SO SÁNH HIỆU QUẢ CỦA GIẢI THUẬT DI TRUYỀN VÀ GIẢI THUẬT TỐI ƯU HÓA ĐÀN KIẾN CHO BÀI TOÁN NGƯỜI DU LỊCH

Lê Quốc Anh
Viện Kỹ thuật Công nghệ, Trường Đại học Vinh
Ngày nhận bài 17/6/2019, ngày nhận đăng 02/8/2019

Tóm tắt: Trong bài báo này, chúng tôi nghiên cứu áp dụng giải thuật di truyền và giải thuật tối ưu hóa đàn kiến, là các giải thuật thuộc lớp giải thuật tìm kiếm meta-heuristics, cho bài toán người du lịch. Chúng tôi thực hiện các thực nghiệm để đánh giá giải thuật nào giải bài toán hiệu quả hơn theo nghĩa đạt được chất lượng nghiệm và thời gian tìm kiếm nghiệm tốt nhất. Kết quả thử nghiệm chỉ ra rằng giải thuật tối ưu hóa đàn kiến là giải thuật hiệu quả trong việc tìm chu trình ngắn nhất, ngược lại giải thuật di truyền là giải thuật hiệu quả về thời gian khi số đỉnh của đồ thị lớn.

Từ khóa: Giải thuật di truyền; giải thuật tối ưu hóa đàn kiến; thuật toán tìm kiếm; bài toán người du lịch.

1. Giới thiệu

Bài toán người du lịch (Travelling Salesman Problem - TSP) là một bài toán tối ưu tổ hợp được nghiên cứu trong lĩnh vực tối ưu hóa và khoa học máy tính. Bài toán TSP được mô tả như sau: cho một tập các thành phố, chúng ta cần tìm một chu trình đi qua tất cả các thành phố, mỗi thành phố đúng một lần sao cho tổng khoảng cách đi qua các thành phố là nhỏ nhất. Bài toán TSP có thể được biểu diễn bởi một đồ thị $G = (V,E)$, trong đó V là tập các thành phố tương ứng các đỉnh của đồ thị và E là tập các đường đi giữa các thành phố tương ứng với cạnh của đồ thị. Mỗi cạnh $(i,j) \in E$ được gán một giá trị d_{ij} tương ứng là khoảng cách của thành phố i đến j. Như vậy, bài toán TSP tương đương với việc tìm chu trình Hamilton có độ dài ngắn nhất trên một đồ thị có trọng số. Bài toán TSP thuộc lớp bài toán NP - khó (NP - hard) vì có độ phức tạp tính toán là hàm giai thừa [1] và có thể được giải bằng cách sử dụng thuật toán vét cạn (exhausive algorithm) hoặc thuật toán tìm kiếm xấp xỉ (approximation algorithm).

Thuật toán vét cạn cho phép tìm được chu trình có chiều dài ngắn nhất cho bài toán TSP, đó là tìm tất cả các chu trình Hamilton trong đồ thị và sau đó lấy chu trình có chiều dài ngắn nhất. Với đồ thị có n đỉnh sẽ có tối đa $(n-1)!/2$ chu trình Haminton, tức là độ phức tạp của thuật toán là hàm giai thừa, do vậy khi số đỉnh của đồ thị tăng thì số phép tính trong thuật toán sẽ tăng cấp giai thừa. Ví dụ với đồ thị 25 đỉnh, thuật toán vét cạn cần thực hiện $25!/2 \approx 8 * 10^{24}$ phép tính. Rõ ràng rằng sử dụng thuật toán vét cạn để giải bài toán TSP là không khả thi khi số đỉnh đồ thị tăng lên nhanh.

Một hướng tiếp cận khác để giải bài toán TSP hiệu quả là sử dụng các giải thuật tìm kiếm xấp xỉ để tìm một chu trình đúng hoặc gần đúng trong một thời gian chấp nhận được. Các giải thuật xấp xỉ có thể được sử dụng để giải bài toán TSP như giải thuật láng giềng gần nhất (nearest neighbour algorithm) [1], giải thuật di truyền (genetic algorithm) [2] - [4], thuật toán tối ưu hóa đàn kiến (ant colony optimization) [5], [6].

Giải thuật di truyền (Genetic Algorithm - GA) được phát triển bởi Holland và cộng sự trong thập niên 1960 tại trường đại học Michigan với ý tưởng dựa trên quá trình

Email: anhquoc.hut@gmail.com

tiến hóa và chọn lọc tự nhiên của sinh vật [3]. Cho đến nay, GA đã được sử dụng để giải quyết hiệu quả các một lớp các bài toán tối ưu tổ hợp với các lời giải chấp nhận được [3].

Đặc tính quan trọng nhất của GA là không cần sử dụng các điều kiện truyền thống như điều kiện liên tục hay khả vi làm điều kiện tiên quyết. Để tìm nghiệm, GA thực hiện tìm kiếm song song đồng thời trong *quần thể (population),* trong đó khái niệm "nhiễm sắc thể" (chromosome) được sử dụng như là nghiệm của bài toán.

Giải thuật 1. Giải thuật di truyền

1. **Vào**: một quần thể và một hàm đánh giá độ thích nghi (fitness)
2. **Ra**: một nhiễm sắc thể, tức là nghiệm của bài toán
3. Khởi tạo quần thể, tỉ lệ đột biến ρ, xác suất chọn lọc ε.
4. Mã hóa nhiễm sắc thể
5. **Repeat**
6. Đánh giá độ thích nghi
7. Chọn lọc
8. Lai ghép
9. Đột biến
10. **Until** (thõa mãn điều kiện dừng).
11. Trả về nghiệm

Giải thuật tối ưu hóa đàn kiến (Ant Colony Optimization - ACO) là một phương pháp tìm kiếm nghiệm tối ưu xấp xỉ dựa trên ý tưởng mô phỏng cách tìm đường đi của các con kiến tự nhiên từ tổ tới nguồn thức ăn của chúng. Khi tìm đường đi, đàn kiến trao đổi thông tin gián tiếp và hoạt động theo phương thức tự tổ chức. Cụ thể, khi đi tìm mồi các con kiến sẽ rải vết mùi (pheromone) dùng để đánh dấu đường đi. Bằng cách cảm nhận vết mùi, các kiến có thể lần theo đường đi đến nguồn thức ăn được các con kiến khác khám phá theo phương thức chọn ngẫu nhiên có định hướng theo nồng độ vết mùi. Kiến chịu ảnh hưởng vết mùi của các con kiến khác (đường đi có nồng độ vết mùi càng cao thì xác suất được kiến chọn càng lớn) về quyết định chọn đường đi chính là ý tưởng thiết kế thuật toán ACO. Sử dụng mô hình đàn kiến, Dorigo đã xây dựng thuật toán *hệ kiến* (Ant System - AS) giải bài toán người du lịch [5]. Thuật toán này đã được phát triển và ứng dụng để giải quyết hiệu quả các bài toán tìm đường đi cho robot [7], bài toán tách cạnh của ảnh [8].

Trong bài báo này, chúng tôi nghiên cứu áp dụng giải thuật GA và giải thuật ACO bằng thực nghiệm để xác định giải thuật nào giải bài toán TSP hiệu quả hơn theo nghĩa đạt được chất lượng nghiệm và thời gian tìm kiếm nghiệm tốt hơn. Phần còn lại của bài báo được tổ chức như sau. Phần 2 mô tả về việc áp dụng giải thuật GA và ACO cho bài toán TSP. Phần 3 mô tả các thực nghiệm và đưa ra các đánh giá của các giải thuật GA và ACO cho bài toán TSP. Phần 4 đưa ra các kết luận của bài báo.

2. Áp dụng giải thuật GA và ACO cho bài toán TSP

2.1. Áp dụng giải thuật GA cho TSP

Giải thuật GA là một chuỗi các hành động bao gồm khởi tạo quần thể, đánh giá độ thích nghi, chọn lọc (selection), lai ghép (reproduction) và đột biến (mutation) các cá thể trong quần thể, như được mô tả ở Giải thuật 1. Khởi tạo quần thể là biểu diễn các cá

thể được chọn ngẫu nhiên theo một dạng mã hóa nào đó mà thường được gọi là nhiễm sắc thể. Mỗi nhiễm sắc thể được đánh giá độ thích nghi thông qua một hàm thích nghi (fitness). Chọn lọc là quá trình chọn các nhiễm sắc thể tốt theo nghĩa của hàm thích nghi để lai ghép sinh ra thế hệ tiếp theo. Lai ghép và đột biến nhằm sinh ra một thế hệ mới tốt hơn thế hệ trước đó. Rõ ràng, GA là giải thuật dựa trên ý tưởng của quá trình tiến hóa và chọn lọc tự nhiên nhằm sinh ra thế hệ tiếp theo tốt hơn thế hệ trước đó theo nghĩa của hàm thích nghi.

Ký hiệu $c_1, c_2, \ldots, c_n$ là tập gồm n thành phố, ký hiệu $d(c_i,c_j)$ là khoảng cách giữa 2 thành phố c_i và c_j. Trong nghiên cứu này, chúng tôi thử nghiệm với đồ thị vô hướng, do đó chúng tôi giả sử rằng $d(c_i,c_j) = d(c_j,c_i)$ và như vậy nghiệm của bài toán TSP là một hoán vị của n thành phố. Để áp dụng giải thuật GA cho bài toán TSP, chúng tôi định nghĩa các phép mã hóa, chọn lọc, lai ghép và đột biến như sau:

- **Mã hóa nhiễm sắc thể:** Phương pháp biểu diễn đường dẫn được sử dụng để biểu diễn các nghiệm (nhiễm sắc thể) của bài toán. Ví dụ với $n = 5$, các nghiệm có thể là các hoán vị {*1, 2, 3, 4, 5*}, {*1, 3, 4, 5, 2*}, {*1, 5, 4, 3, 2*}, {*5, 1, 4, 3, 2*}.

- **Hàm thích nghi**: Mục tiêu của bài toán là tìm chu trình ngắn nhất đi qua tất cả các thành phố với mỗi thành phố đúng một lần, do vậy hàm thích nghi của giải thuật được định nghĩa như công thức (1). Điều này có nghĩa rằng những cá thể tốt là những cá thể có hàm thích nghi là bé.

$$D = \sum_{k=1}^{n} d(c_k, c_{k+1}) \ \textit{với thành phố}\ c_{n+1} = c_1 \tag{1}$$

- **Chọn lọc các nhiễm sác thể***:* Để chọn lọc các nhiễm sắc thể cho thế hệ sau thì mỗi nhiễm sắc thể cần được đánh giá độ thích nghi. Sau đó, các nhiễm sắc thể được sắp xếp giảm dần theo hàm thích nghi. Giả sử N_{keep} là số cá thể được giữ lại và cũng chính là số cá thể được chọn để lai ghép, khi đó xác suất để chọn cá thể thứ i $(i = 1,2, \ldots, N_{keep})$ được định nghĩa như công thức (2).

$$p(c_i) = \frac{N_{keep} - i + 1}{\sum_{i=1}^{N_{keep}} i} \tag{2}$$

- **Lai ghép nhiễm sắc thể**: Nếu sử dụng toán tử lai ghép cho bài toán TSP như giải thuật di truyền nhị phân [2], [3] thì giải thuật sinh ra lỗi. Ví dụ nếu 2 cá thể x = {*3, 5, 1, 2, 4*}, y = {*1, 4, 5, 3, 2*} và điểm ghép k =2 thì sẽ tạo ra *2* con là {*3, 5, 5, 3, 2*} và {*1, 4, 1, 2, 4*}. Hiển nhiên 2 con tạo ra không phải là 2 chu trình. Do vậy, chúng tôi định nghĩa toán tử lai ghép như sau:

- Chọn một vị trí ngẫu nhiên trong *2* cá thể và *2* cá thể hoán đổi *2* số nguyên ở vị trí được chọn để tạo ra *2* cá thể mới.

- Tiếp tục hoán đổi *2* số nguyên trong *2* cá thể tạo ra nếu bị trùng giá trị cho đến khi không có giá trị trùng trong mỗi cá thể.

Ví dụ với *2* cá thể cha, x = {*4, 1, 5, 3, 2, 6*}, và mẹ, y = {*3, 4, 6, 2, 1, 5*}, với điểm ghép bắt đầu $k = 4$. Khi đó toán tử lai ghép được thực hiện như sau:

Hai cá thể cha và mẹ: *x, y*	Bước 1	Bước 2	Bước 3	Bước 4
*4 1 5|**3**|2 6* *3 4 6|**2**|1 5*	*4 1 5 2|**2**|6* *3 4 6 3|**1**|5*	*4|**1**|5 2 1 6* *3|**4**|6 3 2 5*	*|**4**|4 5 2 1 6* *|**3**|1 6 3 2 5*	*3 4 5 2 1 6* *4 1 6 3 2 5*

- **Đột biến nhiễm sắc thể**: Toán tử đột biến được thực hiện với một xác suất nhỏ nhằm tránh bẫy cục bộ, đó là hoán đổi *2* vị trí bất kỳ của một nhiễm sắc thể sau khi thực hiện toán tử lai ghép.

2.2. Áp dụng giải thuật ACO cho TSP

Giải thuật ACO là một lớp các giải thuật dựa trên ý tưởng mô phỏng cách tìm đường đi của các con kiến tự nhiên từ tổ tới nguồn thức ăn của chúng. Giải thuật ACO cho bài toán TSP được mô tả như Giải thuật 2 [5]. Trong bài báo này chúng tôi áp dụng giải thuật hệ kiến (Ant System - AS) và giải thuật hệ đàn kiến (Ant Colony System - ACS) [5], là lớp giải thuật ACO, để giải bài toán TSP. Sự khác biệt các thuật toán AS và ACS là cách thức cập nhật vết mùi trên đường đi của các kiến.

Giải thuật 2. Giải thuật ACO cho bài toán TSP

1. **Vào:** Một đồ thị có trọng số *G = (V,E)*
2. **Ra:** Một chu trình
3. Khởi tạo tham số , ma trận vết mùi τ, khởi tạo *m* con kiến
4. **Repeat**
5. **for** $k \leftarrow 1$ **to** *m* **do**
6. Kiến thứ *k* xây dựng lời giải
7. Cập nhật vết mùi theo luật cập nhật cục bộ
8. **end for**
9. Cập nhật vết mùi theo luật cập nhật tổng thể
10. Cập nhật chu trình tối ưu nhất
11. **Until** (thõa mãn điều kiện dừng)
12. Trả về chu trình tìm được

2.2.1. Giải thuật hệ kiến (Ant System - AS)

Giải thuật AS giải bài toán TSP được dựa trên Giải thuật 2, tuy nhiên không có luật cập nhật cục bộ (dòng 7). Ban đầu mỗi kiến được khởi tạo ngẫu nhiên ở một thành phố xuất phát. Trong quá trình tìm nghiệm, mỗi con kiến *k* ở thành phố *i* chọn thành phố lân cận *j* dựa trên xác suất chuyển trạng thái (random-proportional rule) được định nghĩa bởi (3):

$$p_k(i,j) = \begin{cases} \dfrac{[\tau(i,j)].[\eta(i,j)]^{\beta}}{\sum_{u \in J_k(i)}[\tau(i,u)].[\eta(i,u)]^{\beta}} & nếu\ s \in J_k(i) \\ 0\ nếu\ ngược\ lại \end{cases} \quad (3)$$

trong đó $\tau(i,j)$ là vết mùi của cạnh *(i,j)*, $\eta_{ij} = \frac{1}{d_{ij}}$ là giá trị hueristic của cạnh *(i,j)*, $J_k(i)$ là tập các thành phố lân cận mà kiến *k* chưa ghé thăm và β là một tham số xác định quan hệ giữa vết mùi và độ dài của các cạnh *($\beta > 0$).* Sau khi tất cả các con kiến hoàn thành chu trình, thuật toán sẽ tiến hành cập nhật tổng thể nhằm thay đổi vết mùi trên các cạnh của đồ thị theo luật (4):

$$\tau(i,j) = (1-\alpha).\tau(i,j) + \sum_{k=1}^{m} \Delta\tau_k(i,j) \quad (4)$$

trong đó $\Delta\tau_k(i,j) = \begin{cases} \frac{1}{L_k}, \text{nếu } (i,j) \text{ thuộc chu trình của kiến } k \\ 0, \text{nếu ngược lại} \end{cases}$, $0 < \alpha < 1$ là tham số bay hơi của vết mùi, L_k là chiều dài của chu trình tạo bởi kiến *k* và *m* là số kiến. Mục đích của luật cập nhật tổng thể là cập nhật càng nhiều giá trị vết mùi cho các chu trình ngắn.

2.2.2. Giải thuật hệ đàn kiến (Ant Colony System - ACS)

Giải thuật ACS giải bài toán TSP khác với giải thuật AS ở ba khía cạnh sau: *(i)* luật chuyển trạng thái của mỗi con kiến là cân bằng giữa quá trình thăm dò cạnh mới và khai thác vết mùi đã tích lũy trước đó; *(ii)* luật cập nhật vết mùi tổng thể chỉ được thực hiện cho cạnh thuộc đường đi tốt nhất; và *(iii)* luật cập nhật vết mùi cục bộ cho mỗi con kiến khi đi qua một cạnh nào đó.

- Luật chuyển trạng thái: Luật chuyển trạng thái của mỗi con kiến di chuyển từ thành phố *i* đến thành phố *j* dựa trên công thức (5):

$$j = \begin{cases} \underset{u \in J_k(i)}{\text{argmax}}\{[\tau(i,j)].[\eta(i,j]^{\beta}\} \text{ nếu } q < q_0, \\ J \text{ nếu ngược lại} \end{cases} \quad (5)$$

trong đó q là một số ngẫu nhiên được phân bố đều trên khoảng [0,1], q_0 là một tham số xác định trước ($0 \le q_0 \le 1$) và *J* là một giá trị được xác định dựa theo (3). Với cách áp dụng luật chuyển trạng thái này, thuật toán được chỉ ra là tìm được nghiệm tối ưu hơn AS [5].

- Luật cập nhật vết mùi tổng thể: Sau khi tất cả các con kiến hoàn thành chu trình, luật cập nhật vết mùi tổng thể thực hiện cập nhật vết mùi trên chu trình có chiều dài ngắn nhất theo luật (6):

$$\tau(i,j) = (1-\alpha).\tau(i,j) + \alpha\Delta\tau(i,j), \quad (6)$$

trong đó $\Delta\tau(i,j) = \begin{cases} \frac{1}{L_{gb}}, \text{nếu } (i,j) \text{ thuộc chu trình ngắn nhất} \\ 0, \text{nếu ngược lại} \end{cases}$ và $0 < \alpha < 1$ là một tham số bay vết mùi.

- Luật cập nhật vết mùi cục bộ: Khi mỗi con kiến đi qua một cạnh *(i,j)* nào đó, luật cập nhật vết mùi cục bộ được thực hiện như luật (7):

$$\tau(i,j) = (1-\rho).\tau(i,j) + \rho\Delta\tau(i,j), \quad (7)$$

trong đó $0 < \rho < 1$ là một tham số, $\Delta\tau(i,j)$ là tham số được xác định bởi thực nghiệm. Trong bài báo này chúng tôi chọn $\Delta\tau(i,j) = 0$.

3. Thực nghiệm

Trong phần này, chúng tôi thực hiện các thử nghiệm bằng phần mềm Matlab 7.0 trên một máy tính Core i7-8550U CPU 1.8 GHz với 16 GB RAM. Để đánh giá hiệu quả của giải thuật GA, AS và ACS cho bài toán TSP, chúng tôi tạo ra các đồ thị liên kết đầy

đủ (complete graph) mà tọa độ các đỉnh được tạo ngẫu nhiên trong đoạn [0,1]. Ma trận kề của đồ thị được xây dựng dựa trên khoảng cách Euclide của tọa độ các đỉnh.

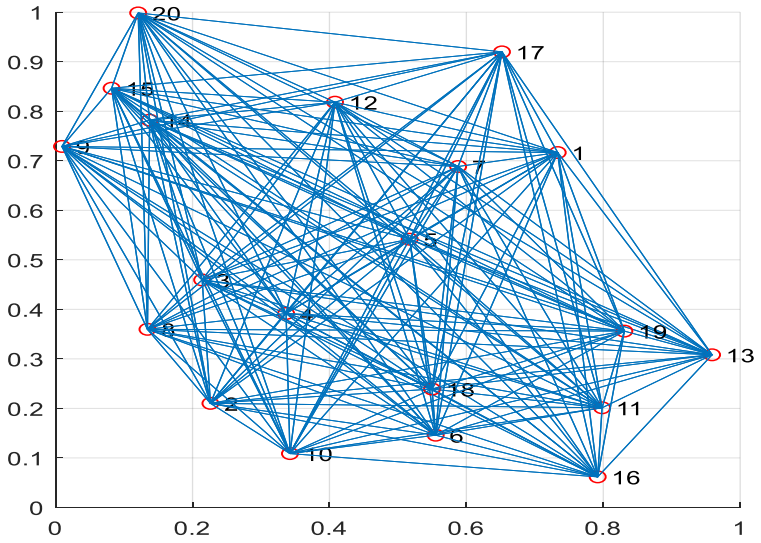
3.1. Chọn các tham số của thuật toán

Để lựa chọn các tham số tối ưu cho các giải thuật GA, AS và ACS, chúng tôi tạo ra 5 đồ thị ngẫu nhiên có số đỉnh là 10, 20, 30, 40, 50 và thực hiện các thử nghiệm để đánh giá sự ảnh hưởng của các tham số đến chất lượng nghiệm cũng như thời gian thực hiện giải thuật. Cụ thể, trong giải thuật GA, các giá trị tốt nhất của các tham số được chọn là ε = 50% số lượng quần thể, xác suất đột biến $\rho = 0.01$, số vòng lặp tối đa là 5000. Các giá trị tốt nhất của các tham số trong các giải thuật AS và ACS được chọn là $\alpha = 0.9$, $\beta = 9.0$, $\rho = 0.1$, $q_0 = 0.05$ và số lượng kiến là m = 50% số đỉnh của đồ thị.

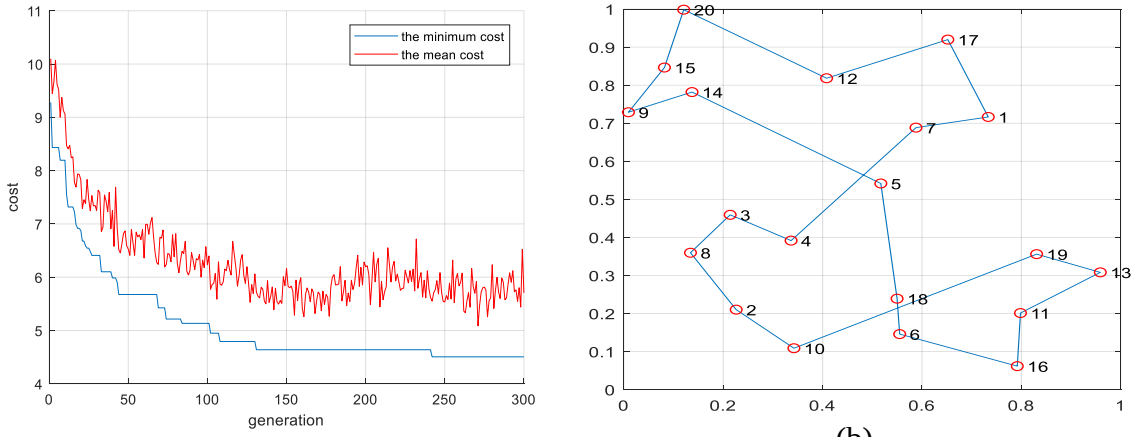
3.2. Kết quả thực nghiệm

Chúng tôi so sánh hiệu quả của các giải thuật GA, AS, ACS trên 2 yếu tố là thời gian thực hiện đến khi thuật toán bắt đầu hội tụ và chiều dài chu trình tốt nhất tìm được của các thuật toán đến khi thuật toán hội tụ. Để so sánh hiệu quả của các giải thuật, chúng tôi tạo ngẫu nhiên 5 đồ thị cho mỗi loại có các kích thước 10, 20, 30, 40 và 50.

Đầu tiên chúng tôi thực hiện các giải thuật GA, AS, ACS cho một đồ thị đầy đủ có 20 đỉnh như Hình 1 để quan sát sự thực hiện của các giải thuật. Hình 2(a) mô tả sự biến thiên của chiều dài chu trình tìm được theo sự tiến hóa của các thế hệ. Sau khoảng 240 thế hệ, giải thuật GA hội tụ và chu trình tìm được như Hình 2(b) với chiều dài của chu trình là 4.5058. Hình 3(a) mô tả chiều dài chu trình tìm được theo các vòng lặp của giải thuật AS. Dễ thấy rằng các chu trình tìm được qua các vòng lặp là không ổn định. Hình 3(b) mô tả chu trình tốt nhất tìm được sau 200 vòng lặp với chiều dài là 4.0216. Hình 4(a) mô tả chiều dài chu trình tìm được theo các vòng lặp của giải thuật ACS. Kết quả thử nghiệm chỉ ra rằng sau khoảng 80 vòng lặp, giải thuật sẽ hội tụ về một chu trình. Hình 4(b) mô tả chu trình sau khi giải thuật hội tụ với chiều dài là 3.9509. Với đồ thị này, dễ thấy rằng giải thuật ACS tìm được chu trình ngắn nhất.

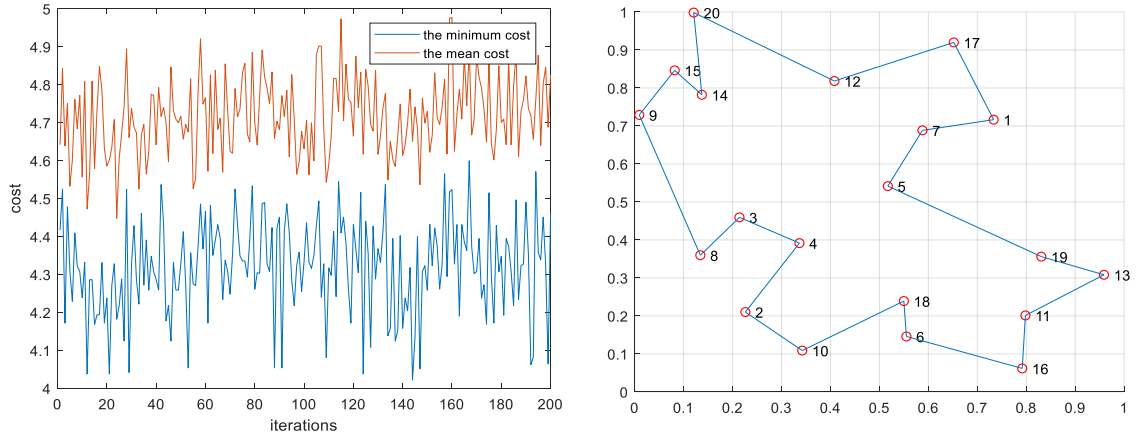


Hình 1: *Đồ thị đầy đủ 20 đỉnh được tạo ngẫu nhiên*



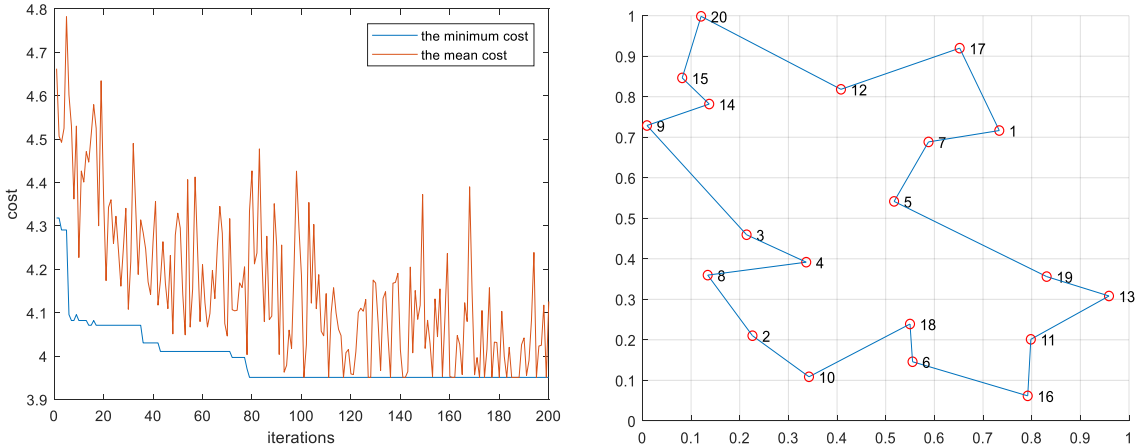
(a) (b)

Hình 2: *Kết quả thực hiện của thuật toán GA*



(a) (b)

Hình 3: *Kết quả thực hiện của thuật toán AS*



(a) (b)

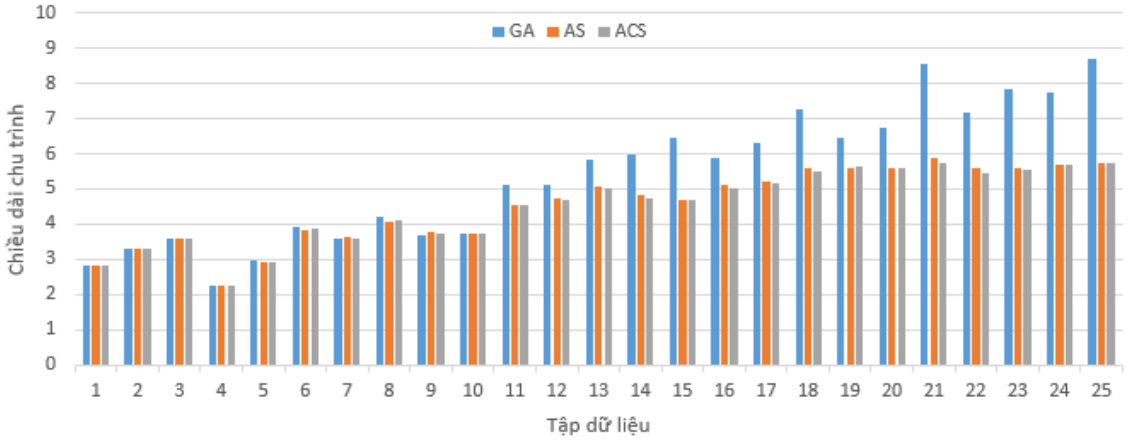
Hình 4: *Kết quả thực hiện của thuật toán ACS*

Kết quả thử nghiệm cho các đồ thị có các kích thước 10, 20, 30, 40 và 50 được chỉ ra ở các Hình 5 và Hình 6. Một số nhận xét chính từ quan sát các kết quả thực nghiệm như sau:

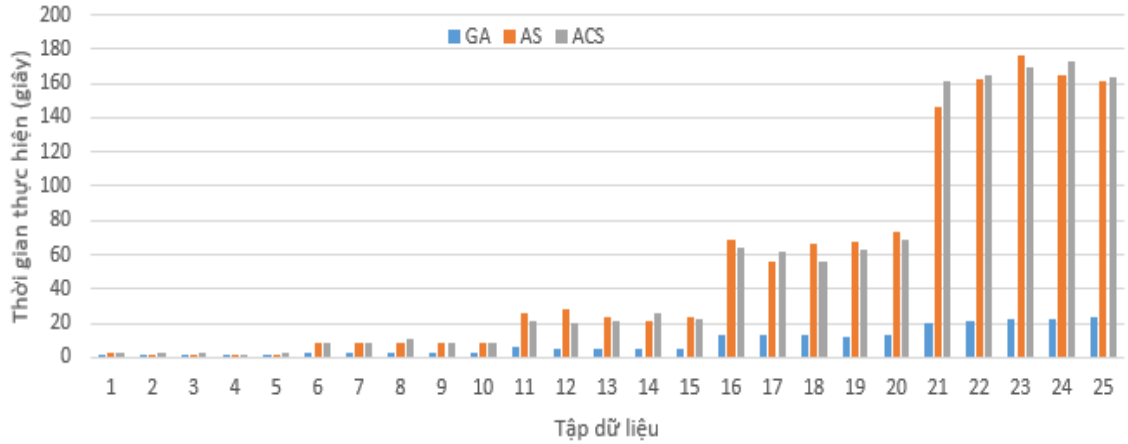
- Hình 5 chỉ ra kết quả so sánh chiều dài chu trình tìm được của các giải thuật. Khi đồ thị có số đỉnh là 10 hoặc 20 (đánh chỉ số từ 1 đến 10), chiều dài chu trình tìm được của giải thuật là xấp xỉ bằng nhau, nhưng khi đồ thị có số đỉnh là 30, chiều dài chu trình tìm của các thuật toán đã bắt đầu thay đổi, đó là thuật tóan ACS tìm được chu trình ngắn nhất, tiếp theo là thuật toán AS và GA. Tuy nhiên, khi số đỉnh của các đồ thị tăng dần, sự khác nhau của chiều dài chu trình tìm được của các thuật toán càng thể hiện rõ. Thuật toán ACS luôn tìm được chu trình ngắn nhất, tiếp theo là thuật toán AS và GA.

- Hình 6 chỉ ra kết quả so sánh thời gian thực hiện của các giải thuật đến khi các giải thuật hội tụ. Giải thuật GA hội tụ nhanh nhất, trong khi 2 giải thuật AS và ACS có thời gian để hội tụ là xấp xỉ nhau.

Tóm lại, giải thuật ACS là giải thuật hiệu quả trong việc tìm chu trình ngắn nhất, giải thuật AS có kết quả xấp xỉ với giải thuật ACS, nhưng giải thuật GA hiệu quả hơn về thời gian khi số đỉnh của đồ thị lớn.



Hình 5*: Chiều dài chu trình tìm được của các thuật toán GA, AS và ACS*



Hình 6: *Thời gian tìm kiếm chu trình của các thuật toán GA, AS và ACS*

4. Kết luận

Bài báo này trình bày các kết quả nghiên cứu thực nghiệm trong việc áp dụng các giải thuật GA, AS và ACS cho bài toán người du lịch. Mục đích của nghiên cứu nhằm đánh giá giải thuật nào giải bài toán hiệu quả hơn theo nghĩa đạt được chất lượng nghiệm và thời gian tìm kiếm nghiệm tốt nhất. Kết quả thử nghiệm chỉ ra rằng giải thuật ACS là giải thuật hiệu quả trong việc tìm chu trình ngắn nhất, giải thuật AS có kết quả xấp xỉ với giải thuật ACS, nhưng giải thuật GA hiệu quả hơn về thời gian khi số đỉnh của đồ thị lớn.

TÀI LIỆU THAM KHẢO

[1] Khushboo Arora, Samiksha Agarwal and Rohit Tanwar, "Solving TSP Using Genetic Algorithm and Nearest Neighbour Algorithm and Their Comparison", *International Journal of Scientific & Engineering Research*, Vol. 7, Issue 1, pp.1014-1018, 2016.

[2] Jenna Carr, *An Introduction to Genetic Algorithms*, Jenna Carr Published, 2014.

[3] Randy L. Haupt, Sue Ellen Haupt, *Practical Genetic Algorithms*, A John Wiley & Sons, Inc., Publication, 2004.

[4] Jean-Yves Potvin, *Genetic Algorithms for the Traveling Salesman Problem*, Annals of Operations Research, Vol. 63, pp. 339-370, 1996.

[5] Dorigo M. and Gambardella M. L., "Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Proble", *IEEE Transactions on Evolutionary Computation*, Vol. 1, No. 1, pp. 53 - 66, 1997.

[6] Zar Chi Su Su Hlaing and May Aye Khine, "An Ant Colony Optimization Algorithm for Solving Traveling Salesman Problem", *International Conference on Information Communication and Management IPCSIT*, Vol. 16, 2011.

[7] Michael Brand, Michael Masuda, Nicole Wehner and Xiao-Hua Yu, "Ant Colony Optimization Algorithm for Robot Path Planning", *International Conference on Computer Design and Applications (ICCDA 2010)*, Vol 5, pp. 436-440, 2010.

[8] Jing Tian, Weiyu Yu and Shengli Xie, An Ant Colony Optimization Algorithm for Image Edge Detection, *IEEE Congress on Evolutionary Computation*, pp: 751-756, 2008.

SUMMARY

COMPARING THE EFFECTIVENESS OF THE GENETIC ALGORITHM AND ANT COLONY OPTIMIZATION ALGORITHMS FOR THE TRAVELING SALESMAN PROBLEM

In this paper, we apply the genetic algorithm and ant colony optimization algorithms, which is a kind of meta-heuristics search algorithm, for the traveling salesman problem. We perform experiments to evaluate which one among these algorithms solves the problem more efficiently by means of the solution quality and the execution time. The experimental results show that the ant colony optimization algorithms are efficient in terms of the solution quality, while the genetic algorithm is efficient in terms of the execution time for large traveling salesman problems

Keyword: Genetic algorithm; ant colony optimization; metaheuristics; Travelling Salesman Problem - TSP.