Tối ưu hóa (4)

TS. Đỗ Đức Đông
dongdoduc@gmail.com

Các cách tiếp cận

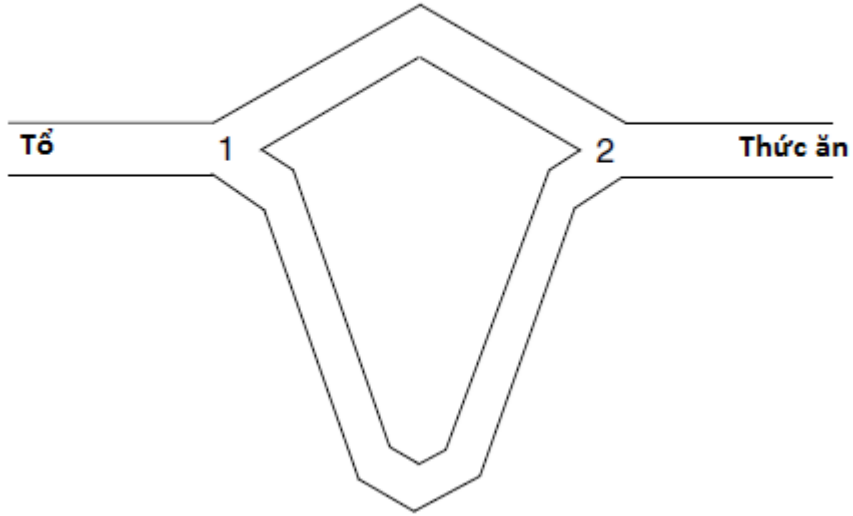
Tối ưu tổ hợp	Tối ưu liên tục
• Các phương pháp truyền thống Chứng minh hội tụ hoặc tỷ lệ tối ưu • Các phương pháp dựa trên thực nghiệm + Heuristic kiến trúc (constructive heuristic) + Tìm kiếm địa phương (local search) + Metaheurisics: - GA (Genetic Algorithms) - ACO (Ant Colony Optimization) - Memetic algorithm	• Quy hoạch tuyến tính • Quy hoạch phi tuyến + Tìm kiếm địa phương - Phương pháp gradient …. + Tối ưu toàn cục - Quy hoạch lồi, hiệu lồi - Tìm kiếm ngẫu nhiên (Monter-Carlo) - GA (Genetic Algorithms) …

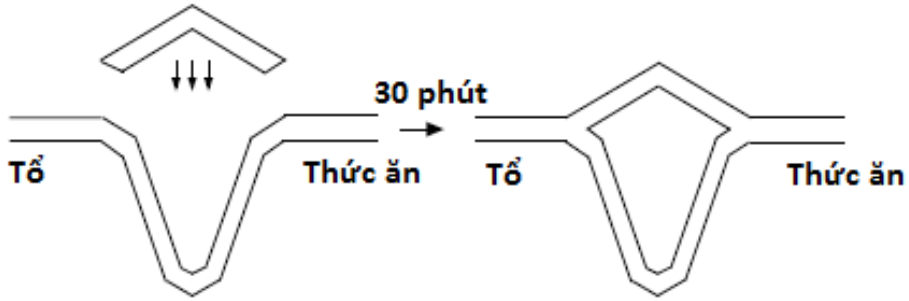
Tối ưu đàn kiến (Ant Colony Optimization - ACO)

- Được đề xuất bởi Dorigo năm 1991 trong luận án tiến sĩ
- 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
- Phương pháp được cải tiến đa dạng và có nhiều ứng dụng
- Hiệu quả hơn so với thuật toán di truyền
- Từ năm 1998, cứ hai năm một lần có một hội nghị quốc tế về chủ đề này với các kết quả phong phú.

Kiến tự nhiên

- Ban đầu đàn kiến lựa chọn hai nhánh đi như nhau. 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 → kinh nghiệm
- Không phải tất cả các kiến đều đi theo nhánh ngắn hơn, điều này minh chứng bầy kiến đã sử dụng phương thức thăm dò, tìm đường mới → khám phá
- 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.





Kiến nhân tạo

- Vết mùi của đàn kiến liên tưởng tới cách học tăng cường (reinforcement learning) trong bài toán chọn tác động tối ưu → mô hình mô phỏng cho bài toán tìm đường đi ngắn nhất giữa hai nút (tổ và nguồn thức ăn) trên đồ thị, trong đó các tác tử (agent) là đàn kiến nhân tạo.
- 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
- Các bài toán ứng dụng các đồ thị thường phức tạp hơn, 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 (có bộ nhớ riêng, có khả năng ghi nhớ các đỉnh đã thăm, tính được độ dài đường đi, trao đổi thông tin với nhau, cập nhật mùi…)
- Sử dụng mô hình kiến nhân tạo, Dorigo (1991) đã xây dựng thuật toán *hệ kiến* (AS) giải bài toán TSP. Hiệu quả của thuật toán so với các phương pháp 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.

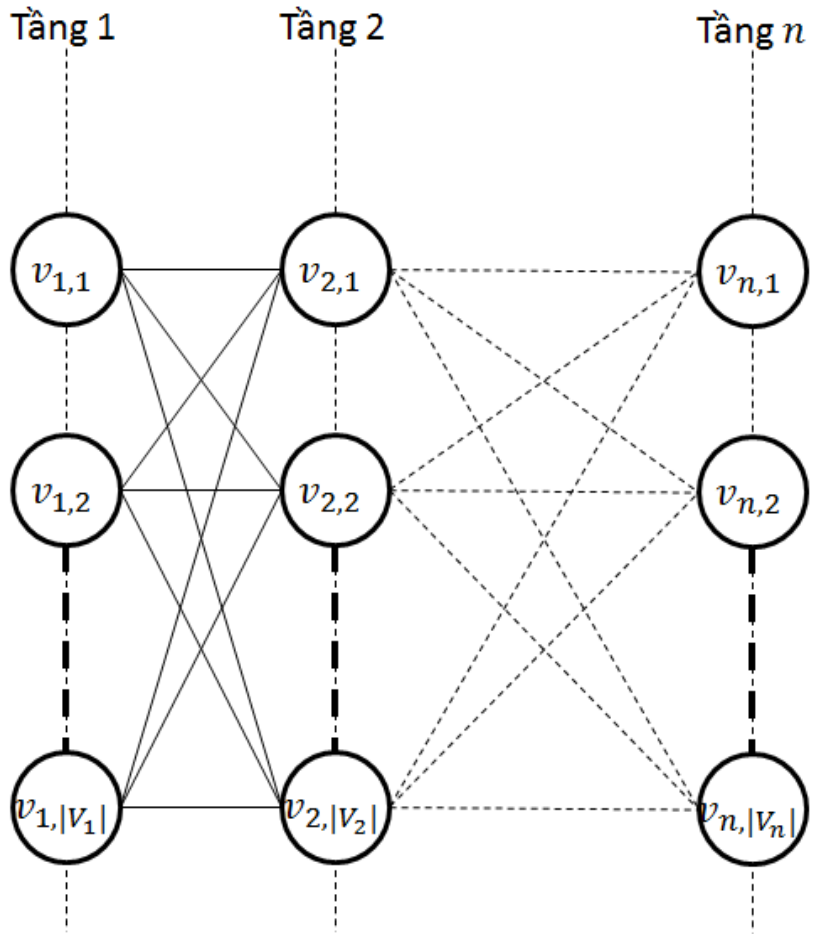
Phương pháp ACO cho bài toán TƯTH tổng quát

- Tối ưu hàm $f(x_1, x_2, .., x_n)$, với x_i thuộc tập giá trị hữu hạn V_i và thỏa mãn tập rang buộc Ω.
- Thủ tục bước ngẫu nhiên

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

- Cập nhật mùi

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



Lược đồ thuật toán

Procedure Thuật toán ACO;

Begin

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

repeat

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

Kiến k xây dựng lời giải;

end-for

Cập nhật mùi;

Cập nhật lời giải tốt nhất;

until (Điều kiện kết thúc);

Đưa ra lời giải tốt nhất;

End;

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:

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.

Các công việc chính khi áp dụng ACO giải bài toán

1) *Xây dựng đồ thị cấu trúc thích hợp*. Việc xây dựng đồ thị cấu trúc để tìm được lời giải cho bài toán theo thủ tục 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.

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.

Phương pháp ACO giải bài toán TSP (1)

- Bài toán người chào hàng (*Traveling Salesman Problem - TSP*) có nhiều ứng dụng trong thực tế.
- Thuộc loại NP-khó và được xem là bài toán chuẩn để đánh giá hiệu quả của các thuật toán giải các bài toán TƯTH mới.
- Thuật toán ACO đầu tiên được gọi là 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.

Phương pháp ACO giải bài toán TSP (2)

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.

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)

Thuật toán AS giải bài toán TSP

Lược đồ thuật toán

Procedure ASTSP;
Dữ liệu vào: $G = (V, E)$;
Kết quả ra: Một chu trình và tổng độ dài của chu trình;
Begin
Khởi tạo tham số, ma trận mùi, khởi tạo m con kiến;
repeat
    Cả k kiến xây dựng lời giải;
    Cập nhật mùi;
    Cập nhật lời giải tốt nhất;
until (Điều kiện kết thúc);
Đưa ra lời giải tốt nhất;
End;

Thuật toán AS giải bài toán TSP

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.
- 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.

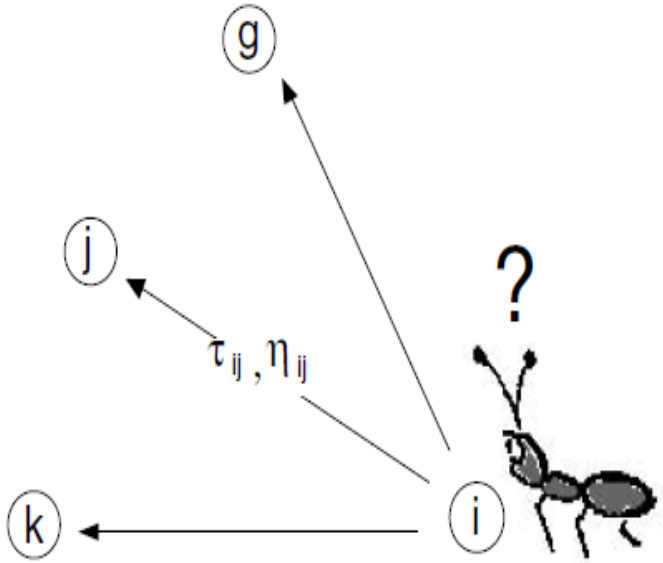
Thuật toán AS giải bài toán TSP

Thủ tục xây dựng lời giải

- m kiến đồng thời xây dựng lời giải.
- 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 để chọn đỉnh tiếp theo. Cụ thể, khi kiến k đang ở đỉnh i sẽ lựa chọn đỉnh j với xác suất:

$$p_{ij}^k = \begin{cases} \dfrac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in N_i^k} [\tau_{il}]^\alpha [\eta_{il}]^\beta}, \text{nếu } j \in N_i^k \\ 0, \qquad\qquad \text{nếu } j \notin N_i^k \end{cases}$$

trong đó $\eta_{ij} = \dfrac{1}{d_{ij}}$ 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



Thuật toán AS giải bài toán TSP

Cập nhật mùi

Sau khi tất cả các kiến xây dựng xong hành trình, vết mùi sẽ được cập nhật:

1) Tất cả các cạnh sẽ bị bay hơi theo một tỉ lệ không đổi

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij},\ 0 < \rho \leq 1 \text{ là hệ số bay hơi,}$$

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.

2) Tất cả kiến sẽ để lại vết mùi trên cạnh đi qua

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

$\Delta\tau_{ij}^{k}$ là lượng mùi do kiến k cập nhật trên cạnh kiến k đi qua.

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

Hệ kiến ACS (1)

Thuật toán ACS (Ant Colony System) do nhóm của Dorigo đề xuất năm 1997, 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).

Hệ kiến ACS (2)

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:

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

trong đó q là một biến ngẫu nhiên, phân bố đều trong [0,1], q_0 $(0 \leq$

Hệ kiến ACS (3)

Cập nhật mùi

Cập nhật mùi toàn cục: Chỉ có duy nhất một kiến tìm được lời giải G-best được phép để lại mùi sau mỗi lần lặp. Với các cạnh(i, j) thuộc lời giải G-best được cập nhật

$\tau_{ij} \leftarrow (1-\rho)\tau_{ij} + \rho\Delta\tau_{ij}^{best}$, trong đó $\Delta\tau_{ij}^{best} = \frac{1}{C^{G-best}}$, C^{G-best} là độ dài lời giải G-best (cũng có thể sử dụng chọn lời giải tốt nhất trong bước lặp hiện tại i-best để cập nhật mùi)

Cập nhật mùi cục bộ: Thực hiện ngay khi cạnh (i, j) có kiến đi qua

$\tau_{ij} \leftarrow (1-\xi)\tau_{ij} + \xi\tau_0$, trong đó $\xi (0 < \xi < 1)$ và τ_0 là hai tham số, τ_0 chính là giá trị khởi tạo mùi cho tất cả các cạnh.

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, tăng khả năng khám phá các cạnh chưa được sử dụng, không bị tắc nghẽn (nghĩa là, các kiến không bị dồn vào một đường đi nào) như trong AS.

Hệ kiến MMAS (1)

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

1) 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.

2) 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 $[\tau_{min}, \tau_{max}]$.

3) 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.

4) 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.

Hệ kiến MMAS (2)

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 và được thêm một lượng mùi cho tất cả các cạnh thuộc lời giải tốt:

$\tau_{ij} \leftarrow \tau_{ij} + \Delta\tau_{ij}^{best}$, 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 $[\tau_{min}, \tau_{max}]$, cụ thể $\tau_{i,j} = \begin{cases} \tau_{max} & nếu\ \tau_{i,j} > \tau_{max} \\ \tau_{i,j} & nếu\ \tau_{i,j} \in [\tau_{min}, \tau_{max}] \\ \tau_{min} & nếu\ \tau_{i,j} < \tau_{min} \end{cases}$

Hệ kiến MMAS (3)

Khởi trị và khởi tạo lại 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.

ACO algorithm	α	β	ρ	m	τ_0
AS	1	2 to 5	0.5	n	m/C^{nn}
EAS	1	2 to 5	0.5	n	$(e+m)/\rho C^{nn}$
AS_{rank}	1	2 to 5	0.1	n	$0.5r(r-1)/\rho C^{nn}$
$\mathcal{MM}$AS	1	2 to 5	0.02	n	$1/\rho C^{nn}$
ACS	—	2 to 5	0.1	10	$1/nC^{nn}$

$\mathcal{MM}AS$: The pheromone trail limits are $\tau_{max} = 1/\rho C^{bs}$ and $\tau_{min} = \tau_{max}(1 - \sqrt[n]{0.05})/((avg - 1) \cdot \sqrt[n]{0.05})$, *where avg* is the average number of different choices available to an ant at each step while constructing a solution (for a justification of these values see Stützle & Hoos (2000). When applied to small TSP instances with up to 200 cities, good results are obtained by using always the iteration-best pheromone update rule, while on larger instances it becomes increasingly important to alternate between the iteration-best and the best-so-far pheromone update rules.

ACS: In the local pheromone trail update rule: $\xi = 0.1$. In the pseudorandom proportional action choice rule: $q_0 = 0.9$.

Hệ kiến SMMAS (1)

Thuật toán SMMAS (Smoothed Max-Min Ant System) do Do đề xuất năm 2008 giống với MMAS chỉ với cách cập nhật mùi khác.

$$\Delta\tau_{i,j} = \begin{cases} \rho\tau_{min} \; n\acute{e}u \; (i,j) \notin w(t) \\ \rho\tau_{max} \; n\acute{e}u \; (i,j) \in w(t) \end{cases}$$

trong đó $w(t)$ là G-best hoặc i-best G-best

Hệ kiến SMMAS (2)

Thuật toán SMMAS đơn giản nhưng tận dụng được các ưu điểm của MMAS và khắc phục các nhược 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 tham số tốt, trong SMMAS thiết đặt τ_{max}=1.0 và chỉ cần xác định tỉ lệ giữa τ_{min} và 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 không cần phải xây dựng hàm này, vết mùi được tiến trơn mịn về hai cận.

3) Quy tắc cập nhật mùi SMMAS không ảnh hưởng khi hàm mục tiêu được biến đổi tuyến tính đơn điệu

Một số chú ý khi áp dụng ACO

- **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.
- **Kết hợp với tìm kiếm cục bộ:** Chỉnh lại những bước đi sai lầm của kiến.
- **Thông tin heuristic:** Khi đó vết mùi không thể giúp kiến tìm đường đi dẫn tới các lời giải tốt (chưa khác nhau nhiều), giúp kiến có thể xây dựng được các hành trình tốt ngay trong giai đoạn đầu.
- **Số lượng kiến:** 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 → nên dùng nhiều kiến. Khi có sử dụng tìm kiếm cục bộ hoặc thông tin heuristic mạnh, sử dụng nhiều kiến là lãng phí.
- **Tham số bay hơi:** Khi sử dụng tìm kiếm cục bộ hoặc thông tin heuristic mạnh, tham số bay hơi nên được xác lập có giá trị lớn, điều này giúp kiến quên đi những lời giải đã xây dựng, tập trung tìm kiếm xung quanh lời giải tốt mới được xây dựng. Trong trường hợp ngược lại, tham số bay hơi nên được thiết lập với giá trị nhỏ.

Thuật toán memetic sử dụng GA hoặc ACO

Begin

Khởi tạo quần thể P_0;// nếu là GA

t:=0;

while chưa gặp điều kiện dừng) **do**

t:=t+1;

Xây dựng quần thể P(t); //với GA, t=0 thì dùng luôn P_0

Đánh giá P(t) và chọn tập Ω(t);

Áp dụng các thuật toán tìm kiếm trong tập A(t) cho Ω(t);

Cập nhật mùi (ACO)/ đánh giá;

End ;

End;

Bài toán suy diễn haplotype

- **Biểu diễn haplotype:** bằng một xâu độ dài m chỉ gồm 2 loại ký tự 0, 1
- **Biểu diễn genotype:** bằng một xâu độ dài m chỉ gồm 3 loại ký tự 0, 1, 2
- **Giải thích genotype**

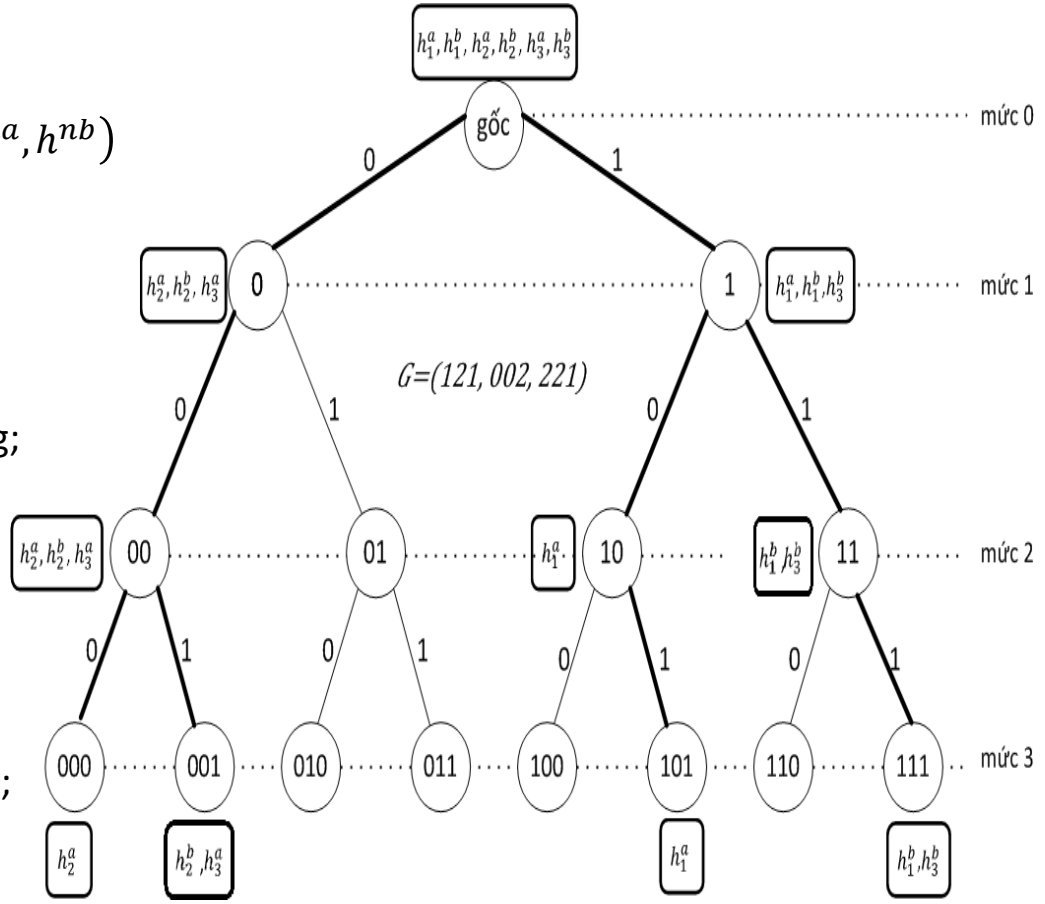
  Cặp haplotype không thứ tự $< h^a, h^b >$ giải thích genotype g nếu:
  - $g_i = 0$ thì $h_i^a = h_i^b = 0$,
  - $g_i = 1$ thì $h_i^a = h_i^b = 1$,
  - $g_i = 2$ thì $(h_i^a = 0 \wedge h_i^b = 1)$ hoặc $(h_i^a = 1 \wedge h_i^b = 0)$
- **Bài toán suy diễn haplotype:** n genotype, $G = (g^1, \dots, g^n)$ có độ dài m đã cho bài toán HI là tìm danh sách $2n$ haplotype $H = (h^{1a}, h^{1b}, h^{2a}, h^{2b}, .., h^{na}, h^{nb})$ giải thích n genotype.
- **Tiêu chuẩn pure parsimony:** lực lượng tập haplotype nhỏ nhất. Bài toán thuộc lớp NP-khó, không gian tìm kiếm lớn.
- Ví dụ: G=(121, 002, 221) H_1=(**101, 111, 000, 001, 011**, 101); H_2=(**101, 111, 000, 001,** 001, 111)

ACO giải bài toán suy diễn haplotype

Thủ tục xây dựng lời giải

procedure Xây dựng lời giải;
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 ở mức i tương ứng;

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



ACO giải bài toán suy diễn haplotype

Cập nhật mùi

- $\tau_{i,v}^{s}$ *:* vết mùi thể hiện sự ưu tiên lựa chọn của haplotype thứ s ở vị trí thứ i nhận giá trị v
- Cập nhật mùi theo SMMAS

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

$$\text{với } \Delta\tau_{i,v}^{s} = \begin{cases} \rho\tau_{min} \text{ nếu } h_i^{sa} \neq v \\ \rho\tau_{max} \text{ nếu } h_i^{sa} = v \end{cases}$$

Kết quả thực nghiệm so sánh với CollHap

Trên bộ dữ liệu chuẩn

Bảng tóm tắt thông tin về các bộ dữ liệu chuẩn

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

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

Bộ dữ liệu	Số lượng test ACOHAP<CollHap	Số lượng test ACOHAP=CollHap	Số lượng test ACOHAP>CollHap
SU-100kb	7%	86%	7%
SU1	76%	24%	0%
SU2	40%	55%	5%
SU3	66%	34%	0%

CollHap (Tininini 2010)

Kết quả thực nghiệm

Trên bộ dữ liệu thực

real data	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**

RPoly(Graca 2008)
PTG (Li 2005)

Đọc và trình bày

- http://www.aco-metaheuristic.org/aco-code/
- https://academic.oup.com/bioinformatics
- https://bmcbioinformatics.biomedcentral.com/
- http://www.springer.com/mathematics/journal/10732