ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH

Giải Thuật Nâng Cao (CO5127)

Bài tập lớn

ỨNG DỤNG GIẢI THUẬT ANT COLONY OPTIMIZATION CHO THỨ TỰ HOÁ CHUỖI (SEQUENCE ORDERING)

GVHD: Lê Hồng Trang
NHÓM: 10
SVTH: Phạm Minh Quang - 2570302
Nguyễn Trung Phong - 2570047
Ngô Nhất Toàn - 2570515
Đoàn Ngọc Thanh Tú – 2470098

Tp. Hồ Chí Minh, Tháng 12/2025

Mục lục

1. Giới thiệu

1.1. Giới thiệu bài toán

Bài toán **Sắp xếp Thứ tự Chuỗi** (Sequential Ordering Problem — SOP) là một trong những bài toán quan trọng trong lĩnh vực **Tối ưu hóa Tổ hợp** (Combinatorial Optimization — CO). Đây là bài toán có độ phức tạp cao và đã thu hút sự quan tâm mạnh mẽ trong cộng đồng nghiên cứu **Khoa học Máy tính** với nhiều ứng dụng thực tế và phương pháp giải quyết khác nhau trong hơn 30 năm qua. Về mặt lý thuyết, SOP là một mô hình tổng quát của nhiều bài toán tổ hợp khác:

- Nếu bỏ ràng buộc thứ tự ưu tiên (precedence constraints), ta thu được Bài toán người du lịch bất đối xứng (ATSP – Asymmetric Traveling Salesman Problem), một bài toán nổi tiếng thuộc lớp NP-hard.
- Nếu chiều dài các cung (arc lengths) là đối xứng, ta thu được Bài toán người du lịch đối xứng (TSP).

Định nghĩa chính thức

Cho một đồ thị có hướng $G = (V, E)$, trong đó:

- V là tập các nút,
- E là tập các cạnh có hướng và có trọng số,
- v_s là nút bắt đầu,
- v_f là nút kết thúc.

Ngoài ra, tồn tại một tập hợp các **ràng buộc ưu tiên** (precedence constraints). Ràng buộc giữa hai nút x và y yêu cầu nút x phải được ghé thăm trước nút y trong chuỗi sắp xếp.

Mục tiêu tối ưu

Mục tiêu của SOP là tìm một **đường đi Hamiltonian** từ v_s đến v_f sao cho:

- đi qua tất cả các nút đúng một lần,
- thỏa mãn toàn bộ ràng buộc ưu tiên,
- tổng chi phí (cost) trên đường đi là nhỏ nhất.

Động lực nghiên cứu

Sự phức tạp do các ràng buộc ưu tiên khiến SOP trở thành mô hình phù hợp cho nhiều ứng dụng công nghiệp và vận hành phức tạp, chẳng hạn như:

- **Lập kế hoạch sản xuất và lập lịch trình:** SOP mô phỏng việc sắp xếp thứ tự thực hiện các công việc hoặc công đoạn trong dây chuyền sản xuất. Chi phí có thể liên quan đến thời gian chuyển đổi, hao hụt vật liệu hoặc chi phí lưu kho. Các ràng buộc ưu tiên biểu diễn sự phụ thuộc công nghệ giữa các công đoạn.
- **Vận tải và logistics:** Trong các bài toán định tuyến phương tiện (Vehicle Routing), SOP giúp xác định chuỗi ghé thăm tối ưu các điểm dừng, nơi ràng buộc ưu tiên có thể liên quan đến việc thu thập/giao hàng theo thứ tự hoặc ràng buộc thời gian.
- **Hệ thống phân tích mạng:** Một số bài toán tối ưu trong hệ thống mạng, như tối ưu truy cập dữ liệu có phụ thuộc, cũng có thể được mô hình hóa dưới dạng SOP.

Do đó, phát triển các thuật toán hiệu quả và mạnh mẽ, có khả năng cân bằng giữa *khai thác* (exploitation) các lời giải tốt hiện tại và *khám phá* (exploration) các vùng mới trong không gian tìm kiếm, là điều thiết yếu để giải quyết SOP ở quy mô lớn.

1.2. Các Phương pháp Giải quyết Phổ biến

Các phương pháp chính được nghiên cứu để giải quyết SOP bao gồm:

Phương pháp Giải Chính xác (Exact Approaches)

Nhiều phương pháp chính xác đã được đề xuất để giải quyết bài toán SOP. Phần lớn nghiên cứu tập trung vào việc tìm kiếm các **cận dưới mạnh** (strong lower bounds).

Một số phương pháp tiêu biểu gồm:

- **Mặt phẳng cắt (Cutting Planes)**.
- **Thuật toán Thư giãn Lagrange kết hợp Cắt (Lagrangian Relax-and-Cut)**.
- **Mô hình Lập trình Toán học với Branch and Bound:** Quá trình phân nhánh được thiết kế để phân rã bài toán tối đa.
- **Phiên bản Uncapacitated m-PDTSP:** Là tổng quát hóa của SOP, đạt kết quả cạnh tranh nhờ thuật toán *Branch and Cut* kết hợp *Generalized Variable Neighborhood Search (GVNS)*.

Một bước tiến đáng kể khác đến từ việc áp dụng **Decision Diagrams**, giúp sinh ra các cận dưới chất lượng cao.

Một thuật toán **Branch and Bound chuyên biệt** được đề xuất, kết hợp:

- **Cận đơn giản và nhanh:** dựa trên prefix, bậc vào/ra và cây khung tối thiểu (MST).
- **Kỹ thuật History Cuts:** lấy cảm hứng từ quy hoạch động của TSP, cho phép loại bỏ các nghiệm trung gian bị chi phối (*dominated partial solutions*).

Mặc dù chỉ sử dụng các cận đơn giản, phương pháp này đạt được kết quả tốt, truyền cảm hứng cho các nghiên cứu sau nhằm phân tích sâu hơn các thành phần của quá trình *branch and bound*.

Các nghiên cứu tiếp theo cải thiện thêm bằng cách tích hợp:

- *Cận gán tùy chỉnh (Custom Assignment Bound)*,
- *Tìm kiếm cục bộ (Local Search)* tại mỗi nút của cây tìm kiếm.

Phương pháp Meta-Heuristic

Nhiều nghiên cứu tập trung vào **phép tìm kiếm cục bộ SOP-3-exchange**, kết hợp với các thuật toán heuristic khác. SOP-3-exchange là phiên bản mở rộng của thuật toán **3-OPT**, có tính đến ràng buộc thứ tự và trọng số cung bất đối xứng. Phép tìm kiếm này được giới thiệu cùng với thuật toán **Tối ưu hóa Đàn Kiến (Ant Colony Optimization — ACO)** và sau đó được áp dụng trong:

- **Thuật toán Bầy hạt (Particle Swarm Optimization)**.
- **Thuật toán Di truyền Lai (Hybrid Genetic Algorithm)** với toán tử lai ghép *Voronoi Quantized Crossover*.
- **Tối ưu hóa Đàn Ong (Bee Colony Optimization)**.
- **Thuật toán Song song Roll-out**.

Trong đó, **HAS-SOP (Hybrid Ant System for SOP)** đạt kết quả rất tốt, nên đã được mở rộng trong nhiều nghiên cứu tiếp theo:

- Cải thiện bằng cấu trúc dữ liệu *"don't push stack"*.
- Tích hợp thêm **mô phỏng tôi luyện (Simulated Annealing)**.
- Cải tiến heuristic **LKH (Lin–Kernighan Heuristic)** và mở rộng để giải các bộ dữ liệu SOP lớn.

Hai phương pháp sau cùng này đạt kết quả tốt nhất trên các bộ dữ liệu lớn trong **SOPLIB**.

1.3. Thuật toán Tối ưu hóa Bầy Kiến (Ant Colony Optimization — ACO)

ACO là một siêu thuật toán được Marco Dorigo đề xuất vào năm 1992 để giải quyết các bài toán CO. Nó mô phỏng hành vi của các tác nhân (kiến) giao tiếp gián tiếp với nhau thông qua các dấu vết pheromone. Cơ chế này được gọi là **stigmergy**.

1.3.1. Cấu trúc Cốt lõi và Cơ chế Pheromone

ACO gồm ba bước chính:

1. Construct Ant Solutions
2. Update Pheromones
3. Daemon Actions

Lựa chọn cạnh và heuristic

Trong quá trình xây dựng lời giải, kiến lựa chọn cạnh tiếp theo dựa trên sự kết hợp của:

$$\tau_{ij} \quad : \text{lượng pheromone trên cạnh } (i,j)$$

$$\eta_{ij} \quad : \text{thông tin heuristic (thường là } \eta_{ij} = 1/d_{ij})$$

Công thức lựa chọn cơ bản (AS).

Xác suất kiến k tại nút i chọn nút tiếp theo j:

$$p_{ij}^k(t) = \frac{[\tau_{ij}(t)]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in N_i^k} [\tau_{il}(t)]^\alpha [\eta_{il}]^\beta} \tag{2.1}$$

Trong đó: α và β điều chỉnh mức ảnh hưởng của pheromone và heuristic. Giá trị β lớn có thể gây **trì trệ** (stagnation).

Cập nhật pheromone:

Bay hơi:

$$\tau_{ij}(t) = (1-\rho)\tau_{ij}(t) \tag{2.2}$$

Lắng đọng: mỗi kiến k lắng đọng pheromone trên giải pháp của nó:

$$\Delta\tau_{ij}^k = \frac{1}{C^k}$$

1.3.2. Các Biến thể Nâng cao của ACO

Từ AS, nhiều biến thể được phát triển:

- **MAX-MIN Ant System (MMAS)**, **Best-Worst Ant System (BWAS)**: Giới hạn giá trị pheromone hoặc chỉ cho phép một tập con kiến cập nhật dấu vết.
- **Ant Colony System (ACS):** Khác AS ở 3 điểm:
  1. Quy tắc lựa chọn *giả ngẫu nhiên* dựa trên tham số q_0
  2. Cập nhật pheromone cục bộ khi xây dựng lời giải
  3. Cập nhật toàn cục: chỉ lời giải tốt nhất (S^{bs}) được lắng đọng pheromone
- **Enhanced Ant Colony System (EACS):** Tăng cường khai thác bằng cách ưu tiên đi theo cạnh thuộc lời giải tốt nhất toàn cục (S^*).
- **EigenAnt (2013):** Dùng một kiến duy nhất, loại bỏ heuristic, chỉ dựa trên pheromone và bay hơi chọn lọc. Tránh stagnation và tự điều chỉnh để giữ tập trung vào đường đi tốt nhất.

2. Phương pháp giải quyết bài toán sử dụng Ant Colony Optimization

2.1. Ant Colony Optimization (ACO)

Ant Colony Optimization (ACO) là một metaheuristic lấy cảm hứng từ hành vi tập thể của loài kiến trong tự nhiên nhằm giải quyết các bài toán tối ưu tổ hợp. Ý tưởng cốt lõi là các kiến nhân tạo xây dựng nghiệm bằng cách di chuyển trên đồ thị biểu diễn bài toán. Quá trình tìm kiếm được hướng dẫn bởi hai thành phần: lượng pheromone τ_{ij} trên các cung (biểu thị kinh nghiệm tích lũy của bầy kiến) và heuristic η_{ij} đánh giá chất lượng cục bộ.

Xác suất để kiến k chọn đỉnh j kế tiếp từ đỉnh i tại vòng lặp t là:

$$p_{ij}^k(t) = \frac{[\tau_{ij}(t)]^\alpha \cdot [\eta_{ij}]^\beta}{\sum_{l \in N_i^k} [\tau_{il}(t)]^\alpha \cdot [\eta_{il}]^\beta} \tag{1}$$

trong đó:

- $\tau_{ij}(t)$: lượng pheromone trên cạnh (i, j) tại thời điểm t;
- η_{ij}: heuristic (thường là $\eta_{ij} = 1/d_{ij}$ với d_{ij} là chi phí);
- α, β: hệ số trọng số cho pheromone và heuristic;
- N_i^k: tập các đỉnh hợp lệ mà kiến k chưa đi qua từ i.

Sau mỗi vòng lặp, pheromone trên các cung được cập nhật gồm hai giai đoạn: bay hơi và cộng thêm pheromone do các kiến đóng góp:

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}^k \tag{2}$$

trong đó ρ là hệ số bay hơi và

$$\Delta\tau_{ij}^k = \begin{cases} \dfrac{Q}{L_k}, & \text{nếu cạnh } (i,j) \text{ nằm trên tour của kiến } k \\ 0, & \text{ngược lại} \end{cases}$$

với L_k là chi phí tour thứ k, Q là hằng số.

2.2. Ant System (AS)

Ant System (AS) là phiên bản ACO đầu tiên, được áp dụng hiệu quả cho các bài toán như TSP. Trong AS, tất cả các kiến hoạt động đồng thời và xây dựng nghiệm dựa trên xác suất (1). Sau khi hoàn thành các tour, pheromone được cập nhật toàn cục qua (2), cho phép các cung thuộc các tour ngắn hơn nhận nhiều pheromone, từ đó được ưu tiên ở các vòng tiếp theo.

Heuristic phổ biến trong AS là $\eta_{ij} = 1/d_{ij}$ (với d_{ij} là trọng số cung). Phương pháp bay hơi pheromone giảm ảnh hưởng của nghiệm cũ, gia tăng khả năng thoát khỏi cực tiểu cục bộ.

2.3. Ant Colony System (ACS)

Ant Colony System (ACS) cải tiến từ AS với hai cơ chế chính:

- **Chọn điểm tiếp theo**: ACS dùng quy tắc pseudo-ngẫu nhiên. Nếu số ngẫu nhiên $q \leq q_0$, kiến chọn cạnh tốt nhất theo công thức:

$$j = \arg\max_{l \in N_i^k} \left(\tau_{il}(t) \cdot [\eta_{il}]^\beta\right)$$

  còn lại chọn xác suất như AS.

- **Cập nhật pheromone**: cập nhật nội tuyến (*local update*) sau mỗi bước

$$\tau_{ij}(t) \leftarrow (1-\xi)\tau_{ij}(t) + \xi \cdot \tau_0$$

  với ξ là tham số cập nhật cục bộ, τ_0 là giá trị pheromone khởi tạo. Sau mỗi vòng, chỉ đường đi tốt nhất toàn cục mới được thưởng thêm pheromone:

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

  với $\Delta\tau_{ij}^{best} = Q/L_{best}$.

ACS cân bằng giữa khả năng khai thác nghiệm tốt nhất đã biết và duy trì khám phá không gian mới.

2.4. Sequential Ordering Problem (SOP)

Sequential Ordering Problem (SOP) tổng quát hóa bài toán TSP bằng cách thêm ràng buộc thứ tự giữa một số đỉnh. Cụ thể, cho đồ thị có hướng $G = (V, E)$, trọng số d_{ij} trên mỗi cung (i, j), hai đỉnh đặc biệt v_s (bắt đầu) và v_f (kết thúc), cùng tập các ràng buộc precedence C với mỗi $(u \prec v) \in C$ yêu cầu u phải đứng trước v trong hành trình.

Bài toán được phát biểu:

$$\min_{P} \sum_{(i,j)\in P} d_{ij}$$

với P: chuỗi các đỉnh đi từ v_s đến v_f, đi qua tất cả V, thỏa mãn:

$$\forall (u \prec v) \in C,\ u \text{ xuất hiện trước } v \text{ trên } P$$
$$\forall v \in V,\ v \text{ xuất hiện đúng một lần trong } P$$

Với ACO áp dụng SOP, tập các đỉnh có thể chọn tiếp được cập nhật chặt chẽ ở mỗi bước:

$$N_i^{\text{valid}} = \{j \in V \setminus P | \forall (j \succ l) \Rightarrow l \notin P\}$$

Pheromone τ_{ij} và heuristic η_{ij} thường thiết kế như với TSP, đồng thời tập N_i^{valid} bị ràng buộc bởi điều kiện precedence để đảm bảo mọi nghiệm xây dựng đều hợp lệ.

Có thể sử dụng các thuật toán tối ưu cục bộ sau giải pháp ban đầu để tăng chất lượng tour tìm được.

2.5. Hybrid Ant System for SOP (HAS-SOP)

HAS-SOP [1] là thuật toán hybrid kết hợp **Ant Colony System (ACS)** với local search đặc biệt **SOP-3-exchange**, được phát triển bởi Gambardella và Dorigo (2000) để giải quyết Sequential Ordering Problem. Điểm đột phá của HAS-SOP nằm ở khả năng xử lý hiệu quả các ràng buộc precedence phức tạp thông qua cơ chế local search

với độ phức tạp $O(n^3)$ và labeling procedure kiểm tra tính khả thi trong thời gian hằng số.

2.5.1. Công Thức Cốt Lõi

1. Quy Tắc Chuyển Tiếp (Transition Rule)

Với xác suất q_0 (deterministic exploitation), kiến chọn đỉnh tốt nhất:

$$j = \arg\max_{l \in N_i^k} \left(\tau_{il}(t) \cdot [\eta_{il}]^\beta\right) \tag{3}$$

Với xác suất $(1 - q_0)$ (probabilistic exploration), chọn theo quy tắc ACO tiêu chuẩn. Tham số q_0 được tính động: $q_0 = 1 - s/n$ với $s \approx 10$.

2. Cập Nhật Pheromone Hai Tầng

Local update sau mỗi bước di chuyển (i, j):

$$\tau_{ij} \leftarrow (1 - \xi)\tau_{ij} + \xi \cdot \tau_0 \tag{4}$$

với $\xi = 0.1$, $\tau_0 = (L_{\text{first}}/n)^{-1}$, nhằm "ăn mòn" pheromone trên cạnh đã đi qua để khuyến khích đa dạng nghiệm.

Global update chỉ cho tour tốt nhất:

$$\tau_{ij} \leftarrow (1 - \rho)\tau_{ij} + \rho \cdot \frac{1}{L_{\text{best}}} \tag{5}$$

với $\rho = 0.1$, tăng cường pheromone trên các cạnh của nghiệm tốt nhất toàn cục.

2.5.2. SOP-3-Exchange: Local Search Đột Phá

Thuật toán **SOP-3-exchange** là một kỹ thuật tìm kiếm cục bộ (local search) được thiết kế chuyên biệt cho bài toán Sắp xếp Tuần tự (Sequential Ordering Problem - SOP). Thuật toán này giải quyết hai vấn đề cốt lõi mà các phương pháp 3-opt truyền thống gặp phải khi áp dụng cho SOP: tính bất đối xứng của chi phí và các ràng buộc về thứ tự ưu tiên.

2.5.2.1. Kỹ thuật Bảo toàn Đường đi (Path Preserving)

Bài toán SOP là một biến thể của bài toán Người du lịch bất đối xứng (Asymmetric TSP), trong đó chi phí từ nút $i \to j$ khác với $j \to i$.

- Các thuật toán hoán đổi cạnh truyền thống (như 2-opt) thường đảo ngược chiều di chuyển của một đoạn đường con để nối lại tour. Điều này đòi hỏi tính toán lại chi phí rất tốn kém trong đồ thị bất đối xứng và dễ vi phạm ràng buộc.
- **Giải pháp:** SOP-3-exchange sử dụng kỹ thuật *Path-Preserving 3-exchange*. Thuật toán cắt 3 cạnh để tách hành trình thành các đoạn rời rạc, sau đó nối lại bằng cách hoán đổi vị trí của hai đoạn đường (*path_left* và *path_right*) mà **không đảo ngược** thứ tự các nút bên trong các đoạn đó.

2.5.2.2. Xử lý ràng buộc thứ tự (Handling Precedence Constraints)

Khi hoán đổi vị trí của *path_left* và *path_right* (đưa *path_right* lên trước *path_left*), ta có nguy cơ vi phạm ràng buộc: một nút nằm trong *path_right* có thể bị bắt buộc phải thực hiện *sau* một nút trong *path_left*.

- **Tìm kiếm Từ điển (Lexicographic Search):** Thay vì chọn ngẫu nhiên, thuật toán xây dựng *path_left* và *path_right* bằng cách mở rộng từng nút một (node-by-node). Nếu việc thêm một nút mới vào gây ra vi phạm ràng buộc, nhánh tìm kiếm đó sẽ bị cắt bỏ ngay lập tức.
- **Thủ tục Gán nhãn (Labeling Procedure):** Để tránh việc kiểm tra thủ công mọi cặp nút (tốn chi phí $O(n^2)$), thuật toán sử dụng cơ chế gán nhãn:
  1. Khi *path_left* được cố định, thuật toán đánh dấu tất cả các "hậu duệ" (successors) của các nút trong *path_left*.
  2. Khi mở rộng *path_right*, nếu gặp một nút đã bị đánh dấu, thuật toán biết ngay lập tức rằng việc hoán đổi là không khả thi.
  3. Nhờ đó, việc kiểm tra tính khả thi giảm xuống độ phức tạp hằng số $O(1)$.

2.5.2.3. Mã giả thuật toán (Pseudo-code)

Dưới đây là mã giả chi tiết cho quy trình SOP-3-exchange, bao gồm cả các bước gán nhãn và kiểm tra tính khả thi.

Algorithm 1 The SOP-3-exchange procedure [**?**]

Require:
a feasible solution given as a sequence $(0 \dots n)$
a sequential ordering problem G
a *SelectionCriterion* for h in $(sequential, dont\_look\_bit, dont\_push\_stack)$
a *WalkingCriterion* for i in $(3 - exchange, OR - exchange)$
an *ExchangeFirstCriterion* in (h, i, j)
Ensure: a new feasible solution that is 3-optimal

repeat
  $h \leftarrow 0$ ▷ h is set to the first node in the sequence
  while there is an available h **do** ▷ h loop
    $h \leftarrow SelectAccordingCriterion(SelectionCriterion, G)$
    $direction \leftarrow forward$ ▷ the search starts in forward direction
    $gain \leftarrow 0$; $i \leftarrow h + 1$; $j \leftarrow i + 1$; $SearchTerminated \leftarrow$ **false**
    while $SearchTerminated$ is **false do** ▷ i loop
      $feasible \leftarrow$ **true**
      if $direction = forward$ **and** $EndOfSequence(i, WalkingCriterion, G)$ **then**
        $direction \leftarrow backward$; $i \leftarrow h - 1$; $j \leftarrow i - 1$
      end if
      $UpdateGlobalVariables(h, i, direction, G)$
      while $feasible$ is **true do** ▷ j loop
        $feasible \leftarrow FeasibleExchange(h, i, j, direction, G)$
        $gain \leftarrow ComputeBestExchange(h, i, j, direction, G, feasible, gain)$
        if $gain > 0$ **and** $(ExchangeFirstCriterion = j)$ **then**
          goto PERFORM-EXCHANGE
        end if
        $j \leftarrow jWalkThroughTheSequence(h, i, j, direction, G)$
        $SearchTerminated \leftarrow f(j, direction, WalkingCriterion)$
      end while
      if $gain > 0$ **and** $(ExchangeFirstCriterion = i)$ **then**
        goto PERFORM-EXCHANGE
      end if
      $i \leftarrow iWalkThroughTheSequence(h, i, direction, G)$
      $SearchTerminated \leftarrow f(i, direction, WalkingCriterion)$
    end while
    PERFORM-EXCHANGE:
    if $gain > 0$ **then**
      $PerformExchange(h, i, j, direction, G)$
      goto REPEAT
    end if
  end while
until no improvement is found

3. Thí nghiệm và đánh giá kết quả

3.1. Giới thiệu

Đây là những thông số (parameters) sẽ được dùng trong thuật toán HAS-SOP L_nn: Đây là tổng khoảng cách được ước tính từ thuật toán Nearest Neighbor (Greedy)

Parameter	Values
Number of ants	10
Phi (Local Evaporation rate)	0.1
Rho (Global Evaporation rate)	0.1
Exploitation Factor	0.9
Beta (Heuristic factor)	2.0
Tau (initial hormone deposit)	: $\tau_0 = \frac{1}{n \cdot L_{nn}}$

Bảng 1: Các thông số ACO được thử nghiệm

heuristic. Phục vụ cho việc ước tính đường đi tốt nhất

3.2. Benchmark với các kích thước khác nhau

3.2.1. Giới thiệu

Ở phần này, chúng ta sẽ khám phá khả năng của bộ giải HAS-ACO bằng cách tìm kết quả các trường hợp SOP bằng tệp dữ liệu TSPLIB95 được tạo ra và phát triển bởi đại học Heidelberge (Đức) [3]

SOP Instance	Best known solution
ESC07	2125
ESC12	1675
ESC25	1681
ESC47	1288
ESC63	62
ESC78	18230
br17.10	55
br17.12	55

ft53.1	7531
ft53.2	8026
ft53.3	10262
ft53.4	14425
ft70.1	39313
ft70.2	[40101, 40419]
ft70.3	42535
ft70.4	53530
kro124p.1	[38762, 39420]
kro124p.2	[39841, 41336]
kro124p.3	[43904, 49499]
kro124p.4	[73021, 76103]
p43.1	28140
p43.2	28480
p43.3	28835
p43.4	83005
prob42	243
prob100	[1045, 1163]
rbg048a	351
rbg050c	467
rbg109a	1038
rbg150a	1750
rbg174a	2033
rbg253a	2950
rbg323a	3140
rbg341a	2568
rbg358a	2545
rbg378a	[2809, 2816]
ry48p.1	15805
ry48p.2	[16074, 16666]

ry48p.3	[19490, 19894]
ry48p.4	31446

Bảng 2: Các test-case SOP có trong TSPLIB95

Đối với các test-case có list dạng [a,b] (Ví dụ như ft70.2 [40101, 40419])

- Lower Bound (LB = 40,101): Đây là giới hạn lý thuyết thường được chứng minh bằng cách sử dụng Linear Programming hoặc Lagrangian relaxation, ý nghĩa rằng khoảng cách tối ưu không thể ngắn hơn 40,101.
- Upper Bound (UB = 40,419): Đây là khoảng cách tốt nhất mà thuật toán meta-heuristic hoặc heuristic đã tìm ra

3.2.2. Chi tiết chạy của một số tệp mẫu

Đây là thử nghiệm chạy thuật toán HAS-SOP trên tệp ft53.1.sop với số lần chạy là 2

```
Starting Run 1/2...
--- Iteration Logs ---
Iteration-0: Best Distance = 8226.0000
Iteration-5: Best Distance = 8133.0000
Iteration-10: Best Distance = 7848.0000
Iteration-15: Best Distance = 7848.0000
Iteration-20: Best Distance = 7848.0000
Iteration-25: Best Distance = 7848.0000
Terminating early at iteration 25 due to no improvement.

========================================
RUN 1 COMPLETED
Final Cost: 7848.0000
Sequence (54 nodes):
```

```
0 -> 3 -> 2 -> 15 -> 37 -> 36 -> 35 -> 40 -> 18 -> 19 -> 16 -> 11 -> 10 ->
    12 -> 14 -> 24 -> 22 -> 47 -> 42 -> 46 -> 41 -> 43 -> 45 -> 44 -> 34 ->
    32 -> 33 -> 31 -> 30 -> 1 -> 8 -> 5 -> 51 -> 49 -> 52 -> 50 -> 48 -> 29
    -> 28 -> 25 -> 27 -> 26 -> 7 -> 6 -> 9 -> 13 -> 17 -> 23 -> 20 -> 21 ->
    39 -> 38 -> 4 -> 53
========================================


Starting Run 2/2...
--- Iteration Logs ---
Iteration-0: Best Distance = 8002.0000
Iteration-5: Best Distance = 7909.0000
Iteration-10: Best Distance = 7909.0000
Iteration-15: Best Distance = 7889.0000
Iteration-20: Best Distance = 7715.0000
Iteration-25: Best Distance = 7715.0000
Iteration-30: Best Distance = 7680.0000
Iteration-35: Best Distance = 7622.0000
Iteration-40: Best Distance = 7622.0000
Iteration-45: Best Distance = 7622.0000
Terminating early at iteration 48 due to no improvement.

========================================
RUN 2 COMPLETED
Final Cost: 7622.0000
Sequence (54 nodes):
0 -> 36 -> 35 -> 40 -> 37 -> 18 -> 15 -> 39 -> 38 -> 45 -> 42 -> 46 -> 43
    -> 41 -> 47 -> 44 -> 34 -> 32 -> 33 -> 31 -> 30 -> 4 -> 2 -> 17 -> 16 ->
    23 -> 21 -> 8 -> 5 -> 51 -> 49 -> 52 -> 50 -> 48 -> 29 -> 28 -> 25 -> 27
    -> 26 -> 7 -> 6 -> 9 -> 3 -> 13 -> 11 -> 10 -> 12 -> 14 -> 20 -> 24 ->
    22 -> 19 -> 1 -> 53
Best Global Solution Found: 7622.0
```

==========

Và chúng ta thấy được ở 2 lần chạy kết quả tốt nhất cho việc tối ưu hoá đường đi là 7622

3.2.3. So sánh thuật toán HAS-SOP và MPO/AI trên tệp dữ liệu TSBLIB95

Thuật toán MPO/AI được giới thiệu ở [2], bài báo đề xuất một khung (framework) gồm hai bước để thiết kế toán tử lai ghép:

- **Xác định lược đồ chung lớn nhất (maximal common schema) của hai cha mẹ**.
- **Hoàn thiện giải pháp bằng cách sử dụng một thuật toán heuristic xây dựng (construction heuristic).**

Áp dụng khung thiết kế trên, tác giả phát triển hai toán tử mới:

- **MPO/AI (Maximum Partial Order/Arbitrary Insertion)**: Sử dụng thứ tự ưu tiên làm cơ sở chung (MPO) và hoàn thiện tour bằng heuristic chèn tùy ý (AI)
- **CST/NN (Common Sub-Tours/Nearest Neighbor)**: Sử dụng các đoạn tour chung (edges) làm cơ sở và kết nối chúng bằng heuristic láng giềng gần nhất (NN)

Và khi áp dụng vào bài toán Lập thứ tự tuần tự (SOP): MPO/AI đạt hiệu quả rất cao và đã tìm thấy các giải pháp mới tốt hơn những kết quả từng được biết trước đó cho nhiều trường hợp SOP

Sau đây là bảng so sánh giữa thuật toán MPO/AI và HAS-SOP trên tệp TSBLIB95 với số lần chạy là 5 cho mỗi testcase

Bảng 3: Comparison Results between MPO/AI and HAS-SOP with run = 5

PROB	MPO/AI+LS				HAS-SOP			
	Best	Avg.	Std.Dev.	Time(s)	Best	Avg.	Std.Dev.	Time(s)
ESC07	**2125**	**2125.0**	0.0	0.1	**2125**	**2125.0**	0.0	0.0
ESC11	**2075**	**2075.0**	0.0	0.1	**2075**	**2075.0**	0.0	0.1
ESC12	**1675**	**1675.0**	0.0	0.1	**1675**	**1675.0**	0.0	0.1
ESC25	1759	1862.0	145.7	0.5	**1684**	**1748.2**	46.7	0.4
ESC47	2231	2432.7	144.9	1.8	**1510**	**1568.4**	58.2	4.5
ESC63	**62**	**62.0**	0.0	2.3	**62**	**62.0**	0.0	3.2
ESC78	18800	18813.3	10.3	3.1	**18230**	**18338.0**	60.4	3.4
br17.10	**55**	**55.0**	0.0	0.2	**55**	**55.0**	0.0	0.1
br17.12	**55**	**55.0**	0.0	0.2	**55**	**55.0**	0.0	0.1
ft53.1	**7531**	7600.0	70.6	1.4	7569	**7587.0**	24.6	9.5
ft53.2	8123	**8161.7**	28.1	2.1	**8072**	8452.8	287.9	7.4
ft53.3	**10432**	**10611.3**	145.3	1.6	10635	10635.4	0.9	3.7
ft53.4	**14425**	14426.3	0.9	1.0	**14425**	**14425.0**	0.0	1.7
ft70.1	39649	39872.0	227.9	4.9	**39400**	**39455.2**	52.2	24.1
ft70.2	41460	41718.7	286.5	4.0	**40627**	**40903.6**	214.4	15.1
ft70.3	42997	43918.3	669.9	3.9	**42566**	**42895.6**	367.1	4.2
ft70.4	**53602**	**53717.3**	81.6	2.1	53629	53748.2	144.2	4.3
kro124p.1	40979	41688.0	508.0	8.3	**39455**	**40208.2**	529.7	46.7
kro124p.2	41836	42996.3	860.5	7.6	**41336**	**42157.6**	925.4	21.1
kro124p.3	51488	51816.0	367.4	6.5	**49576**	**50860.8**	858.1	11.9
kro124p.4	**76372**	77147.3	973.3	3.6	**76372**	**76951.2**	545.4	5.1
p43.1	**28140**	28156.7	23.6	1.8	**28140**	**28140.0**	0.0	1.6
p43.2	**28480**	28485.0	4.1	1.4	**28480**	**28484.0**	2.2	1.2
p43.3	**28835**	**28835.0**	0.0	1.4	28855	28855.0	0.0	1.0
p43.4	**83005**	**83020.0**	21.2	0.9	83050	83050.0	0.0	0.8
prob.100	2581	2736.3	186.1	12.4	**1499**	**1615.0**	112.1	11.4
prob.42	299	312.0	9.3	1.7	**258**	**263.2**	6.1	2.1
rbg048a	**351**	**351.0**	0.0	1.3	**351**	**351.0**	0.0	1.8
rbg050c	468	471.3	2.9	1.9	**467**	**467.4**	0.5	1.7
rbg109a	1041	1043.7	2.1	3.5	**1038**	**1042.8**	4.9	8.8
rbg150a	1756	1758.0	2.2	5.8	**1750**	**1752.8**	1.6	19.4
rbg174a	2043	2047.3	3.3	9.2	**2036**	**2040.2**	4.0	29.0
rbg253a	2966	2970.3	4.8	17.9	**2950**	**2953.2**	2.6	189.3
rbg323a	3207	3214.0	6.7	41.2	**3162**	**3171.2**	10.5	304.8
rbg341a	2671	2680.3	6.6	44.9	**2632**	**2642.0**	14.3	462.4
rbg358a	2715	2729.0	10.7	59.1	**2618**	**2641.0**	19.0	416.1
rbg378a	2894	2927.7	25.3	73.6	**2879**	**2903.4**	18.8	431.7
ry48p.1	15996	16058.3	53.7	1.8	**15805**	**15819.2**	19.7	3.4
ry48p.2	16693	**16818.7**	159.7	1.5	**16678**	16859.0	155.2	3.0
ry48p.3	20027	20251.3	195.8	1.4	**19894**	**20216.8**	241.7	1.7
ry48p.4	31614	31691.7	109.8	1.0	**31486**	**31519.2**	74.2	1.8
Total wins	**5**	**6**	—	29	**25**	**28**	—	10

Chúng ta có thể dễ dàng nhận thấy sự ưu việt của thuật toán HAS-SOP trong việc đi tìm lời giải tối ưu , song thuật toán HAS-SOP cũng có nhược điểm là thời gian chạy lâu hơn so sánh với MPO/AI

4. Đề xuất cải thiện cho bộ giải HAS-SOP

4.1. Kiến Trúc Cốt Lõi

Thiết kế cốt lõi của HAS-SOP gồm hai thành phần chính:

1. **ACS-SOP (Constructive Phase)**: Sinh ra các giải pháp khả thi bằng cách sử dụng pheromone trail và thông tin heuristic
2. **SOP-3-exchange (Local Optimizer)**: Thực hiện tìm kiếm địa phương sử dụng đổi cạnh bảo toàn đường dẫn (path-preserving-3-exchange) với procedura ghi nhãn (labeling procedure) để kiểm tra khả thi trong thời gian hằng số

4.2. Cải Tiến Thành Phần Học Tập: Phương Pháp Reinforcement Learning (RL)

Transfer Reinforcement Learning (TRL) + Automated RL [?]

Bước phát triển gần đây nhất là tích hợp Transfer Learning với Automated Reinforcement Learning (Auto_TL_RL), được công bố năm 2024.

Cơ Chế Hoạt Động:

- Sử dụng SARSA algorithm (State-Action-Reward-State-Action) và Q-learning
- Học chính sách tối ưu hóa giữa TSP và SOP

Kết Quả Thực Nghiệm:

- Cải thiện 85.7% so với phương pháp truyền thống
- Giảm thời gian tính toán 92.8% trên các test instance
- Áp dụng được cho cả ATSP và SOP

Ưu Điểm So Với HAS-SOP Gốc: Không cần tuning thủ công tham số như trong ACS, mà tự động hóa việc lựa chọn cấu hình thuật toán.

Automated Reinforcement Learning Simulator (AutoRL-Sim) [5]

Công cụ hỗ trợ việc tối ưu hóa hyperparameter:

- Sử dụng Response Surface Models (RSM) để tìm kiếm vùng tham số tối ưu
- Cho phép tuning tự động các tham số q_0, ρ, ϕ, s trong ACS-SOP
- Hỗ trợ thiết kế thực nghiệm (Design of Experiments)

Đề Xuất: Thay thế hoặc bổ sung quá trình tuning thủ công hiện tại bằng AutoRL system.

4.3. Cải Tiến Local Search: Thuật Toán Advanced k-opt

Lin-Kernighan Heuristic (LKH) Variable-Depth Search

Thuật toán Lin-Kernighan là một trong những phương pháp local search mạnh nhất cho TSP/ATSP.

Khác Biệt Với SOP-3-exchange:

- SOP-3-exchange là cố định (k=3), LKH là variable-depth
- LKH sử dụng adaptive heuristic rules để hướng dẫn tìm kiếm
- LKH dùng candidate list (5-nearest neighbors) để giới hạn tìm kiếm

Tại Sao Áp Dụng Được Cho SOP:

- LKH đã được thành công adapt cho Generalized TSP (GTSP) với multiple constraints
- Có thể mở rộng với lexicographic search strategy để xử lý precedence constraints

Đề Xuất Cụ Thể:

```
# SOP-Lin-Kernighan variant
# Thay SOP-3-exchange bang:
# - Variable-depth search thay vi fixed k=3
# - Maintain candidate lists (k-nearest neighbors, k=5-10)
# - Adaptive selection heuristics dua tren nganh canh
# - Sensitivity analysis de huong tim kiem
# Complexity: O(n^3) nhu SOP-3-exchange
# nhung co the O(n^2 log n) cho LK variant
```

4.4. Cải Tiến Framework Metaheuristic: Variable Neighborhood Search (VNS)

VNS as Meta-framework for HAS-SOP [6]

Variable Neighborhood Search là một metaheuristic framework cho phép thay đổi systematically các vùng lân cận.

Kiến Trúc HAS-SOP + VNS Hybrid:

```
procedure HAS-SOP-VNS
    Initialize pheromone trails
    k_min <- 1, k_max <- 5

    while termination_not_met do
        BuildSolutions (ACS-SOP)

        FOR k = k_min TO k_max do
            SELECT LocalSearch(k):
                k=1: SOP-2-exchange
                k=2: SOP-3-exchange (current)
                k=3: SOP-4-exchange
                k=4: SOP-LK (Lin-Kernighan variant)
                k=5: OR-exchange (variable-depth)

            LocalOptimize (solutions, LocalSearch(k))

            IF improved THEN
                k <- k_min
                break
            ENDIF
        END FOR

        UpdatePheromone
    end-while
```

```
end procedure
```

Lợi Ích:

- Thoát khỏi local optima hiệu quả hơn
- Kết hợp fine-grained (2-opt) và coarse-grained (LK) neighborhoods
- Adaptive neighborhood selection giảm tổng số iteration

4.5. Cải Tiến Cơ Chế Cập Nhật Pheromone

Adaptive Pheromone Volatilization [7]

Thay vì sử dụng evaporation rate cố định ($\phi = 0.1$), cải tiến sử dụng dynamic strategy:

$$\phi(t) = \phi_{\min} + (\phi_{\max} - \phi_{\min}) \cdot e^{-t/\tau} \tag{6}$$

- $\mathbf{t}$: iteration count
- τ: decay constant
- $\phi_{\min} \approx 0.05, \phi_{\max} \approx 0.15$

Giải Thích: Early iterations cần exploration (ϕ cao), late iterations cần exploitation (ϕ thấp).

Multi-Best Ant Pheromone Deposits

Thay vì chỉ best ant cập nhật pheromone, sử dụng ranked ants:

$$\text{FOR rank } r = 1 \text{ TO } m_{\text{best}} \text{ do} \tag{7}$$

$$w(r) = \frac{m_{\text{best}} - r + 1}{m_{\text{best}}} \tag{8}$$

$$\tau(\text{edges in path}_r) \mathrel{+}= w(r) \cdot \rho / L_r \tag{9}$$

Ưu Điểm:

- Giữ diversity cao hơn, giảm stagnation
- Nghiên cứu chỉ ra: 2 best ants ($w_1 = 10, w_2 = 5$) cho kết quả tốt

4.6. Cải Tiến Heuristic Information & Cost Function

Multi-Objective Heuristic Desirability [8]

Thay vì $\eta_{ij} = 1/t_{ij}$ (heuristic desirability đơn), sử dụng multi-criteria:

$$\eta_{ij} = (1/t_{ij})^{\alpha} \cdot (1/c_{ij})^{\beta} \cdot (\text{availability}_{ij})^{\gamma} \tag{10}$$

- t_{ij}: Waiting time (như HAS-SOP gốc)
- c_{ij}: Cost factor (có thể là complexity của edge)
- $\textbf{availability}_{ij}$: Feasibility score dựa trên precedence constraints
- $\alpha, \beta, \gamma \geq 0$: Tuning parameters

Ứng Dụng Thực Tế: Trong production scheduling, có thể tích hợp setup cost, resource utilization, energy consumption.

4.7. Cải Tiến Heuristic Lựa Chọn Node: Don't-Push Stack

HAS-SOP gốc đã sử dụng "don't push stack" trong SOP-3-exchange local search. Có thể mở rộng:

Adaptive Stack Management

```
procedure AdaptivePushStack
    stack <- InitializeWithAllNodes()
    improvement_history <- []

    while stack not empty do
        h <- Pop(stack)
```

```
        gain_before <- CurrentCost()
        PerformLocalSearch(h)
        gain_after <- CurrentCost()

        improvement = (gain_before - gain_after) / gain_before
        improvement_history.append(improvement)

        // Adaptive re-pushing based on history
        IF improvement > average(improvement_history[-5:]) THEN
            PushNeighbors(h, stack)
        ELSE
            PushNearestNeighbors(h, stack)
        END IF
    end while
end procedure
```

Benefit: Tập trung tìm kiếm vào vùng có tiềm năng cải tiến cao.

4.8. Cải Tiến Memetic Algorithm Framework

Population-based Memetic Design [9]

Kết hợp GA + HAS-SOP thay vì chỉ dùng một ant colony:

```
procedure HAS-SOP-Memetic
    // Phase 1: Population initialization
    population <- Generate_random_feasible_solutions(pop_size)
    population <- Apply_HAS_local_search(population)

    FOR generation = 1 TO max_gen do
        // Phase 2: Evolutionary operators
        offspring <- Selection(population)
        offspring <- Crossover(offspring)
        offspring <- Mutation(offspring)
```

```
        // Phase 3: Lamarckian learning (local search)
        offspring <- Apply_HAS_local_search(offspring)

        // Phase 4: Environmental selection
        population <- Replace_worst_solutions(population, offspring)

        // Phase 5: Pheromone update (from best of population)
        best_solution <- FindBest(population)
        UpdatePheromone(best_solution)
    END FOR

    // Phase 6: Final intensification
    Return Apply_intensive_local_search(BestInPopulation)
end procedure
```

Advantages:

- Implicit parallelization (multiple search directions)
- Better balance between diversification (GA) + intensification (HAS)
- Population-level memory retention

4.9. Bảng So Sánh Các Đề Xuất Cải Tiến

Đề Xuất	Mức Độ Phức Tạp	Cải Tiến Dự Kiến	Khó Triển Khai	Phù Hợp Với SOP
Transfer RL (Auto_TL_RL)	Cao	85.7%	Trung bình	Rất tốt
Lin-Kernighan Local Search	Cao	30-50%	Cao	Tốt (cần adapt)
Variable Neighborhood Search	Trung bình	20-30%	Thấp	Tốt
Adaptive Pheromone	Thấp	10-15%	Rất thấp	Tốt
Multi-Best Ant Update	Thấp	5-10%	Rất thấp	Tốt
Memetic Algorithm Wrapper	Cao	40-60%	Trung bình	Tốt
Multi-Objective Heuristic	Trung bình	15-25%	Trung bình	Tốt

Bảng 4: So Sánh Các Đề Xuất Cải Tiến HAS-SOP

5. Tổng kết và Kết luận

5.1. Tổng kết về hiệu quả của thuật toán HAS-SOP

Thông qua quá trình nghiên cứu và thực nghiệm trên bộ dữ liệu chuẩn TSPLIB95, báo cáo đã làm rõ hiệu quả của thuật toán **Hybrid Ant System for Sequential Ordering Problem (HAS-SOP)**. Đây là sự kết hợp giữa khả năng tìm kiếm toàn cục của hệ thống kiến (Ant Colony System) và khả năng tối ưu cục bộ chuyên biệt (SOP-3-exchange).

- **Chất lượng lời giải vượt trội:** HAS-SOP đã chứng minh khả năng tìm kiếm các lời giải tốt hơn so với thuật toán Di truyền MPO/AI (Maximum Partial Order/Arbitrary Insertion) trên đa số các bộ test. Cụ thể, HAS-SOP giành chiến thắng về chất lượng lời giải tốt nhất trong **25/40** trường hợp thử nghiệm so với chỉ 5/40 của MPO/AI

- **Khả năng phá vỡ kỷ lục (Upper Bounds):** Thuật toán đã tìm ra các cận trên mới (new upper bounds) cho nhiều bài toán khó mà các phương pháp trước đó chưa đạt được, đặc biệt là trên các tập dữ liệu lớn và phức tạp.

- **Độ ổn định cao:** Độ lệch chuẩn (Std. Dev) của HAS-SOP trong nhiều trường hợp thấp hơn so với đối thủ, cho thấy tính ổn định của thuật toán qua các lần chạy khác nhau[cite: 884].

Thành công của HAS-SOP không chỉ đến từ metaheuristic ACO mà phần lớn dựa vào thiết kế của bộ tìm kiếm cục bộ **SOP-3-exchange**. Hai yếu tố kỹ thuật quan trọng nhất đã được phân tích bao gồm:

1. **Kỹ thuật bảo toàn đường đi (Path Preserving):** Khắc phục nhược điểm của các phép lật cạnh truyền thống (như 2-opt) vốn không phù hợp với đồ thị bất đối xứng, giúp giảm thiểu chi phí tính toán lại hàm mục tiêu[cite: 853].

2. **Thủ tục gán nhãn (Labeling Procedure):** Đây là đóng góp quan trọng giúp kiểm tra tính khả thi của các ràng buộc thứ tự (precedence constraints) trong thời gian hằng số $O(1)$ thay vì $O(n^2)$, cho phép thuật toán mở rộng được trên các tập dữ liệu lớn mà không bị bùng nổ độ phức tạp tính toán[cite: 858, 860].

5.2. Hạn chế và Thách thức

Mặc dù đạt chất lượng lời giải cao, HAS-SOP vẫn tồn tại sự đánh đổi về mặt thời gian.Theo bảng so sánh thực nghiệm, thời gian chạy trung bình của HAS-SOP thường cao hơn so với MPO/AI (ví dụ: trên test case *rbg341a*, HAS-SOP tốn 462.4s so với 44.9s của MPO/AI). Điều này đặt ra yêu cầu cần tối ưu hóa hơn nữa về mặt hiệu năng để ứng dụng vào các bài toán thời gian thực.

5.3. Đề xuất cải tiến và Hướng phát triển

Dựa trên các phân tích về kiến trúc hiện tại và các nghiên cứu gần đây, báo cáo đề xuất các hướng cải tiến sau để nâng cao hiệu suất của bộ giải HAS-SOP:

- **Tích hợp Reinforcement Learning (AutoRL):** Thay thế việc tinh chỉnh tham số thủ công bằng cơ chế học tăng cường chuyển giao (Transfer RL), giúp tự động hóa việc chọn cấu hình tối ưu (ρ, α, β) cho từng loại dữ liệu, kỳ vọng giảm thời gian tính toán lên đến 92.8%[cite: 2, 4].

- **Cải tiến Local Search với Lin-Kernighan (LKH):** Nâng cấp từ SOP-3-exchange (độ sâu cố định $k = 3$) lên tìm kiếm độ sâu biến thiên (variable-depth search) theo phong cách Lin-Kernighan.Việc sử dụng danh sách ứng viên (candidate lists) và chiến lược thích nghi có thể giúp thoát khỏi cực tiểu địa phương tốt hơn.

- **Áp dụng khung Metaheuristic VNS:** Kết hợp Variable Neighborhood Search để thay đổi hệ thống các vùng lân cận (từ 2-opt đến k-opt) một cách có hệ thống, giúp cân bằng giữa khai thác và khám phá.

- **Chiến lược Pheromone thích nghi:** Sử dụng tỷ lệ bay hơi động $\phi(t)$ thay vì cố định, giúp giai đoạn đầu tập trung khám phá và giai đoạn sau tập trung hội tụ nhanh hơn.

Tóm lại, HAS-SOP là một phương pháp mạnh mẽ và hiệu quả cho bài toán SOP. Với các đề xuất cải tiến về tích hợp Học máy (Machine Learning) và các kỹ thuật Local Search nâng cao, thuật toán hứa hẹn sẽ giải quyết tốt hơn nữa bài toán tối ưu hóa tổ hợp phức tạp này trong tương lai.

Tài liệu

[1] Gambardella, Luca Maria Dorigo, Marco. (2000). An Ant Colony System Hybridized with a New Local Search for the Sequential Ordering Problem. INFORMS Journal on Computing. 12. 237-255. 10.1287/ijoc.12.3.237.12636.

[2] S. Chen and S.F. Smith. (1996) “Commonality and Genetic Algorithms.” tech. report CMU-RI-TR-96-27, Robotics Institute, Carnegie Mellon University, December, 1996

[3] http://comopt.ifi.uniheidelberg.de/software/TSPLIB95/SOP.html

[4] Transfer Reinforcement Learning for Combinatorial Optimization Problems, Algorithms 2024, 17(2), 87; https://doi.org/10.3390/a17020087

[5] AutoRL-Sim: Automated Reinforcement Learning Simulator for Combinatorial Optimization Problems https://doi.org/10.3390/modelling5030055

[6] Variable neighborhood search: Principles and applications https://doi.org/10.1016/S0377-2217(00)00100-4

[7] L. Cheng, K. Wang, L. Wei, Y. Liang and P. Song, "A Novel Automatic Voltage Control Strategy Based on Adaptive Pheromone Update Improved Ant Colony Algorithm," 2022 2nd International Conference on Electrical Engineering and Mechatronics Technology (ICEEMT), Hangzhou, China, 2022, pp. 232-236, doi: 10.1109/ICEEMT56362.2022.9862803.

[8] Pareto Ant Colony Optimization: A Metaheuristic Approach to Multiobjective Portfolio Selection https://doi.org/10.1023/B:ANOR.0000039513.99038.c6

[9] https://link.springer.com/article/10.1007/s00521-025-11171-zciteas https://doi.org/10.1007/s00521-025-11171-z