chất lượng của kết quả tìm kiếm, chẳng hạn như bài toán TSP.

Bên cạnh phát triển thuật toán mới, luận án cũng đề xuất các giải pháp cho ba bài toán quan trọng trong sinh học phân tử: suy diễn haplotype, tìm tập hạt giống tối ưu và dự báo hoạt động điều tiết gen.

Đối với bài toán suy diễn haplotype, luận án đề xuất thuật toán ACOHAP. Kết quả thực nghiệm cho thấy ACOHAP cho kết quả tối ưu như RPoly (phương pháp chính xác tốt nhất hiện nay) trong nhiều trường hợp, hơn nữa, ACOHAP hiệu quả nổi trội hơn hẳn CollHap (phương pháp xấp xỉ tốt nhất hiện nay).

Đối với bài toán tìm tập hạt giống tối ưu, luận án đề xuất thuật toán AcoSeeD. Kết quả thực nghiệm cho thấy AcoSeeD cho kết quả tốt hơn hai phương pháp tốt nhất hiện nay là SpEED và SpEEDfast.

Đối với bài toán dự báo hoạt động điều tiết gen, dựa trên phương pháp đề xuất của Zinzen và các cộng sự, luận án đề xuất hai thuật toán metaheuristic: GASVM và ACOSVM. Các thuật toán này tương ứng sử dụng phương pháp GA hoặc ACO để tìm tham số tốt nhất cho bộ học SVM. Thực nghiệm cho thấy hiệu quả hơn cách tiếp cận áp dụng phương pháp tìm kiếm trên lưới của Zinzen.

Hiện tại hệ ACOHAP, AcoSeeD, GASVM và ACOSVM sẽ có ích cho các nhà nghiên cứu sinh học và những người quan tâm.

Trong tương lai, chúng tôi sẽ cùng với nhóm nghiên cứu Tin-Sinh của Đại học Công nghệ ứng dụng các đề xuất mới này cho các bài toán khác.

DANH MỤC CÁC CÔNG TRÌNH KHOA HỌC CỦA TÁC GIẢ LIÊN QUAN ĐẾN LUẬN ÁN

[1] Huy Q. Dinh, Dong Do Duc, and Huan X. Hoang (2006), "Multi-Level Ant System - A new approach through the new pheromone update for Ant Colony Optimization", *Proc. of the 4th IEEE International Conference in Computer Sciences, Research, Innovation, and Vision for Future*, pp. 55-58.

[2] D. Do Duc, Huy.Q. Dinh, and H. Hoang Xuan (2008), "On the pheromone update rules of ant colony optimization approaches for the job shop scheduling problem," *Proc. of the Pacific Rim Int. Workshop on Multi-Agents*, 2008, pp. 153-160.

[3] Hoàng Xuân Huấn và Đỗ Đức Đông (2010), "Về vết mùi trong các thuật toán ACO và khung cảnh mới", *Kỷ yếu hội thảo quốc gia các vấn đề chọn lọc của CNTT lần thứ XII*, tr. 534-547.

[4] Dong Do Duc, Huan Hoang Xuan (2010), "Smoothed and Three-Level Ant Systems: Novel ACO Algorithms for the Traveling Salesman Problem", *Ad. Cont. to the IEEE RIFV2010*, pp. 37-39.

[5] Đỗ Đức Đông và Hoàng Xuân Huấn (2011), "Về biến thiên của vết mùi trong phương pháp ACO và các thuật toán mới", *Tạp chí Tin học và điều khiển học*, Tập 27, tr. 263-275.

[6] Dong Do Duc and Hoang Xuan Huan (2011), "A Fast and Efficient Ant Colony Optimization for Haplotype Inference by Pure Parsimony", *Proc. of the Third International Conference on Knowledge and Systems Engineering*, pp. 128-134.

[7] Dong Do Duc, Tri-Thanh Le, Trung Nghia Vu, Huy Q. Dinh, Hoang Xuan Huan (2012), "GA_SVM: A genetic algorithm for improving gene regulatory activity prediction", *Proc. of the 9th IEEE-RIVF International Conference on Computing and Communication Technologies*, pp. 234-237.

[8] Dong Do Duc, Huan Hoang Xuan, and Huy Q. Dinh (2012), "META-REG: A computational meteheuristic method to improve the regulatory activity prediction", *Proc. of the 4th International Conference on the Development of Biomedical Engineering*, pp. 450-453.

[9] Dong Do Duc, H. Q. Dinh, T.H. Dang, K. Laukens, and H. Hoang Xuan (2012), "AcoSeeD: an Ant Colony Optimization for finding optimal spaced seeds in biological sequence search", *Proc. Of the ANTS2012: Eighth Int. Conf. on swarm intelligence*, pp. 204-211.

TÀI LIỆU THAM KHẢO

Tiếng Việt

[1] **Đỗ Đức Đông** và Hoàng Xuân Huấn (2011), "Về biến thiên của vết mùi trong phương pháp ACO và các thuật toán mới", *Tạp chí Tin học và điều khiển học*, Tập 27, tr. 263-275.

[2] Hoàng Xuân Huấn và **Đỗ Đức Đông** (2010), "Về vết mùi trong các thuật toán ACO và khung cảnh mới", *Kỷ yếu hội thảo quốc gia các vấn đề chọn lọc của CNTT lần thứ XII*, tr. 534-547.

Tiếng Anh

[3] E. Alpaydın (2010), *Introduction to Machine Learning*, Massachusetts Institute of Technology, Second Edition.

[4] S. F. Altschul and W. Gish, and W. Miller, and E. W. Myers, and D. J. Lipman, (1990), "Basic local alignment search tool", *J. Mol. Biol.*, Vol 215 (3), pp. 403-410.

[5] J. E. Beasley (1990), "OR-Library: distributing test problems by electronic mail," *J. Oper. Res. Soc.*, Vol 41(11), pp. 1069-1072.

[6] A. Ben-Hur, C. S. Ong, S. Sonnenburg, B. Scholkopf, and G. Ratsch (2008), "Support vector machines and kernels for computational biology" PLoS Comput. Biol., Vol 4 (10), e1000173.

[7] S. Benedettini, A. Roli, and L. Gaspero (2008), "Two-level ACO for haplotype inference under pure parsimony", *Proc. of the 6th international conference on Ant Colony Optimization and Swarm Intelligence*, pp. 179-190.

[8] M. Birattari, P. Pellegrini, and M. Dorigo (2007), "On the invariance of ant colony optimization", *IEEE Transactions on Evolutionary Computation*, Vol 11 (6), pp. 732-742.

[9] C. Blum (2002), "ACO applied to group shop scheduling: A case study on intensification and diversification", *Proc. of ANTS 2002, Third International Workshop on ant algorithms*, Vol 2463, pp. 14-27.

[10] C. Blum and M. Dorigo (2004), "The Hyper-Cube Framework for Ant Colony Optimization", *IEEE Transactions on Systems, Man, and Cybernetics – Part B*, Vol 34 (2), pp. 1161-1172.

[11] D. G. Brown and I. M. Harrower (2006), "Integer programming approaches to haplotype inference by pure parsimony", *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, Vol 3 (2), pp. 141-154.

[12] B. Bullnheimer, R.F. Hartl, C. Strauss (1999), "A new rank based version of the ant system - a computational study", *Central European J. Oper. Res. Econom*, pp. 25-38.

[13] D. G. Brown (2007), "A survey of seeding for sequence alignments", *Bioinformatics Algorithms: Techniques and Applications*, pp. 117-142.

[14] P. Collas (2010), "The Current State of Chromatin Immunoprecipitation", *Molecular Biotechnology*, Vol 45 (1), pp. 87-100.

[15] C. Cortes and V. Vapnik (1995), "Support-vector networks", *Machine Learning*, Vol 20, pp. 273-297.

[16] M. J. Daly, J. D. Rioux, S. F. Schaffner, T. J. Hudson, and E.S. Lander (2001), "High-Resolution Haplotype Structure in the Human Genome" *Nature Genetics*, Vol 29, pp. 229-232.

[17] M. David, M. Dzamba, D. Lister, L. Ilie, and M. Brudno (2011), "SHRiMP2: sensitive yet practical SHort Read Mapping", *Bioinformatics*, Vol 27 (7), pp. 1011-1012.

[18] E. H. Davidson (2006), "The Regulatory Genome: Gene Regulatory Networks in Development and Evolution", *Elsevier*, pp. 1–86.

[19] L. Di Gaspero and A. Roli (2008), "Stochastic local search for large-scale instances of the haplotype inference problem by pure parsimony", *Journal of Algorithms*, Vol 63 (3), pp. 55-69.

[20] Dinh Quang Huy, Do Duc Dong, Hoang Xuan Huan (2006), "Multi-level Ant System : A New Approach Through the New Pheromone Update for Ant ColonyOptimization", *Proc. of The IEEE RIFV06*, pp. 55-58.

[21] Do Duc Dong, Dinh Quang Huy, Hoang Xuan Huan (2008), "On the Pheromone Update Rules of Ant Colony Optimization Approaches for the Job Shop Scheduling Problem", *Proc. Of PRIMA 2008,* pp. 153-160.

[22] Dong Do Duc, H. Q. Dinh, T.H. Dang, K. Laukens, and H. Hoang Xuan (2012), "AcoSeeD: an Ant Colony Optimization for finding optimal spaced seeds in biological sequence search", *Proc. Of the ANT 2012: Eighth Int. Conf. on swarm intelligence*, pp. 204-211.

[23] Dong Do Duc**,** Huan Hoang Xuan, and Huy Q. Dinh (2012), "META-REG: A computational meteheuristic method to improve the regulatory activity prediction*", Proc. of the 4th International Conference on the Development of Biomedical Engineering*, pp. 450-453.

[24] Dong Do Duc and Hoang Xuan Huan (2011), "A Fast and Efficient Ant Colony Optimization for Haplotype Inference by Pure Parsimony*", Proc. of the Third International Conference on Knowledge and Systems Engineering*, pp. 128-134.

[25] D. D. Do and H. Hoang Xuan (2010), "Smooth and Three-levels Ant Systems: Novel ACO Algorithms for Solving Traveling Salesman Problem", *Ad. Cont. to the International Conference: IEEE-RIVF 2010*, pp. 33-37.

[26] Dong Do Duc, Tri-Thanh Le, Trung Nghia Vu, Huy Q. Dinh, Hoang Xuan Huan (2012), "GA_SVM: A genetic algorithm for improving gene regulatory activity prediction", *Proc. of the 9th IEEE-RIVF International Conference on Computing and Communication Technologies*, pp. 234-237.

[27] B. Doerr, F. Neumann, D. Sudholdt, and C. Witt (2007), *On the influence of pheromone updates in ACO algorithms*, Technical Report CI-223/07, University of Dortmund, SFB 531.

[28] M. Dorigo, V. Maniezzo and A. Colorni (1991), *The Ant System: An autocatalytic optimizing process*, Technical Report 91-016 Revised, Dipartimento di Elettronica, Politecnico di Milano, Milano, Italy.

[29] M. Dorigo (1992), *Optimization, learning and natural algorithms*, PhD. dissertation, Milan Polytechnique, Italy.

[30] M. Dorigo and L.M. Gambardella (1997), "Ant colony system: A cooperative learning approach to the traveling salesman problem", *IEEE Trans. on evolutionary computation*, Vol 1 (1), pp. 53-66.

[31] M. Dorigo, and T. Stützle (2004), *Ant Colony Optimization*, The MIT Press, Cambridge, Masachusetts.

[32] A. S. Graca, J. Marques-silva, I. Lynce, and A. L. Oliveira (2007), "Efficient haplotype inference with pseudo-boolean optimization", *Algebraic Biology*, Vol 4545, pp. 125-139.

[33] S. Graca, I. Lynce, J. Marques and A. L. Oliveira (2010), "Haplotye inference by pure parsimony: a survey", *Journal of computational biology*, Vol 17 (8), pp. 969-992.

[34] D. Gusfield (2001), "Inference of haplotypes from samples of diploid populations: complexity and algorithms", *Comput Biology*, Vol 8 (3), pp. 305-523.

[35] D. Gusfield (2003), "Haplotype Inference by Pure Parsimony", *Proc. 14th Ann. Symp. Combinatorial Pattern Matching*, pp. 144-155.

[36] W. J. Gutjahr (2000), "An Ant based System and its convergence", *future generation Comput. Systems*, Vol 16, pp. 873-888.

[37] W. J. Gutjahr (2002), "ACO algorithms with guaranteed convergence to the optimal solution", *Info. Proc. Lett.*, Vol 83 (3), pp. 145-153.

[38] W. J. Gutjahr (2007), "Mathematical runtime analysis of ACO algorithms: survey on an emerging issue", *Swarm Intelligence*, Vol 1 (1), pp. 59-79.

[39] M. S. Halfon, S. M. Gallo, and C. M. Bergman (2008), "REDfly 2.0: an integrated database of cis-regulatory modules and transcription factor binding sites in Drosophila", *Nucleic Acids Res.*, Vol 36, pp. 594-598.

[40] Hoang Xuan Huan and Dinh Trung Hoang (2002), "On the ant colony system for postman problem", *Journal of science, Vietnam National University, Hanoi*, Vol XVIII (1), pp. 29-36.

[41] N. Homer, B. Merriman, and S. F. Nelson (2009), "BFAST: an alignment tool for large scale genome resequencing", *PLoS ONE*, Vol 4 (11), e7767.

[42] C. L. Huang and C. J. Wang (2006), "A GA-based feature selection and parameters optimization for support vector machines," *Expert Systems with Applications*, Vol 31 (2), pp. 231-240.

[43] Ji-Hong Zhang , Ling-Yun Wu , Jian Chen , Xiang-Sun Zhang (2008), "A fast haplotype inference method for large population genotype data", *Computational Statistics & Data Analysis*, Vol 52 (11), pp. 4891-4902.

[44] H. Ji and W. H. Wong (2005), "TileMap: create chromosomal map of tiling array hybridizations", *Bioinformatics*, Vol 21, pp. 3629-3636.

[45] G. Kucherov, L. Noe and M Roytberg (2006), "A unifying framework for seed sensitivity and its application to subset seeds", *Bioinform Comput Biol*, Vol 4 (2), pp. 553-569.

[46] G. Lancia, M. C. Pinotti, and R. Rizzi (2004). "Haplotyping populations by pure parsimony: Complexity of exact and approximation algorithms", *Journal on Computing*, Vol 16 (4), pp. 348-359.

[47] M. Li, B. Ma, D. Kisman, and J.Tromp (2004), "PatternHunter II: highly sensitive and fast homology search", *Bioinformatics and Computational Biology*, Vol 2 (3), pp. 417-440.

[48] Z. Li, W. Zhou, X. S. Zhang, and L. A.Chen (2005), "Parsimonious tree-grow method for haplotype inference", *Bioinformatics*, Vol 21 (17), pp. 3475-3481.

[49] L. Ilie, and S. Ilie (2007), "Multiple spaced seeds for homology search", *Bioinformatics*, Vol 23 (22), pp. 2969-2977.

[50] L. Ilie, S. Ilie, and A. M. Bigvand (2011), "SpEED: fast computation of sensitive spaced seeds", *Bioinformatics*, Vol 27 (17), pp. 2433-2434.

[51] S. Ilie (2012), "Efficient Computation of Spaced Seeds", *BMC Research Notes*, Vol 5 (123).

[52] J. Marchini, D. Cutler, N. Patterson, M. Stephens, E. Eskin, E. Halperin, S. Lin, Z.S. Qin, H. M. Munro, G. R. Abecasis, P. Donnelly, and International HapMap Consortium (2006), "A Comparison of Phasing Algorithms for Trios and Unrelated Individuals", *Amercan Journal of Human Genetics*, Vol 78, pp. 437-450.

[53] F. Neumann, D. Sudholt and CarstenWitt (2008), "Rigorous Analyses for the Combination of Ant Colony Optimization and Local Search", *Proceedings of the Sixth International Conference on Ant Colony Optimization and Swarm Intelligence*, pp. 132-143.

[54] P. J. Park (2009), "ChIP-seq: advantages and challenges of a maturing technology" *Nat. Rev. Genet*., Vol 10, pp. 669–680.

[55] P. Pellegrini and A. Ellero (2008), "The Small World of Pheromone Trails", *Proc. of the 6th international conference on Ant Colony Optimization and Swarm Intelligence*, pp. 387-394.

[56] M. Randall (2006), "Search Space Reduction as a Tool for Achieving Intensification and Diversification in Ant Colony Optimisation", *Proc. of the 19th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, pp. 254-261.

[57] C. Reeves (1995), *Genetic Algorithms and Combinatorial Optimisation: Applications of Modern Heuristic Techniques*, In V.J. Rayward-Smith, Alfred Waller Ltd, Henley-on-Thames, UK.

[58] R. S. Rosa and K. S. Guimarães (2010), "Insights on Haplotype Inference on Large Genotype Datasets*", Advances in Bioinformatics and Computational Biology*, Vol 6268, pp. 47-58.

[59] M. Sampels, J. Knowles and K. Socha (2002), "A MAX-MIN Ant System for the University Timetabling Problem", *Proc. of the 3rd International Workshop on Ant Algorithms*, pp. 1-13.

[60] A. Schrijver (2006), *A Course in Combinatorial Optimization*, Department of Mat., University of Amsterdam.

[61] T. Sing, O. Sander, N. Beerenwinkel, and T. Lengauer (2005), "ROCR: visualizing classifier performance in R", *Bioinformatics*, Vol 21, pp. 3940-3941.

[62] T. F. Smith and M. S Waterman (1981), *"Identification of common molecular subsequences", J. Mol. Biol.*, Vol 147 (1), pp. 195-197.

[63] K. Socha, M. Sampels and M. Manfrin (2003). "Ant Algorithms for the Univerrsity Course Timetabling Problem with Regard to the State-of-the-Art",

Applications of Evolutionary Computing, Proceedings of the EvoWorkshops 2003, pp. 334-345.

[64] A. Stark (2009), "*Learning the transcriptional regulatory code*" Mol. Syst. Biol., Vol 5, pp. 329.

[65] T. Stützle and M. Dorigo (2002), "A short convergence proof for a class of ACO algorithms", *IEEE-EC*, Vol 6 (4), pp. 358-365.

[66] T. Stützle and H. H. Hoos (2000), "Max-Min ant system", *Future Gene. Comput*. Syst., Vol 26 (8), pp. 889-914.

[67] Y. Sun, and J. Buhler (2005), "Designing multiple simultaneous seeds for DNA similarity search", *J. Comput. Biol.*, Vol 12 (6), pp. 847-861.

[68] L. Tininini, P. Bertolazzi,A.Godi and G.Lancia (2010), "Collhaps: a heuristic approach to haplotype inference by parsimony", *IEEE/ACM Trans Comput Biol Bioinformatic*, Vol 7 (3), pp. 511-523.

[69] R. S. Wang, X. S. Zhang, and L. Sheng (2005), "Haplotype inference by pure parsimony via genetic algorithm", *In Operations Research and Its Applications, the Fifth International Symposium (ISORA'05)*, Vol 5, pp. 308-318.

[70] Z. Zang and Z. Feng (2012), "Two-stage updating pheromone forinvariant ant colony optimization algorithm", *Expert System with applications*, Vol 39 (1), pp. 706-712.

[71] R. P. Zinzen, C. Girardot, J. Gagneur, M. Braun, and E. E. Furlong (2009), "Combinatorial binding predicts spatio-temporal cis-regulatory activity", *Nature*, Vol 462, pp. 65–70.

[72] S. Zwaan, C. Marques (1998), "Ant colony Optimisation for Job shop Scheduling", ISR – Instituto de Sistemas e Robótica Instituto Superior Técnico (IST).

[73] http://cran.r-project.org/web/packages/e1071/index.html

[74] http://cran.r-project.org/web/packages/genalg/index.html

[75] http://en.wikipedia.org/wiki/Memetic_algorithm

[76] http://iridia.ulb.ac.be/~mdorigo/ACO/aco-code/public-software.htm

[77] http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/

[78] http://www.nature.com/nature/journal/v462/n7269/extref/nature08531-s10.zip

[79] http://www.stats.ox.ac.uk/~marchini/phaseoff.html