So sánh hiệu quả các thuật toán tối ưu trong bài toán TSP (Traveling Salesman Problem)

N.Tiến Đạt N.Thành Trung N.Thị Ánh N. Khánh Đô

HUS
VNU UNIVERSITY OF SCIENCE

MATHEMATICS
INFORMATICS
MECHANICS

Nội dung

Nội dung

Bài toán Người Du Lịch (TSP)

Mô tả chung

Bài toán Người Du Lịch (TSP - **Traveling Salesman Problem**) là một bài toán tối ưu tổ hợp nổi bật, với mục tiêu tìm ra lộ trình ngắn nhất mà một người du lịch cần đi qua tất cả các thành phố một lần và quay lại điểm xuất phát.

- Các thành phố được mô hình hóa dưới dạng đồ thị, trong đó các thành phố là các đỉnh và các con đường nối các thành phố là các cạnh.
- Trọng số của mỗi cạnh tương ứng với khoảng cách giữa các thành phố.

Mô hình hóa bài toán TSP

Mô hình đồ thị

- Nếu giữa hai thành phố không có đường đi trực tiếp, trọng số của cạnh này sẽ được coi là một giá trị rất lớn.
- Mục tiêu của bài toán TSP là tìm một chu trình Hamilton ngắn nhất, tức là tìm ra lộ trình khép kín sao cho tổng quãng đường đi qua tất cả các thành phố là ngắn nhất.

Đặc điểm của bài toán TSP

Tính chất NP-khó

- TSP là bài toán thuộc loại **NP-khó**, có nghĩa là không có thuật toán hiệu quả nào có thể giải quyết bài toán này trong thời gian đa thức với kích thước đầu vào lớn.
- Đối với bài toán cỡ nhỏ, có thể sử dụng phương pháp tìm kiếm vét cạn để tìm giải pháp tối ưu.
- Khi kích thước bài toán lớn, các phương pháp tìm kiếm vét cạn không thể đáp ứng yêu cầu về thời gian tính toán hợp lý.
- Vì vậy, các giải pháp gần đúng hoặc đủ tốt trở thành nhu cầu thiết yếu trong các ứng dụng thực tế.

Khả năng áp dụng thực tế

TSP có ứng dụng rộng rãi trong logistics, vận tải, và các lĩnh vực cần tối ưu hóa đường đi, ví dụ như tối ưu hóa lộ trình giao hàng.

Mục tiêu nghiên cứu và các thuật toán so sánh

Mục tiêu nghiên cứu

So sánh hiệu quả của 4 thuật toán nổi bật trong việc giải quyết bài toán TSP:

- **ACO (Ant Colony Optimization)**
- **PSO (Particle Swarm Optimization)**
- **Greedy Algorithm (Thuật toán tham lam)**
- **Branch and Bound (Thuật toán nhánh cận)**
- **Brute Force (Thuật toán vét cạn)**

Tiêu chí đánh giá

Các thuật toán sẽ được đánh giá theo các yếu tố:

- Độ chính xác
- Tốc độ hội tụ
- Khả năng giải quyết bài toán với số lượng thành phố lớn

Nội dung

Tập Dữ Liệu Sử Dụng Trong Thực Nghiệm

Để đánh giá hiệu suất và khả năng mở rộng của các thuật toán giải bài toán người bán hàng (TSP), nhóm đã sử dụng một tập hợp các tệp dữ liệu có tên `cities_N`, trong đó:

$$N \in \{8, 16, 20, 22, 32, 40, 48, 51, 64\}$$

Mỗi tệp tương ứng với một bài toán TSP gồm N thành phố.

Cấu trúc dữ liệu:

- Dữ liệu được lưu dưới dạng danh sách các cặp tọa độ (x, y) trong mặt phẳng hai chiều.
- Mỗi cặp tọa độ đại diện cho vị trí của một thành phố.

Ma trận khoảng cách giữa các thành phố được xây dựng theo công thức Euclid:

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

Trong đó d_{ij} là khoảng cách giữa thành phố i và j; (x_i, y_i), (x_j, y_j) là tọa độ tương ứng.

Nội dung

Giới thiệu thuật toán đàn kiến (ACO)

Hành vi tự nhiên của đàn kiến

- Thuật toán đàn kiến (ACO) mô phỏng hành vi tự nhiên của bầy kiến trong việc tìm kiếm thức ăn.
- Kiến sống bầy đàn và giao tiếp qua pheromone - một chất đặc biệt mà chúng để lại trên đường đi.
- Khi một con kiến di chuyển qua một đoạn đường, nó để lại pheromone. Lượng pheromone này sẽ tăng lên nếu có nhiều kiến đi qua cùng đoạn đó.
- Các con kiến khác sẽ ưu tiên chọn đường có lượng pheromone cao, coi đó là con đường tốt nhất.

Ứng dụng trong bài toán TSP

Thuật toán sử dụng hành vi này để tối ưu lộ trình di chuyển qua các thành phố trong bài toán Người Du Lịch (TSP), giúp tìm ra con đường ngắn nhất.

Hàm Xác Suất Lựa Chọn Thành Phố Tiếp Theo

Xác suất để một kiến chọn thành phố tiếp theo được biểu diễn như sau:

$$p_{ij}^{k}(t) = \begin{cases} \dfrac{[\tau_{ij}(t)]^{\alpha} \cdot [\eta_{ij}(t)]^{\beta}}{\sum_{s \in \text{allow}_k} [\tau_{is}(t)]^{\alpha} \cdot [\eta_{is}(t)]^{\beta}}, & \text{nếu } j \in \text{allow}_k \\ 0, & \text{nếu } j \notin \text{allow}_k \end{cases}$$

Trong đó:

- allow_k: Tập các thành phố mà kiến k chưa thăm và có thể chọn tiếp theo.
- $\tau_{ij}(t)$: Lượng pheromone trên cạnh nối giữa thành phố i và j tại thời điểm t.
- $\eta_{ij}(t)$: Mức hấp dẫn (heuristic desirability), thường được định nghĩa là $\eta_{ij}(t) = 1/d_{ij}$.
- α: Tham số điều chỉnh ảnh hưởng của pheromone.
- β: Tham số điều chỉnh ảnh hưởng của yếu tố khoảng cách.
- Mẫu số: Đảm bảo chuẩn hóa xác suất sao cho tổng xác suất bằng 1.

Sự bay hơi và tích lũy pheromone

Sự bay hơi hoặc tích lũy liên tục của pheromone được mô tả bằng các công thức lặp sau:

$$\begin{cases} \tau_{ij}(t+1) = \rho \times \tau_{ij}(t) + \Delta\tau_{ij}, \quad 0 < \rho < 1 \\ \Delta\tau_{ij}(t, t+1) = \sum_{k=1}^{M} \Delta\tau_{ij}^{k}(t, t+1) \\ \Delta\tau_{ij}^{k}(t, t+1) = Q_t \times Z(N_c, k) \end{cases}$$

Trong đó:

- ρ: Hệ số pheromone còn sót lại, với $0 < \rho < 1$.
- $\Delta\tau_{ij}(t, t+1)$: Tổng pheromone được thả bởi tất cả các con kiến đi qua đường ij.
- $\Delta\tau_{ij}^{k}(t, t+1)$: Lượng pheromone mới được thả bởi con kiến thứ k trên đường ij.
- Q_t: Hệ số tăng cường pheromone, quyết định lượng pheromone tối đa mà một con kiến có thể thả.
- $Z(N_c, k)$: Hiệu quả của giải pháp mà kiến thứ k tìm được trong vòng lặp N_c.

Quy Trình Thuật Toán Bầy Kiến (ACO)

1. **Khởi tạo:**
   - Phân bố ngẫu nhiên các con kiến vào các thành phố ban đầu.
   - Khởi tạo lượng pheromone trên tất cả các cạnh ở mức cơ sở.
2. **Xây dựng tuyến đường:**
   - Mỗi con kiến lần lượt chọn thành phố tiếp theo để đi qua, dựa trên xác suất phụ thuộc vào lượng pheromone và khoảng cách (mức hấp dẫn).
3. **Cập nhật pheromone:**
   - Sau khi tất cả kiến hoàn thành hành trình, cập nhật lại pheromone trên các cạnh dựa trên chất lượng tuyến đường (thường là độ ngắn của hành trình).
4. **Lặp lại:**
   - Tiếp tục quá trình xây dựng hành trình và cập nhật pheromone qua nhiều vòng lặp, cho đến khi đạt đến số chu kỳ tối đa hoặc điều kiện dừng được thỏa mãn.

Giới thiệu về thuật toán PSO

Particle Swarm Optimization (PSO) là một thuật toán tối ưu bầy đàn, được truyền cảm hứng từ hành vi di chuyển của các sinh vật trong thiên nhiên, như đàn chim hoặc đàn cá.

Trong PSO, mỗi cá thể trong bầy đàn (hay ”hạt”) đại diện cho một giải pháp có thể có, và chúng tìm kiếm giải pháp tối ưu qua các lần di chuyển trong không gian tìm kiếm.

Mỗi hạt di chuyển dựa trên:

- **Kinh nghiệm của chính hạt**: Điều chỉnh tốc độ và hướng dựa trên vị trí tốt nhất mà hạt đã tìm thấy.
- **Thông tin từ các hạt khác**: Sự tương tác và chia sẻ thông tin giữa các hạt giúp bầy đàn cải thiện kết quả.

PSO đã được áp dụng rộng rãi trong việc giải quyết các bài toán tối ưu phức tạp, bao gồm bài toán người bán hàng TSP.

Mô hình PSO trong bài toán TSP

Để áp dụng thuật toán PSO vào bài toán TSP, ta cần xác định:

- **Vị trí** (P) và **Vận tốc** (V) của cá thể.
- **Hàm thích nghi** (fitness function).
- Vị trí tối ưu cục bộ (Pbest)
- Vị trí tối ưu toàn cục (Gbest) của bầy đàn.

Quần thể gồm Y cá thể, mỗi cá thể x có M_x chiều không gian và lặp N lần để tìm kiếm giải pháp tối ưu.
Vị trí và vận tốc của cá thể tại vòng lặp thứ i được biểu diễn như sau:

$$P_{ix} = \{p_{ix}^1, p_{ix}^2, \ldots, p_{ix}^{M_x}\} \tag{1}$$

$$V_{ix} = \{v_{ix}^1, v_{ix}^2, \ldots, v_{ix}^{M_x}\} \tag{2}$$

Trong đó:

- p_{ix}^j là vị trí của cá thể x ở vòng lặp i tại chiều j, $(j = 1 \ldots M_x)$.
- v_{ix}^j là vận tốc của các thể x ở vòng lặp i tại chiều j, $(j = 1 \ldots M_x)$.

Cập nhật vị trí tối ưu cục bộ trong PSO

Vị trí tối ưu cục bộ của mỗi cá thể được cập nhật qua các bước sau:

Nếu giá trị hàm thích nghi tại vị trí $j+1$ tốt hơn hoặc bằng giá trị tại vị trí j, thì cập nhật vị trí tối ưu cục bộ của cá thể tại chiều $j+1$:

$$p_{ix}^{j+1} = \begin{cases} p_{ix}^{j+1} & \text{nếu } F(p_{ix}^{j+1}) \geq F(p_{ix}^{j}) \\ p_{ix}^{j} & \text{ngược lại} \end{cases}$$

Giải thích

- $F(p_{ix}^{j})$: Giá trị hàm thích nghi của cá thể x tại vị trí j.
- p_{ix}^{j}: Vị trí của cá thể x tại vòng lặp thứ i và chiều j.
- Nếu $F(p_{ix}^{j+1})$ lớn hơn hoặc bằng $F(p_{ix}^{j})$, thì cá thể cập nhật vị trí tại $j+1$; ngược lại giữ lại giá trị trước đó.

Cập nhật vị trí tối ưu toàn cục trong PSO

Sau khi cập nhật vị trí tối ưu cục bộ của mỗi cá thể, ta tiến hành cập nhật vị trí tối ưu toàn cục (Gbest) cho toàn bộ bầy đàn.

Vị trí tối ưu cục bộ của cá thể (Pbest) được xác định:

$$Pbest_x = \max_{x=1,\ldots,Y, j=1,\ldots,M_x} \{p_{ix}^j\}$$

Vị trí tối ưu toàn cục của đàn được xác định từ vị trí tối ưu cục bộ lớn nhất của các cá thể:

$$Gbest = \max\{Pbest_x\}$$

Trong đó

- $Pbest_x$: Vị trí tối ưu cục bộ của cá thể x, là giải pháp tốt nhất mà cá thể đó tìm được.
- *Gbest*: Vị trí tối ưu toàn cục của bầy đàn, là giải pháp tốt nhất được tìm thấy trong tất cả các cá thể.

Cập nhật vị trí trong PSO

Sau khi cập nhật vận tốc, vị trí của cá thể được cập nhật theo công thức sau:

Cập nhật vị trí

Vị trí của cá thể x tại vòng lặp tiếp theo được cập nhật theo công thức:

$$pos_x^{j+1} = pos_x^j + v_x^j$$

Giải thích

- pos_x^j: Vị trí hiện tại của cá thể x tại chiều j.
- v_x^j: Vận tốc của cá thể x tại chiều j.
- Vị trí mới của cá thể được tính bằng cách cộng vận tốc hiện tại với vị trí trước đó.

Cập nhật vận tốc trong PSO

Vận tốc của cá thể được cập nhật theo công thức sau để điều chỉnh chuyển động của cá thể trong không gian tìm kiếm.

Cập nhật vận tốc

Vận tốc của cá thể x tại vòng lặp tiếp theo được cập nhật theo công thức:

$$v_x^{j+1} = v_x^j + c_1 z_1 (Pbest_x - pos_x^j) + c_2 z_2 (Gbest - pos_x^j)$$

Trong đó

- v_x^j: Vận tốc của cá thể x tại chiều j.
- $Pbest_x$: Vị trí tối ưu cục bộ của cá thể x.
- $Gbest$: Vị trí tối ưu toàn cục của bầy đàn.
- c_1, c_2: Hệ số gia tốc.
- z_1, z_2: Các số ngẫu nhiên giữa 0 và 1.

Quy Trình Thuật Toán PSO trong Bài Toán TSP

Để áp dụng thuật toán PSO vào bài toán người bán hàng, quy trình bao gồm các bước cơ bản sau:

1. Khởi tạo

- Khởi tạo vị trí và vận tốc của mỗi cá thể (hạt) trong bầy đàn một cách ngẫu nhiên.
- Vị trí ban đầu của các cá thể sẽ đại diện cho các giải pháp tiềm năng, trong khi vận tốc quyết định bước nhảy của mỗi cá thể trong không gian tìm kiếm.

2. Đánh giá

- Mỗi cá thể sẽ tính giá trị hàm thích nghi (fitness) dựa trên tuyến đường mà nó đang di chuyển.
- Giá trị hàm thích nghi sẽ phản ánh chất lượng của giải pháp (ví dụ: độ dài hành trình trong bài toán TSP).

Quy Trình Thuật Toán PSO trong Bài Toán TSP

3. Cập nhật vị trí và vận tốc

- Cập nhật vận tốc và vị trí của mỗi cá thể dựa trên hai yếu tố chính:
  - Vị trí tốt nhất của cá thể đó (Pbest).
  - Vị trí tốt nhất của bầy đàn (Gbest).
- Quy trình này giúp cá thể điều chỉnh di chuyển sao cho tiến gần hơn đến giải pháp tối ưu.

4. Lặp lại

- Lặp lại quá trình đánh giá, cập nhật vị trí và vận tốc cho đến khi đạt được số lần lặp tối đa hoặc điều kiện dừng được thỏa mãn.
- Quá trình này sẽ dẫn đến việc tối ưu dần dần trong không gian tìm kiếm, giúp thuật toán tìm ra giải pháp tối ưu cho bài toán TSP.

Giới thiệu

Bài toán người bán hàng đi qua tất cả các thành phố sao cho tổng quãng đường di chuyển là ngắn nhất có thể được gọi là bài toán TSP (Traveling Salesman Problem). Một trong những phương pháp giải quyết bài toán này là sử dụng thuật toán tham lam (greedy algorithm). Thuật toán tham lam là một phương pháp giải quyết bài toán bằng cách chọn lựa lựa chọn tối ưu trong từng bước một cách cục bộ, với hy vọng rằng những lựa chọn này sẽ dẫn đến một lời giải tối ưu toàn cục.

Cách thức áp dụng Thuật toán Tham lam

Thuật toán tham lam giải quyết bài toán TSP theo các bước sau:

1. **Khởi tạo:** Bắt đầu từ một thành phố bất kỳ.
2. **Chọn thành phố gần nhất:** Tại mỗi bước, thuật toán sẽ chọn thành phố chưa thăm gần nhất với thành phố hiện tại và di chuyển đến thành phố đó.
3. **Lặp lại:** Quá trình này được lặp lại cho đến khi tất cả các thành phố đã được thăm.
4. **Quay lại điểm xuất phát:** Sau khi thăm hết các thành phố, thuật toán quay lại thành phố ban đầu để hoàn thành hành trình.

Để tìm thành phố gần nhất từ một thành phố hiện tại, thuật toán sẽ tính khoảng cách giữa thành phố hiện tại và tất cả các thành phố chưa thăm. Thành phố có khoảng cách nhỏ nhất sẽ được chọn làm điểm đến tiếp theo.

Ví dụ

Giả sử có 5 thành phố $1, 2, 3, 4, 5$ với ma trận chi phí giữa các thành phố như sau:

$$C = \begin{bmatrix} \infty & 3 & 14 & 18 & 15 \\ 3 & \infty & 4 & 22 & 20 \\ 17 & 9 & \infty & 16 & 4 \\ 6 & 2 & 7 & \infty & 12 \\ 9 & 15 & 11 & 5 & \infty \end{bmatrix}$$

1. **Bước 1:** Bắt đầu từ thành phố 1. Các thành phố chưa thăm là $2, 3, 4, 5$. Tính toán khoảng cách từ thành phố 1 đến các thành phố chưa thăm:

$$1 \to 2 = 3, \quad 1 \to 3 = 14, \quad 1 \to 4 = 18, \quad 1 \to 5 = 15$$

Thành phố gần nhất là 2, thuật toán đi từ $1 \to 2$.

Ví dụ

1. **Bước 2:** Từ thành phố 2, các thành phố chưa thăm là 3, 4, 5:

$$2 \to 3 = 4, \quad 2 \to 4 = 22, \quad 2 \to 5 = 20$$

   Thành phố gần nhất là 3, thuật toán đi từ $2 \to 3$.
2. **Bước 3:** Từ thành phố 3, các thành phố chưa thăm là 4, 5:

$$3 \to 4 = 16, \quad 3 \to 5 = 4$$

   Thành phố gần nhất là 5, thuật toán đi từ $3 \to 5$.
3. **Bước 4:** Từ thành phố 5, còn lại thành phố 4:

$$5 \to 4 = 5$$

   Đi từ $5 \to 4$.
4. **Bước 5:** Quay về thành phố ban đầu:

$$4 \to 1 = 6$$

Tuyến đường: $1 \to 2 \to 3 \to 5 \to 4 \to 1$ với tổng chi phí là:

$$3 + 4 + 4 + 5 + 6 = \boxed{22}$$

Giới thiệu

Giải thuật Nhánh Cận (Branch and Bound) là một phương pháp hiệu quả trong việc giải các bài toán tối ưu có không gian lời giải lớn. Bằng cách kết hợp giữa chiến lược chia nhỏ bài toán (phân nhánh) và sử dụng các giới hạn (cận) để loại bỏ sớm những hướng đi không khả thi, thuật toán giúp giảm đáng kể số lượng phương án cần xét, từ đó tìm ra lời giải tối ưu một cách thông minh và tiết kiệm thời gian.

Ý tưởng

Ý tưởng chính của giải thuật Nhánh Cận là không cần xét hết mọi khả năng, mà chỉ tập trung vào những nhánh hứa hẹn có thể dẫn đến lời giải tốt nhất.
Để làm được điều đó, ta cần xây dựng một hàm đánh giá cận nhằm ước lượng giá trị tốt nhất mà một nhánh có thể đạt được. Nếu cận này không tốt hơn lời giải tạm thời đang có, ta có thể loại bỏ nhánh đó, giúp tiết kiệm rất nhiều thời gian tính toán.

- Cận dưới được đánh giá bằng cách ước lượng chi phí nhỏ nhất còn lại để hoàn thành hành trình.
- Tính tổng khoảng cách ngắn nhất từ mỗi thành phố chưa thăm đến một thành phố khác — vì đây là chi phí tối thiểu cần đi qua.
- Cộng thêm khoảng cách ngắn nhất từ các thành phố đã thăm đến các thành phố chưa thăm — để đảm bảo hành trình kết nối với phần chưa đi.
- Tổng các khoảng cách này được chia đôi, vì mỗi cạnh được tính hai lần trong quá trình ước lượng (một lần khi xét từ thành phố đã thăm và một lần khi xét từ thành phố chưa thăm).
- Cận dưới này giúp loại bỏ sớm các nhánh không thể cho kết quả tốt hơn lời giải hiện tại.

Ví dụ

Cho ma trận chi phí C của bài toán người bán hàng:

$$C = \begin{bmatrix} \infty & 3 & 14 & 18 & 15 \\ 3 & \infty & 4 & 22 & 20 \\ 17 & 9 & \infty & 16 & 4 \\ 6 & 2 & 7 & \infty & 12 \\ 9 & 15 & 11 & 5 & \infty \end{bmatrix}$$

Giả sử ta đã đi qua các thành phố 1 và 2.

Bước 1: Tính khoảng cách tối thiểu từ các thành phố chưa thăm đến các thành phố khác

Các thành phố chưa thăm là: 3, 4, 5.

- Thành phố 3: $\min(17, 9, 16, 4) = 4$
- Thành phố 4: $\min(6, 2, 7, 12) = 2$
- Thành phố 5: $\min(9, 15, 11, 5) = 5$

$\Rightarrow$ Tổng tối thiểu từ các thành phố chưa thăm $= 4 + 2 + 5 = 11$

Bước 2: Tính khoảng cách tối thiểu từ các thành phố đã thăm đến các thành phố chưa thăm

Các thành phố đã thăm là: 1, 2. Các thành phố chưa thăm là: 3, 4, 5.

- Từ thành phố 1 đến 3, 4, 5: $\min(14, 18, 15) = 14$
- Từ thành phố 2 đến 3, 4, 5: $\min(4, 22, 20) = 4$

$$\Rightarrow \text{Tổng tối thiểu từ các thành phố đã thăm} = 14 + 4 = 18$$

Bước 3: Tính cận dưới

$$\text{Cận dưới} = \frac{\text{Tổng bước 1} + \text{Tổng bước 2}}{2} = \frac{11 + 18}{2} = \frac{29}{2} = 14.5$$

$$\Rightarrow \boxed{\text{Cận dưới cho hành trình hiện tại là 14.5}}$$

Ý tưởng

Ý tưởng chính của thuật toán vét cạn là duyệt qua tất cả các hoán vị có thể của các thành phố, tính tổng khoảng cách của mỗi hành trình và tìm hành trình có tổng chi phí nhỏ nhất.

- Bắt đầu từ một thành phố.
- Sinh ra tất cả các hoán vị của các thành phố còn lại.
- Với mỗi hoán vị, tính tổng quãng đường đi qua tất cả các thành phố theo thứ tự đó, rồi quay về thành phố xuất phát.
- Chọn hành trình có tổng khoảng cách nhỏ nhất.

Thuật toán này có độ phức tạp thời gian là $O(n!)$, với n là số lượng thành phố.

Ví dụ

- Giả sử có 5 thành phố: $1, 2, 3, 4, 5$, với ma trận chi phí đi lại giữa các thành phố như sau:

$$C = \begin{bmatrix} \infty & 3 & 14 & 18 & 15 \\ 3 & \infty & 4 & 22 & 20 \\ 17 & 9 & \infty & 16 & 4 \\ 6 & 2 & 7 & \infty & 12 \\ 9 & 15 & 11 & 5 & \infty \end{bmatrix}$$

Ta sẽ xét tất cả các hành trình bắt đầu từ thành phố 1, đi qua các thành phố còn lại theo mọi thứ tự, rồi quay về thành phố 1.

Ví dụ

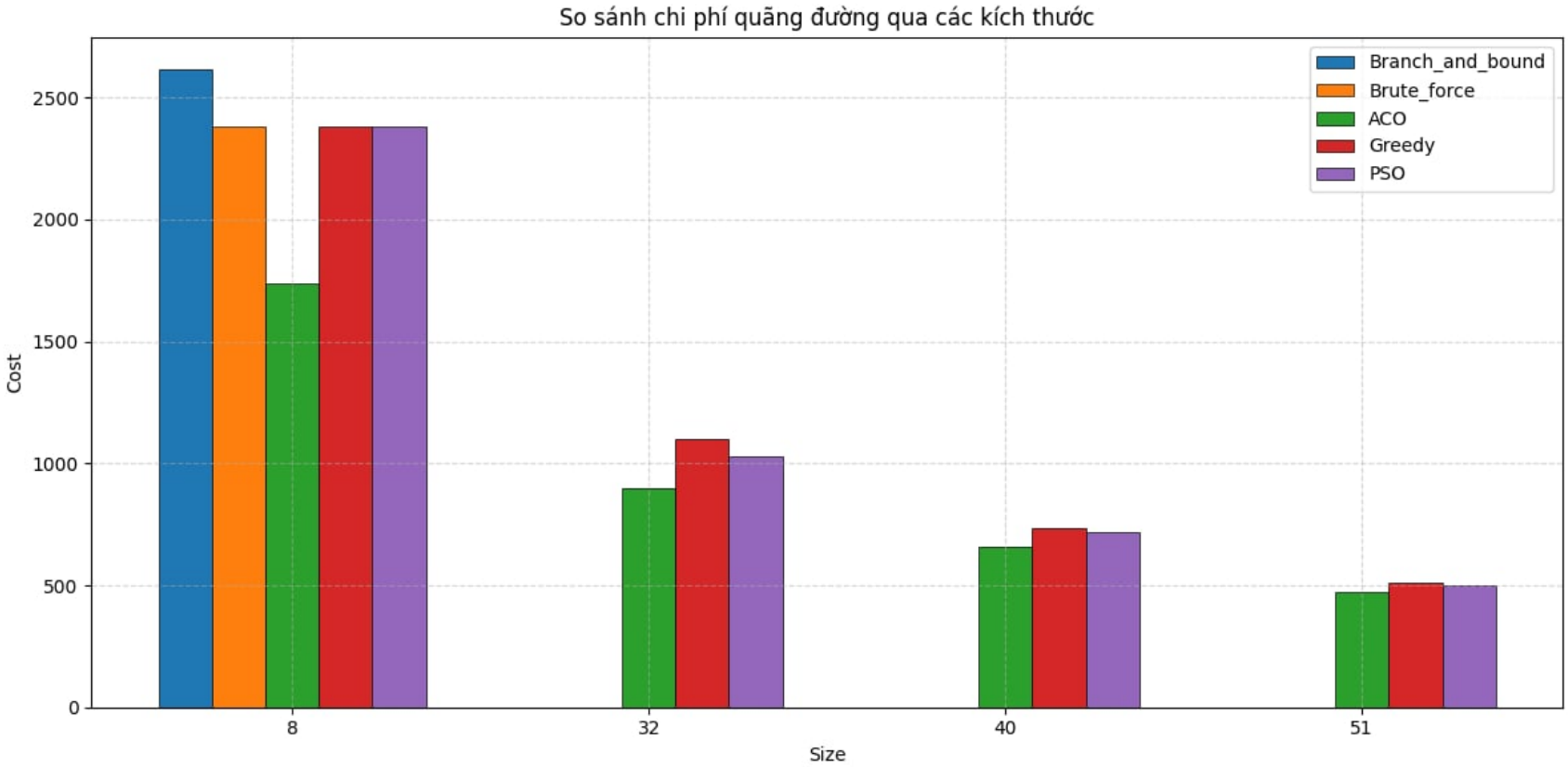
Một số hành trình có thể:

- $1 \to 2 \to 3 \to 4 \to 5 \to 1$: $3 + 4 + 16 + 12 + 9 = 44$
- $1 \to 3 \to 2 \to 4 \to 5 \to 1$: $14 + 9 + 22 + 12 + 9 = 66$
- $1 \to 5 \to 4 \to 3 \to 2 \to 1$: $15 + 5 + 7 + 9 + 3 = 39$
- $1 \to 4 \to 2 \to 3 \to 5 \to 1$: $18 + 2 + 4 + 4 + 9 = 37$

Hành trình tối ưu (ví dụ): $1 \to 4 \to 2 \to 3 \to 5 \to 1$, tổng chi phí là $\boxed{37}$

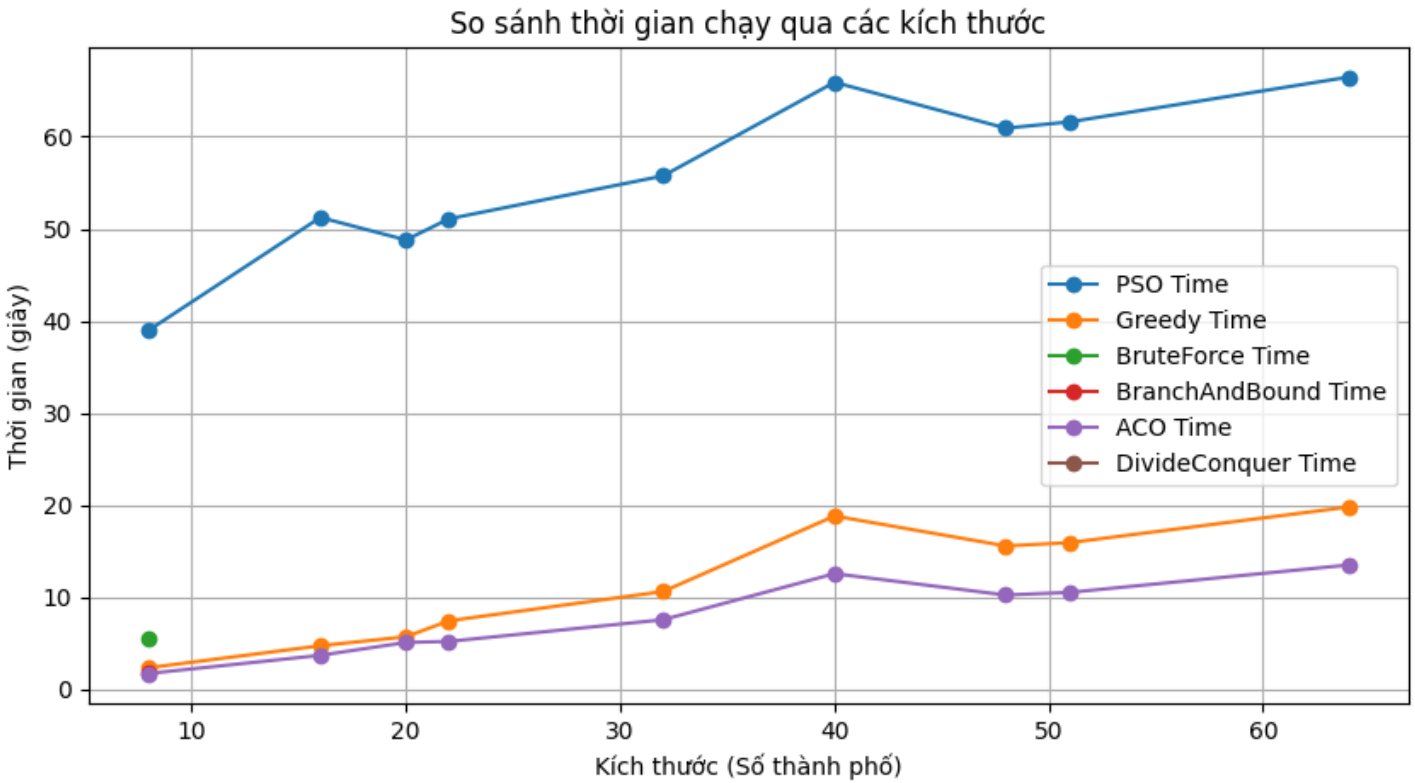
Nội dung

So sánh chi phí giữa các thuật toán



Biểu đồ so sánh chi phí giữa các thuật toán

So sánh chi phí giữa các thuật toán



Biểu đồ so sánh thời gian giữa các thuật toán

Nội dung

Tài liệu tham khảo

1. Yong Wang and Zunpu Han (August 2021) Ant colony optimization for traveling salesman problem based on parameters optimization
2. Analytics Vidhya. (2021, October). An Introduction to Particle Swarm Optimization Algorithm
3. A. T. B. University (2016) Implementation of Greedy Algorithm in Travel Salesman Problem
4. S. M. S. R. (2018) Travelling Salesman Problem Using Branch And Bound Technique