

Giải thuật đàn kiến tự thích ứng cho bài toán điều hướng thu thập

Lê Thế Việt - 20520093, Huỳnh Hoàng Vũ - 20520864

Trường Đại học Công nghệ Thông tin, Đại học Quốc gia Thành phố Hồ Chí Minh



Giảng viên hướng dẫn: TS. Lương Ngọc Hoàng

Tháng 1, 2024

Vu Hoang Huynh, The Viet Le and Ngoc Hoang Luong. “Self-Adaptive Ant System with Hierarchical Clustering for the Thief Orienteering Problem”. In: Proceedings of the 12th International Symposium on Information and Communication Technology. SOICT 2023. ACM, Dec. 2023.

- 1 Bài toán Điều Hướng Thu Thập (ThOP)
- 2 Các công trình liên quan
- 3 Thuật toán tân tiến trước đây (ACO++)
- 4 Giải thuật đàn kiến tự thích ứng (SAAS)
- 5 Thực nghiệm
- 6 Kết luận

1 Bài toán Điều Hướng Thu Thập (ThOP)

- Định nghĩa
- Ví dụ
- ThOP benchmark

2 Các công trình liên quan

3 Thuật toán tân tiến trước đây (ACO++)

4 Giải thuật đàn kiến tự thích ứng (SAAS)

5 Thực nghiệm

6 Kết luận

1 Bài toán Điều Hướng Thu Thập (ThOP)

- Định nghĩa
 - Ví dụ
 - ThOP benchmark

2 Các công trình liên quan

3 Thuật toán tân tiến trước đây (ACO++)

4 Giải thuật đàn kiến tự thích ứng (SAAS)

5 Thực nghiệm

6 Kết luận

Bài toán Điều Hướng Thu Thập (ThOP)

Bài toán Điều Hướng Thu Thập (ThOP)

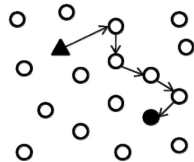
Bài toán Điều Hướng Thu Thập (**T**hief **O**rienteeing **P**roblem, ThOP) ¹là bài toán tối ưu hóa **đa thành phần** bao gồm 2 thành phần, **bài toán Điều Hướng** (**O**rienteeing **P**roblem, OP) và **bài toán Ba Lô** (**K**napsack **P**roblem, KP).

¹André G. Santos et al. “The Thief Orienteering Problem: Formulation and Heuristic Approaches”. In: *2018 IEEE Congress on Evolutionary Computation (CEC)*. 2018, pp. 1–9

Bài toán Điều Hướng Thu Thập (ThOP)

Bài toán Điều Hướng (OP)

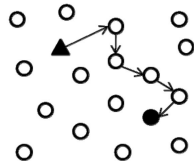
OP là một bài toán định tuyến với mục tiêu **xác định một con đường** trong tập thành phố cho trước, nhằm **tối đa hóa tổng điểm đạt được** từ các thành phố đi qua, trong khi vẫn **đảm bảo giới hạn thời gian**.



Bài toán Điều Hướng Thu Thập (ThOP)

Bài toán Điều Hướng (OP)

OP là một bài toán định tuyến với mục tiêu **xác định một con đường** trong tập thành phố cho trước, nhằm **tối đa hóa tổng điểm đạt được** từ các thành phố đi qua, trong khi vẫn **đảm bảo giới hạn thời gian**.



Bài toán Ba Lô (KP)

KP là một bài toán tối ưu hóa tổ hợp với mục tiêu **lựa chọn nhất các vật phẩm** trong tập vật phẩm cho trước, nhằm **tối đa hóa lợi nhuận thu thập được** từ các vật phẩm, trong khi vẫn **đảm bảo giới hạn sức chứa**.



7

2

1

9



5

4

7

2

A

B

C

D



Max Weight: 15kg

1 Bài toán Điều Hướng Thu Thập (ThOP)

- Định nghĩa
- Ví dụ
- ThOP benchmark

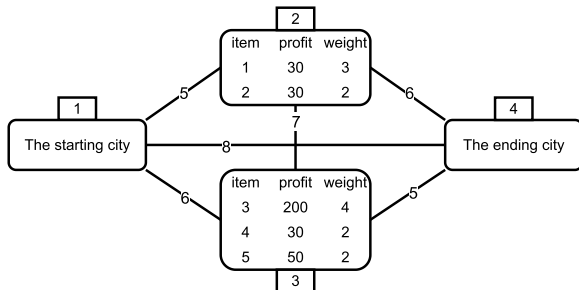
2 Các công trình liên quan

3 Thuật toán tân tiến trước đây (ACO++)

4 Giải thuật đàn kiến tự thích ứng (SAAS)

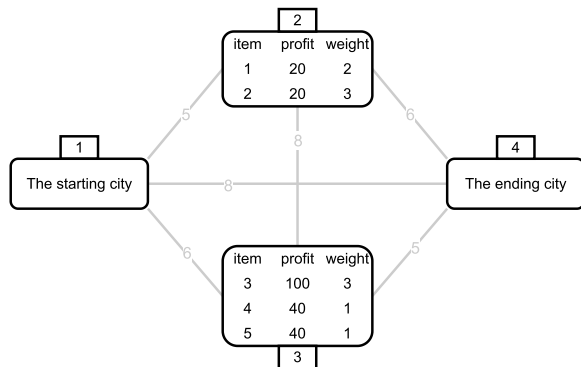
5 Thực nghiệm

6 Kết luận



Ràng buộc

- $n = 4, m = 5$
- $v_{min} = 0.1, v_{max} = 1.0, W = 3, T = 75$



Ràng buộc

- $n = 4, m = 5$
- $v_{min} = 0.1, v_{max} = 1.0, W = 3, T = 75$

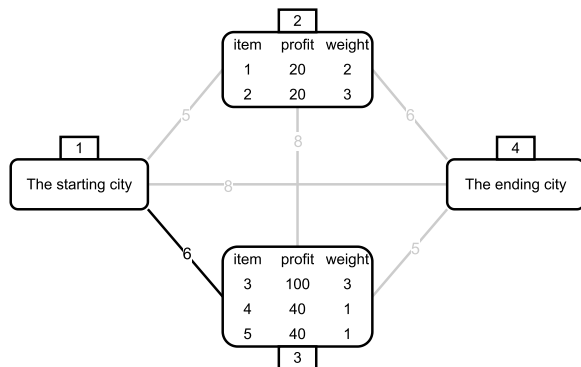
Lời giải

- $\pi = \langle 1 \rangle$
- $p = \langle 0, 0, 0, 0, 0 \rangle$

Thuộc tính

- $p = 0$
- $w = 0$
- $v = v_{max} = 1.0$
- $t = 0$

ThOP - Ví dụ



Ràng buộc

- $n = 4, m = 5$
- $v_{min} = 0.1, v_{max} = 1.0, W = 3, T = 75$

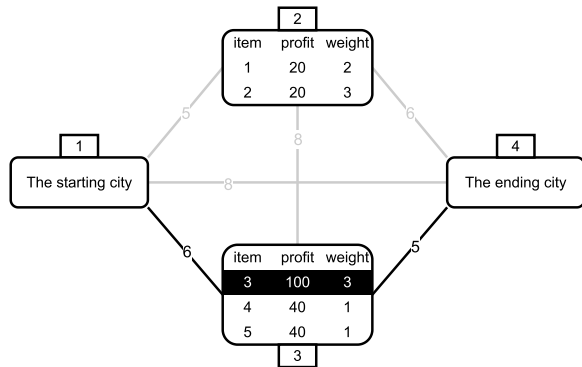
Lời giải

- $\pi = \langle 1, 3 \rangle$
- $p = \langle 0, 0, 0, 0, 0 \rangle$

Thuộc tính

- $p = 0$
- $w = 0$
- $v = v_{max} = 1.0$
- $t = d_{1,3}/v = 6/1.0 = 6$

ThOP - Ví dụ



Ràng buộc

- $n = 4, m = 5$
- $v_{min} = 0.1, v_{max} = 1.0, W = 3, T = 75$

Lời giải

- $\pi = \langle 1, 3, 4 \rangle$
- $p = \langle 0, 0, 1, 0, 0 \rangle$

Thuộc tính

- $p = 100$
- $w = 0 + w_3 = 3$
- $v = v_{max} - w(v_{max} - v_{min})/W = 0.1$
- $t = t + d_{3,4}/v = 6 + 5/0.1 = 56$

1 Bài toán Điều Hướng Thu Thập (ThOP)

- Định nghĩa
- Ví dụ
- ThOP benchmark

2 Các công trình liên quan

3 Thuật toán tân tiến trước đây (ACO++)

4 Giải thuật đàn kiến tự thích ứng (SAAS)

5 Thực nghiệm

6 Kết luận

ThOP Benchmark

ThOP benchmark là tập hợp gồm 432 trường hợp của bài toán. Các trường hợp mang tính chất khác nhau về:

- Số lượng thành phố: 51, 107, 280, hoặc 1000.
- Số lượng vật phẩm tại mỗi thành phố: 01, 03, 05, hoặc 10.
- Mỗi quan hệ giữa cân nặng và lợi nhuận: uncorrelated (unc), uncorrelated with similar weights (usw), hoặc bounded and strongly correlated (bsc).
- Kích thước balo: 01, 05, hoặc 10 lần kích thước nhỏ nhất.
- Thời gian di chuyển tối đa: 50%, 75%, hoặc 100%.

- 1 Bài toán Điều Hướng Thu Thập (ThOP)
- 2 Các công trình liên quan
- 3 Thuật toán tân tiến trước đây (ACO++)
- 4 Giải thuật đàn kiến tự thích ứng (SAAS)
- 5 Thực nghiệm
- 6 Kết luận

Các thuật toán trước đây cho ThOP

- Tìm Kiếm Cục Bộ Tuần Tự (**I**terated **L**ocal **S**earch, ILS) ¹.
- Thuật Giải Di Truyền Điểm Thiên Lệch Ngẫu Nhiên (**B**iased **R**andom-**K**ey **G**enetic **A**lgorithm, BRKGA) ¹.
- Thuật Giải Di Truyền (**G**enetic **A**lgorithm, GA) ².
- Giải Thuật Tối Ưu Hóa Đàn Kiến (**A**nt **C**olony **O**ptimization, ACO) ³.
- ACO++ ⁴.

²Leonardo M. Faêda et al. "A Genetic Algorithm for the Thief Orienteering Problem". In: *2020 IEEE Congress on Evolutionary Computation (CEC)*. 2020, pp. 1–8

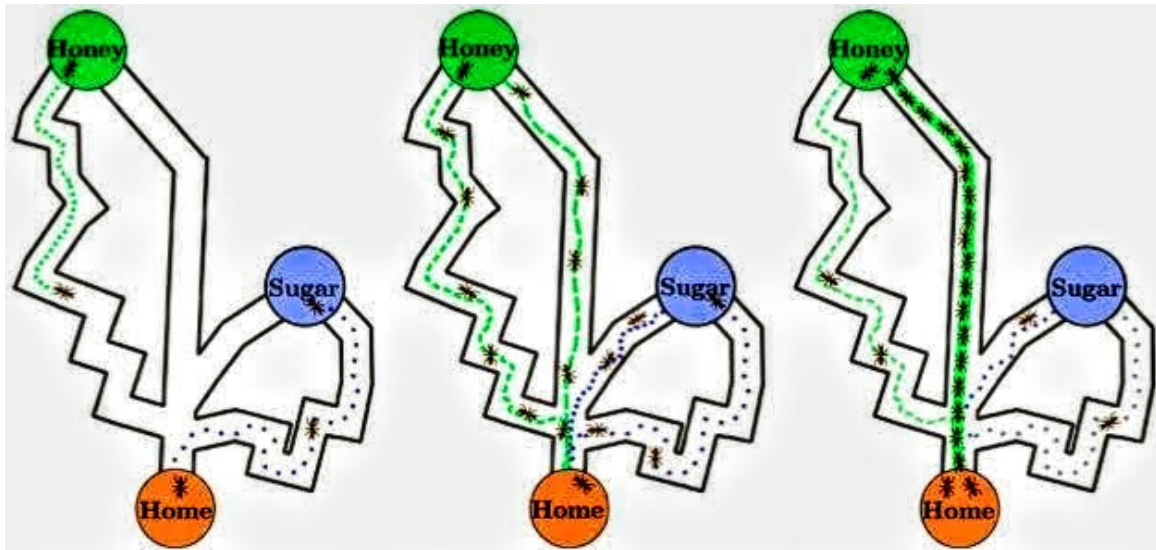
³Jonatas B.C. Chagas et al. "Ants can orienteer a thief in their robbery". In: *Operations Research Letters* 48.6 (2020), pp. 708–714. ISSN: 0167-6377

⁴Jonatas B. C. Chagas et al. "Efficiently solving the thief orienteering problem with a max–min ant colony optimization approach". In: *Optimization Letters* 16.8 (Nov. 2021), pp. 2313–2331

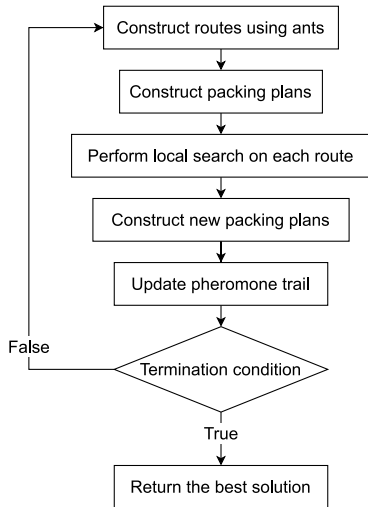
- 1 Bài toán Điều Hướng Thu Thập (ThOP)
- 2 Các công trình liên quan
- 3 Thuật toán tân tiến trước đây (ACO++)
 - Tổng quan
 - Nhược điểm
- 4 Giải thuật đàn kiến tự thích ứng (SAAS)
- 5 Thực nghiệm
- 6 Kết luận

- 1 Bài toán Điều Hướng Thu Thập (ThOP)
- 2 Các công trình liên quan
- 3 Thuật toán tân tiến trước đây (ACO++)
 - Tổng quan
 - Nhược điểm
- 4 Giải thuật đàn kiến tự thích ứng (SAAS)
- 5 Thực nghiệm
- 6 Kết luận

Ý tưởng của ACO



Tổng quan ACO++



ACO++

- ACO++ được Chagas và Wagner đề xuất và trở thành thuật toán tân tiến cho ThOP.
- ACO++ là sự kết hợp của Hệ thống Kiến Max-Min Heuristic nhất ngẫu nhiên và Tìm kiếm cục bộ.
- ACO++ vượt trội hơn các thuật toán trước đây trên 95% benchmark.

Heuristic nhặt ngẫu nhiên của ACO++

Điểm vật phẩm s_i

$$s_i = \frac{p_i^\theta}{w_i^\delta * d_i^\gamma}$$

Heuristic nhặt ngẫu nhiên (Randomized Packing Heuristic)

- 1 Tính điểm cho các vật phẩm.
- 2 Xem xét vật phẩm có điểm cao nhất tiếp theo.
- 3 Nếu nhặt thêm vật phẩm không làm vi phạm ràng buộc, thêm nó vào balo.
- 4 Nếu chưa xem xét qua tất cả vật phẩm, quay lại Bước 2.

- 1 Bài toán Điều Hướng Thu Thập (ThOP)
- 2 Các công trình liên quan
- 3 Thuật toán tân tiến trước đây (ACO++)
 - Tổng quan
 - Nhược điểm
- 4 Giải thuật đàn kiến tự thích ứng (SAAS)
- 5 Thực nghiệm
- 6 Kết luận

Nhược điểm của ACO++

Nhược điểm của ACO++

- 1 Heuristic nhất vật phẩm lệ thuộc nhiều vào yếu tố ngẫu nhiên.

- 2

Heuristic nhất vật phẩm lệ thuộc nhiều vào ngẫu nhiên

Điểm vật phẩm s_i

$$s_i = \frac{p_i^\theta}{w_i^\delta * d_i^\gamma}$$

Trong đó

- p_i : Lợi nhuận của vật phẩm i .
- w_i : Trọng lượng của vật phẩm i .
- d_i : Độ dài đường đi từ thành phố chứa vật phẩm i đến thành phố cuối cùng theo thứ tự đường đi đang xét.
- θ, δ, γ : **Giá trị được tạo ngẫu nhiên** trong khoảng $[0,1]$.

Nhược điểm của ACO++

Nhược điểm của ACO++

- ① Heuristic nhất vật phẩm lệ thuộc nhiều vào yếu tố ngẫu nhiên.
- ② Hiệu suất nhạy với giá trị siêu tham số. Các siêu tham số cần được điều chỉnh riêng cho mỗi 9 trường hợp.

Hiệu suất phụ thuộc nhiều vào siêu tham số

Quá trình điều chỉnh siêu tham số tốn kém

- ACO++ đòi hỏi một quá trình điều chỉnh tốn kém để đạt được kết quả vượt trội.
- 240.000 thí nghiệm đã được tiến hành để điều chỉnh 48 bộ siêu tham số cho 48 nhóm trường hợp.
- Mỗi nhóm bao gồm 9 trường hợp.

Hiệu suất phụ thuộc nhiều vào siêu tham số

Thí nghiệm 1

Sử dụng bộ siêu tham số được điều chỉnh riêng cho nhóm trường hợp.

Thí nghiệm 2

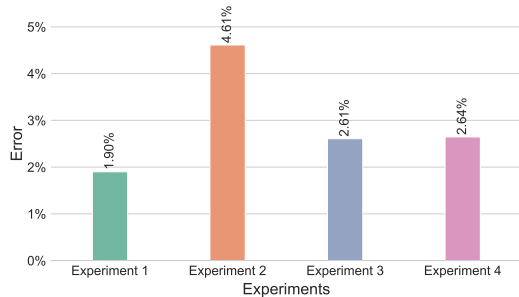
Hoán đổi bộ siêu tham số của hai nhóm trường hợp.

Thí nghiệm 3

Tăng 3% cho α , β , ρ và tăng 1 cho số lần thử nhất vật phẩm.

Thí nghiệm 4

Lấy trung bình của 48 bộ siêu tham số.



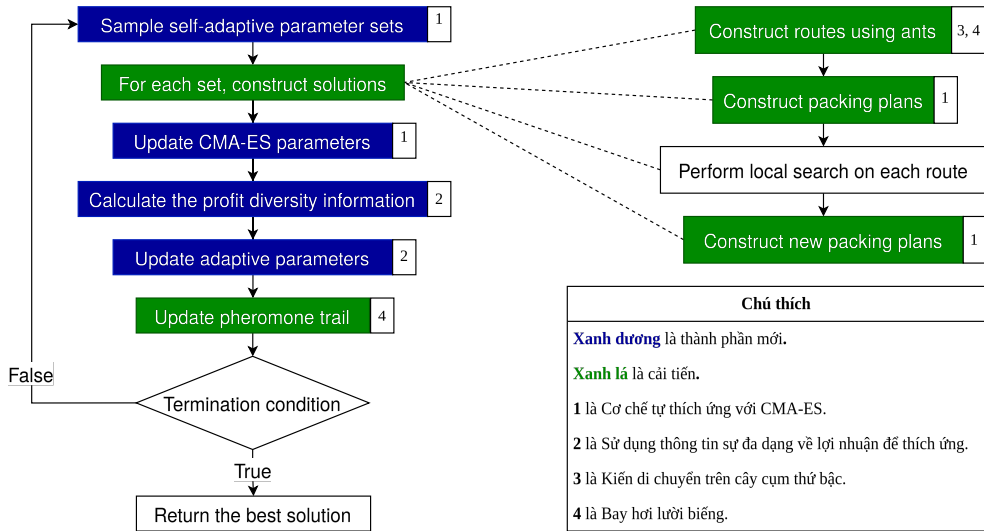
Hình: Sai số trung bình của kết quả từ 4 thí nghiệm ACO++ với các bộ tham số khác nhau trên 18 trường hợp thuộc 2 nhóm trường hợp ThOP. Nhỏ hơn là tốt hơn.

- 1 Bài toán Điều Hướng Thu Thập (ThOP)
- 2 Các công trình liên quan
- 3 Thuật toán tân tiến trước đây (ACO++)
- 4 **Giải thuật đàn kiến tự thích ứng (SAAS)**
 - Cơ chế tự thích ứng với CMA-ES
 - Sử dụng thông tin sự đa dạng về lợi nhuận để thích ứng
 - Kiến di chuyển trên cây cụm thứ bậc
 - Bay hơi lười biếng
- 5 Thực nghiệm
- 6 Kết luận

Đóng góp của đề tài

- Chúng tôi đề xuất Giải Thuật Đàn Kiến Tự Thích Ứng (**Self-Adaptive Ant System**, SAAS), là phiên bản mở rộng của ACO++.
- SAAS có thể điều chỉnh các tham số của mình dựa trên đặc điểm riêng biệt của trường hợp bài toán và quá trình tìm kiếm.
- Đồng thời, nó có độ phức tạp thời gian thấp hơn ACO++ ở giai đoạn kiến chọn thành phố và giai đoạn bay hơi pheromone.

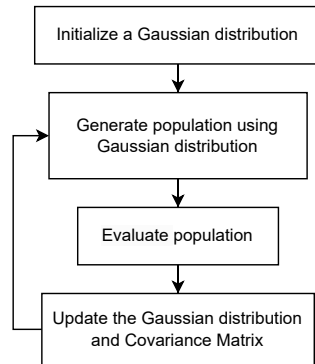
Tổng quan thuật toán SAAS



- 1 Bài toán Điều Hướng Thu Thập (ThOP)
- 2 Các công trình liên quan
- 3 Thuật toán tân tiến trước đây (ACO++)
- 4 Giải thuật đàn kiến tự thích ứng (SAAS)
 - Cơ chế tự thích ứng với CMA-ES
 - Sử dụng thông tin sự đa dạng về lợi nhuận để thích ứng
 - Kiến di chuyển trên cây cụm thứ bậc
 - Bay hơi lười biếng
- 5 Thực nghiệm
- 6 Kết luận

Giới thiệu

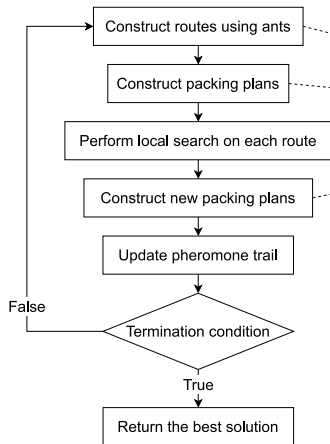
CMA-ES⁵(**C**ovariance **M**atrix **A**daptation **E**volution **S**trategy, Chiến lược Tiến hóa Thích ứng Ma trận Hiệp Phương Sai) là một phương pháp tối ưu hóa số, ngẫu nhiên, **không sử dụng đạo hàm** cho các bài toán tối ưu hóa **phi tuyến tính** hoặc **không lồi** trên không gian **liên tục**.



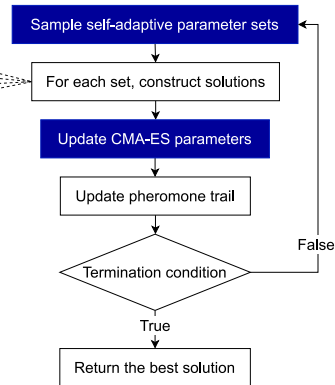
Hình: Sơ đồ đơn giản hóa CMA-ES.

⁵Nikolaus Hansen. "The CMA Evolution Strategy: A Comparing Review". In: *Towards a New Evolutionary Computation: Advances in the Estimation of Distribution Algorithms*. Ed. by Jose A. Lozano et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 75–102. ISBN: 978-3-540-32494-2

Cơ chế thích ứng với CMA-ES



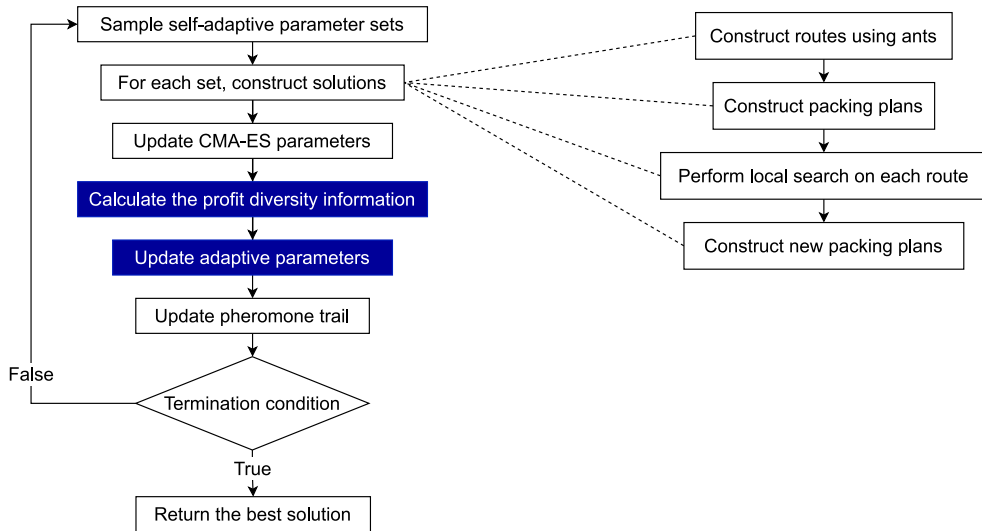
ACO++



CMA-ES + ACO++

- 1 Bài toán Điều Hướng Thu Thập (ThOP)
- 2 Các công trình liên quan
- 3 Thuật toán tân tiến trước đây (ACO++)
- 4 **Giải thuật đàn kiến tự thích ứng (SAAS)**
 - Cơ chế tự thích ứng với CMA-ES
 - Sử dụng thông tin sự đa dạng về lợi nhuận để thích ứng
 - Kiến di chuyển trên cây cụm thứ bậc
 - Bay hơi lười biếng
- 5 Thực nghiệm
- 6 Kết luận

Sử dụng thông tin sự đa dạng về lợi nhuận để thích ứng



Sử dụng thông tin sự đa dạng về lợi nhuận để thích ứng

Ý tưởng chính

Lấy cảm hứng từ thuật toán AACO-NC⁶, chúng tôi sử dụng thông tin sự đa dạng về lợi nhuận để thay đổi linh hoạt cả tỷ lệ bay hơi pheromone và số lượng kiến cho mỗi cá thể của CMA-ES.

Thông tin sự đa dạng về lợi nhuận

$$p_i = \frac{\text{\#số lần xuất hiện}(P_i)}{n_{\text{ants}}} \quad (1)$$

$S = \{P \mid P \text{ là giá trị độc nhất được tìm bởi đàn kiến hiện tại}\},$

$$H = - \sum_{i=1}^{|S|} p_i \cdot \log_2 p_i \quad (2)$$

⁶Petr Stodola et al. “Adaptive Ant Colony Optimization with node clustering applied to the Travelling Salesman Problem”. In: *Swarm and Evolutionary Computation* 70 (2022), p. 101056. ISSN: 2210-6502

Sử dụng thông tin sự đa dạng về lợi nhuận để thích ứng

Thích ứng tỷ lệ bay hơi pheromone

- Tỷ lệ bay hơi **tăng khi** lợi nhuận **đa dạng cao** và **giảm khi** lợi nhuận **đa dạng thấp**.

$$\rho = \rho_{\min} + (\rho_{\max} - \rho_{\min}) \cdot \frac{H - H_{\min}}{H_{\max} - H_{\min}}. \quad (3)$$

Thích ứng số lượng kiến cho mỗi cá thể CMA-ES

- Trái ngược với tỷ lệ bay hơi, giá trị của n_{indv} **tăng khi** lợi nhuận **đa dạng thấp** để khuyến khích khám phá và **giảm khi** lợi nhuận **đa dạng cao** để tập trung khai thác.

$$n_{\text{indv}} = n_{\text{indv_max}} - (n_{\text{indv_max}} - n_{\text{indv_min}}) \cdot \frac{H - H_{\min}}{H_{\max} - H_{\min}}. \quad (4)$$

Các tham số được điều khiển

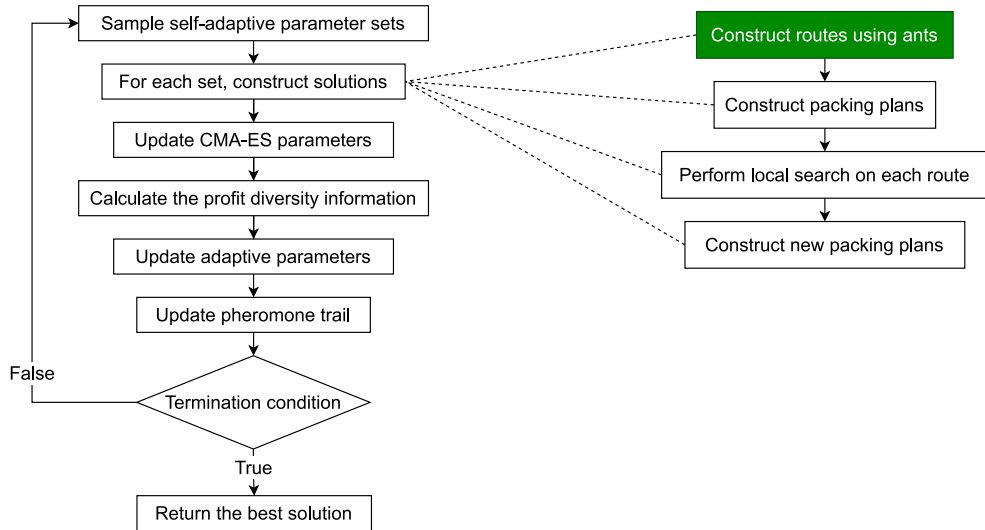
Bảng: Danh sách các tham số được điều khiển bởi cơ chế điều khiển tham số.

Tham số	Cơ chế điều khiển tham số	Khoảng giá trị
α	Tự thích ứng	$[0, 1]$
β	Tự thích ứng	$[0, 1]$
ρ_{\min}, ρ_{\max}	Tự thích ứng	$[0, 1]$
θ	Tự thích ứng	$[0, 1]$
δ	Tự thích ứng	$[0, 1]$
γ	Tự thích ứng	$[0, 1]$
n_{indv}	Thích ứng	$[n_{\text{indv_max}}, n_{\text{indv_min}}]$
ρ	Thích ứng	$[\rho_{\min}, \rho_{\max}]$

- Ba số mũ ngẫu nhiên θ, δ, γ trong Heuristic nhát vật phẩm đã được điều chỉnh bởi CMA-ES.

- 1 Bài toán Điều Hướng Thu Thập (ThOP)
- 2 Các công trình liên quan
- 3 Thuật toán tân tiến trước đây (ACO++)
- 4 **Giải thuật đàn kiến tự thích ứng (SAAS)**
 - Cơ chế tự thích ứng với CMA-ES
 - Sử dụng thông tin sự đa dạng về lợi nhuận để thích ứng
 - **Kiến di chuyển trên cây cụm thứ bậc**
 - Bay hơi lười biếng
- 5 Thực nghiệm
- 6 Kết luận

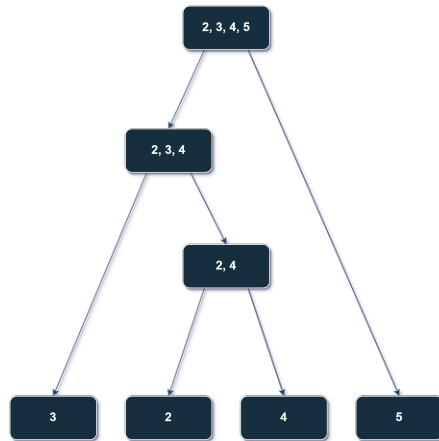
Kiến di chuyển trên cây cụm thứ bậc



Kiến di chuyển trên cây cụm thứ bậc

Ý tưởng chính

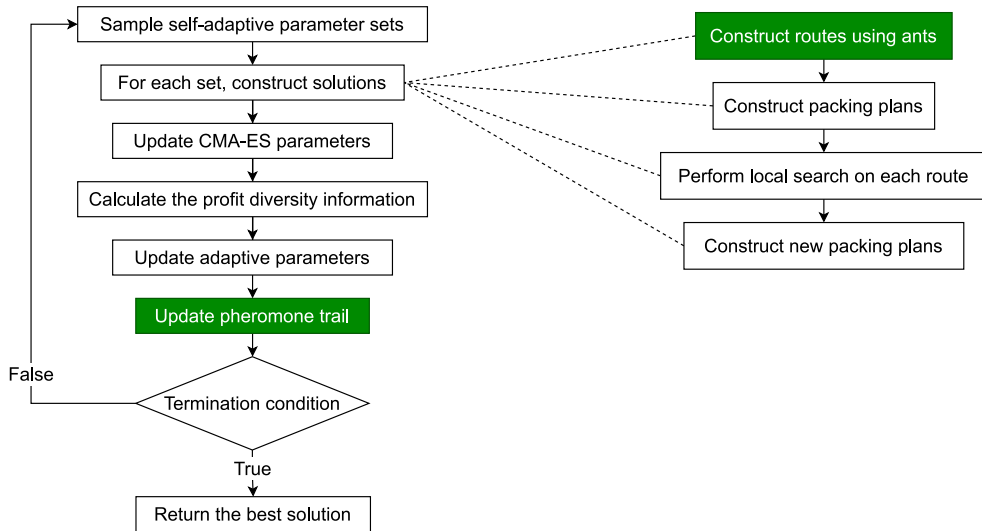
- Chúng tôi sử dụng phân cụm thứ bậc (hierarchical clustering) để xây dựng kiến trúc cây.
- Mỗi thành phố có một cây cụm thứ bậc đại diện cho các cạnh đi đến n thành phố.
- Kiến sẽ duyệt cây từ gốc đến lá thay vì di chuyển trực tiếp từ thành phố đến thành phố.
- Bằng cách này, giai đoạn kiến chọn thành phố tiếp theo sẽ có độ phức tạp $\Theta(\log n)$.



Hình: Ví dụ cây cụm thứ bậc.

- 1 Bài toán Điều Hướng Thu Thập (ThOP)
- 2 Các công trình liên quan
- 3 Thuật toán tân tiến trước đây (ACO++)
- 4 **Giải thuật đàn kiến tự thích ứng (SAAS)**
 - Cơ chế tự thích ứng với CMA-ES
 - Sử dụng thông tin sự đa dạng về lợi nhuận để thích ứng
 - Kiến di chuyển trên cây cụm thứ bậc
 - Bay hơi lười biếng
- 5 Thực nghiệm
- 6 Kết luận

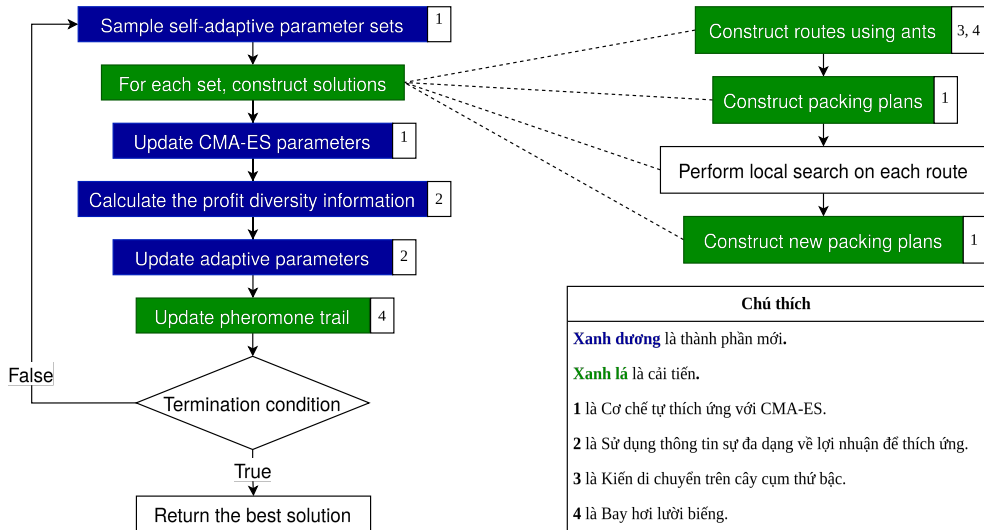
Bay hơi lười biếng



Ý tưởng chính

- Nguyên tắc cốt lõi của bay hơi lười biếng là ghi nhận trạng thái mong muốn và trạng thái lịch sử.
- Trạng thái mong muốn bao gồm số lần bay hơi pheromone mong muốn.
- Mỗi cạnh sẽ có trạng thái lịch sử riêng, bao gồm số lần nồng độ pheromone của cạnh thực sự được bay hơi.
- Bằng cách so sánh trạng thái mong muốn và trạng thái lịch sử, ta có thể xác định nồng độ pheromone với số lần bay hơi mong muốn khi cần thiết.

Tổng quan thuật toán SAAS



- 1 Bài toán Điều Hướng Thu Thập (ThOP)
- 2 Các công trình liên quan
- 3 Thuật toán tân tiến trước đây (ACO++)
- 4 Giải thuật đàn kiến tự thích ứng (SAAS)
- 5 Thực nghiệm**
- 6 Kết luận

Thực nghiệm

Để có sự so sánh công bằng

- Thực nghiệm được tiến hành trên **cùng phần cứng**.
- Các thuật toán có **cùng giới hạn thời gian chạy**.

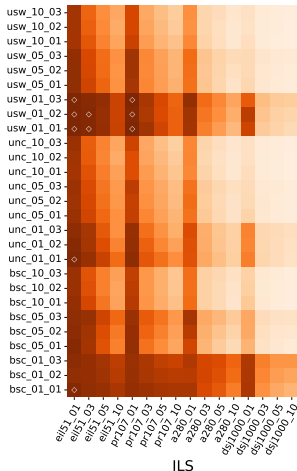
Điều chỉnh siêu tham số

- ILS không có siêu tham số.
- BRKGA, ACO, **ACO++** sử dụng siêu tham số được điều chỉnh trong nghiên cứu của ACO++. Mỗi thuật toán, **240.000 thí nghiệm** đã được tiến hành để điều chỉnh **48 bộ siêu tham số** cho 48 nhóm trường hợp.
- **SAAS** với **45.000 thí nghiệm** để điều chỉnh ra **một bộ siêu tham số** dùng chung cho cả benchmark.

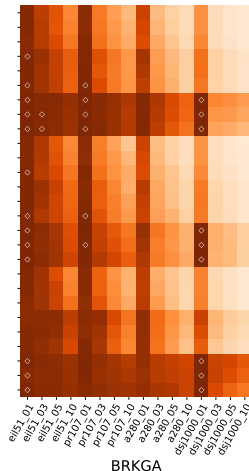
Tỷ lệ xấp xỉ so với lời giải tốt nhất

Chú thích

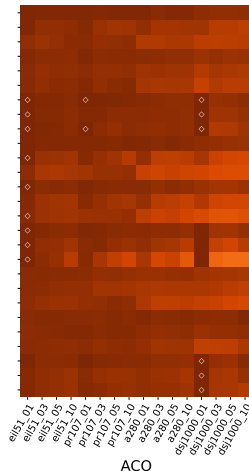
- Mỗi ô là một trường hợp bài toán.
- Màu thể hiện chất lượng của lời giải.
- Màu càng đậm chất lượng càng tốt.



ILS



BRKGA

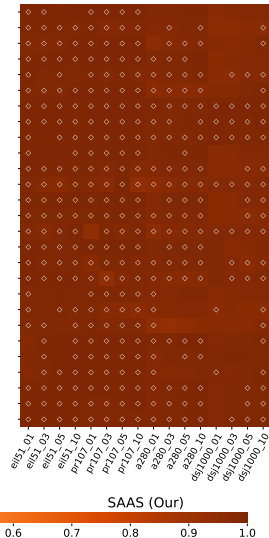


ACO

Tỷ lệ xấp xỉ so với lời giải tốt nhất

Chú thích

- Mỗi ô là một trường hợp bài toán.
- Hình thoi thể hiện tìm được lời giải tốt nhất trong thực nghiệm.
- Càng nhiều hình thoi, thuật toán càng tốt.

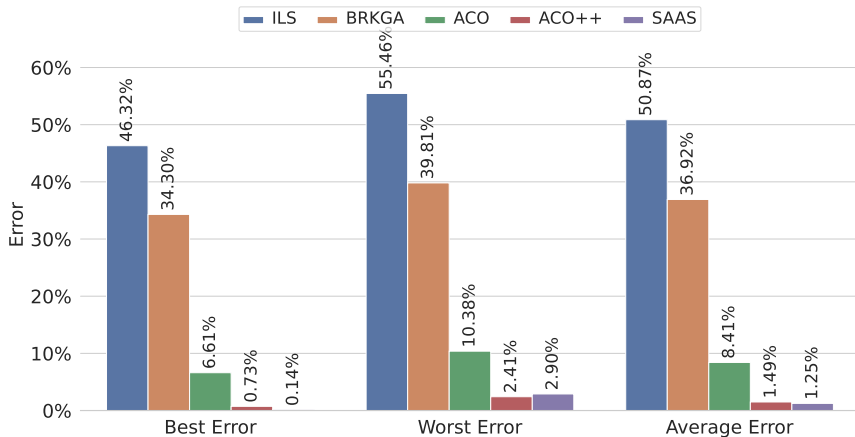


So sánh hiệu suất thuật toán

Bảng: Tỷ lệ các trường hợp mà thuật toán i tìm ra giải pháp có chất lượng tốt hơn hoặc bằng thuật toán j

$i \downarrow \quad j \rightarrow$	ILS	BRKGA	ACO	ACO++	SAAS
ILS	-	2.55%	4.40%	2.55%	2.31%
BRKGA	100.00%	-	16.20%	8.80%	7.18%
ACO	97.22%	87.27%	-	5.79%	4.86%
ACO++	99.54%	95.83%	97.69%	-	41.90%
SAAS (Ours)	99.77%	97.92%	98.61%	78.24%	-

So sánh chất lượng lời giải



Hình: Độ lỗi trung bình qua toàn bộ benchmark. Thấp hơn là tốt hơn.







- 1 Bài toán Điều Hướng Thu Thập (ThOP)
- 2 Các công trình liên quan
- 3 Thuật toán tân tiến trước đây (ACO++)
- 4 Giải thuật đàn kiến tự thích ứng (SAAS)
- 5 Thực nghiệm
- 6 Kết luận

Kết luận

- Hai kỹ thuật kiểm soát tham số được tích hợp để cải thiện khả năng thích nghi và tính linh hoạt.
- Kỹ thuật Bay hơi lười biếng giúp làm giảm độ phức tạp thời gian của giai đoạn bay hơi pheromone.
- Phân cụm thứ bậc (hierarchical clustering) được tận dụng giúp làm giảm chi phí kiến chọn thành phố tiếp theo.

Kết luận

- Hai kỹ thuật kiểm soát tham số được tích hợp để cải thiện khả năng thích nghi và tính linh hoạt.
- Kỹ thuật Bay hơi lười biếng giúp làm giảm độ phức tạp thời gian của giai đoạn bay hơi pheromone.
- Phân cụm thứ bậc (hierarchical clustering) được tận dụng giúp làm giảm chi phí kiến chọn thành phố tiếp theo.
- Giải Thuật Đàn Kiến Tự Thích Ứng (SAAS) hiệu quả hơn ACO++ khi chỉ cần đúng một bộ siêu tham số để chạy toàn bộ benchmark.
- SAAS cho thấy hiệu suất đáng chú ý khi vượt qua tất cả các thuật toán trước đây cho bài toán Điều Hướng Thu Thập (ThOP).

-  Santos, André G. et al. “The Thief Orienteering Problem: Formulation and Heuristic Approaches”. In: *2018 IEEE Congress on Evolutionary Computation (CEC)*. 2018, pp. 1–9.
-  Faêda, Leonardo M. et al. “A Genetic Algorithm for the Thief Orienteering Problem”. In: *2020 IEEE Congress on Evolutionary Computation (CEC)*. 2020, pp. 1–8.
-  Chagas, Jonatas B.C. et al. “Ants can orienteer a thief in their robbery”. In: *Operations Research Letters* 48.6 (2020), pp. 708–714. ISSN: 0167-6377.
-  —. “Efficiently solving the thief orienteering problem with a max–min ant colony optimization approach”. In: *Optimization Letters* 16.8 (Nov. 2021), pp. 2313–2331.
-  Hansen, Nikolaus. “The CMA Evolution Strategy: A Comparing Review”. In: *Towards a New Evolutionary Computation: Advances in the Estimation of Distribution Algorithms*. Ed. by Jose A. Lozano et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 75–102. ISBN: 978-3-540-32494-2.
-  Stodola, Petr et al. “Adaptive Ant Colony Optimization with node clustering applied to the Travelling Salesman Problem”. In: *Swarm and Evolutionary Computation* 70 (2022), p. 101056. ISSN: 2210-6502.