NGHIÊN CỨU KHUNG THUẬT TOÁN CHUNG PSO ĐỂ GIẢI BÀI TOÁN TSP

Nguyễn Hoàng Hà

Khoa Công nghệ thông tin, Trường Đại học khoa học, Đại học Huế

Email: nhha@husc.edu.vn

Ngày nhận bài: 27/4/2021; ngày hoàn thành phản biện: 12/5/2021; ngày duyệt đăng: 02/11/2021

TÓM TẮT

Bài toán người du lịch (TSP) là một bài toán tối ưu tổ hợp kinh điển. Nó thuộc lớp các bài toán NP-khó và không thể giải được trong thời gian đa thức. Trên thực tế người ta thường giải quyết các bài toán này bằng các phương pháp heuristic, chúng cho ra nghiệm gần tối ưu. Các phương pháp heuristic bao gồm phương pháp nhánh cận, heuristic ACO (Ant Colony Optimization), thuật toán GA (Genetic Algorithm), … nhưng các phương pháp này chỉ áp dụng cho lớp các bài toán nhỏ, khi kích cỡ bài toán lớn thì thời gian chạy của bài toán là rất lớn. Trong bài báo này, chúng tôi nghiên cứu khung thuật toán chung PSO (Particle Swarm Optimization) , từ đó xây dựng mô hình toán học và áp dụng để giải bài toán TSP với số đỉnh của bài toán lớn và tối ưu thời gian thực hiện

Từ khóa: TSP, PSO, bài toán người du lịch, Metaheuristic PSO.

1. MỞ ĐẦU

Bài toán tối ưu tổ hợp là bài toán tìm ra tổ hợp tốt nhất trong những tổ hợp có thể tạo ra, thỏa mãn yêu cầu cho trước. Với các bài toán tối ưu tổ hợp NP-khó có cỡ nhỏ, người ta có thể tìm lời giải tối ưu nhờ phương pháp tìm kiếm vét cạn. Tuy nhiên, với các bài toán cỡ lớn thì đến nay chưa thể có thuật toán tìm lời giải đúng với thời gian đa thức nên chỉ có thể tìm lời giải gần đúng hay đủ tốt [1].

Theo cách tiếp cận truyền thống hay là tiếp cận cứng, các thuật toán gần đúng phải được chứng minh tính hội tụ hoặc ước lượng được tỷ lệ tối ưu. Với việc đòi hỏi khắt khe về toán học như vậy làm hạn chế số lượng các thuật toán công bố, không đáp ứng được nhu cầu ngày càng phong phú và đa dạng trong nghiên cứu và ứng dụng. Để khắc phục tình trạng này, người ta dùng tiếp cận đủ tốt để xây dựng các thuật toán tối ưu mềm[1].

Bài toán người du lịch (TSP: Traveling Salesman Problem) có thể biểu diễn thành một đồ thị, trong đó các thành phố là các đỉnh, các con đường đi lại giữa các thành phố là các cạnh, đồng thời khoảng cách giữa các thành phố là trọng số tương ứng của cạnh nối chúng, nếu giữa hai thành phố không có đường đi trực tiếp mà phải thông qua một thành phố khác thì ta gán trọng số của cạnh đó là số lớn nhất có thể. Khi đó, bài toán người du lịch trở thành bài toán tìm một chu trình Hamilton ngắn nhất, và lộ trình của người du lịch chính là chu trình Hamilton ngắn nhất.

Nếu dùng phương pháp vét cạn để giải bái toán TSP thì ta luôn cho ta một đáp án tối ưu nhất, tuy nhiên độ phức tạp của nó là quá cao (O(n!)). Vì vậy, để giải bài toán này người ta thường dùng các thuật toán tối ưu mềm.

Năm 2017, tác giả Đỗ Như An [2] đã nghiên cứu thuật toán nhánh-cận để giải bài toán TSP nhưng độ phức tạp thời gian tính toán dạng hàm mũ. Vì vậy, nếu đưa thuật toán vào sử dụng cũng chưa đáp ứng được yêu cầu thực tế.

Thuật toán di truyền [3] là thuật toán metaheuristic, metaheuristic là một cách gọi chung cho các thuật toán heuristic trong việc giải quyết các bài toán tối ưu tổ hợp. Abid Hussain [4] và các đồng nghiệp đã cải tiến thuật toán GA để giải bài toán TSP nhưng bài báo chỉ tập trung vào đánh giá các phương pháp cải tiến chưa đưa ra được số đỉnh tối đa và thời gian chạy của thuật toán. Omar [5] và các đồng nghệp đã cải tiến thuật toán GA để giải bài toán TSP nhưng chỉ mô phỏng được tối đa 20 đỉnh.

PSO (Particle Swarm Optimization) là khung thuật toán chung dựa trên kinh nghiệm của bầy đàn được đề xuất bởi bởi Kennedy và Eberhat [5]. Nó là một khung thuật toán thông minh dựa trên bầy đàn, mô phỏng lại hành vi xã hội của bầy chim hay đàn cá khi đi tìm nguồn thức ăn.

Một cá thể (particle) được thể hiện trong PSO tương tự như một con chim hoặc một con cá tìm kiếm thức ăn trong không gian tìm kiếm của nó. Sự di chuyển của mỗi cá thể là sự kết hợp giữa vận tốc và hướng di chuyển. Vị trí của mỗi cá thể tại bất kỳ thời điểm nào cũng bị ảnh hưởng bởi vị trí tốt nhất của nó và vị trí tốt nhất của cả bầy đàn. Hiệu quả đạt được của một cá thể được xác định bởi một giá trị thích nghi, giá trị này được xác định phụ thuộc vào từng bài toán [5].

Trong PSO, quần thể bao gồm các cá thể trong không gian của bài toán. Các cá thể được khởi tạo một cách ngẫu nhiên. Mỗi cá thể sẽ có một giá trị thích nghi, giá trị này được xác định bởi một hàm thích nghi để tối ưu trong mỗi thế hệ. Trong mỗi thế hệ, mỗi cá thể thay đổi vận tốc và thay đổi vị trí của nó theo thời gian. Dựa vào giá trị thích nghi, mỗi cá thể tìm ra giải pháp tối ưu cục bộ trong không gian tìm kiếm nhiều chiều. Sau đó, giải pháp tối ưu cục bộ được so sánh với giải pháp tối ưu toàn cục của cả bầy đàn để cập nhật lại giá trị cho giải pháp tối ưu toàn cục. Dựa vào giải pháp tối ưu toàn cục để tìm ra giải pháp tối ưu nhất.

Bài báo này nghiên cứu khung thuật toán chung PSO để giải bài toán TSP, từ đó xây dựng mô hình toán học, đưa ra giải thuật và cài đặt thử nghiệm. Kết quả thực nghiệm được phân tích và đánh giá với thuật toán di truyền của Luca Benci (2015).

2. MÔ HÌNH VÀ THUẬT TOÁN

2.1 Mô hình hệ thống

Để áp dụng khung thuật toán PSO vào bài toán cụ thể chúng ta phải xác định được vị trí, vận tốc, hàm thích nghi, vị trí tối ưu cục bộ của từng cá thể và vị trí tối ưu toàn cục của cả bầy đàn [5]. Phần này bài báo tập trung xây dựng mô hình toán học để giải bài toán TSP.

Chúng ta xét quần thể gồm Y cá thể, mỗi cá thể x lặp N lần để tìm kiếm thức ăn trong không gian M_x chiều. Như vậy, tại vòng lặp thứ i, (i=1…N) mỗi cá thể x, (x=1..Y) sẽ có M_x vị trí, M_x vận tốc và được biểu diễn:

$$P_{ix} = \{p_{ix}^1, \dots, p_{ix}^{M_x}\} \quad (1.1)$$

$$V_{ix} = \{v_{ix}^1, \dots, v_{ix}^{M_x}\} \quad (1.2)$$

Trong đó:

p_{ix}^j là vị trí của cá thể x ở vòng lặp i tại chiều j, (j=1...M_x).

v_{ix}^j là vận tốc của các thể x ở vòng lặp i tại chiều j, (j=1...M_x).

Trong bài toán người du lịch ta có Y thành phố, mỗi thành phố được nối với nhiều thành phố lân cận. Mỗi cá thể x (x= 1..Y) tìm kiếm thành phố trên không gian M_x chiều để tìm ra thành phố hợp lý tại lần lặp thứ i, mỗi cá thể x sẽ tìm ra các thành phố và M_x chính là số thành phố lân cận với x. Giá trị của p_{ix}^j chính là thành phố j lân cận với x. Giá trị này được lấy ngẫu nhiên từ 1 … M_x, và giá v_{ix}^j được lấy ngẫu nhiên từ -M_x … M_x, với M_x là số thành phố lân cận với thành phố x.

Để đạt được mục tiêu là tổng chi phí đi qua các thành phố là nhỏ nhất, chúng ta xây dựng hàm thích nghi để cá thể x chọn ra thành phố lân cận tại lần lặp thứ i như sau:

$$F(p_{ix}^j) = \frac{1}{C_{ijx}} \quad (1.3)$$

Trong đó: C_{ijx} là trọng số của thành phố x với thành phố j tại lần lặp thứ i. Khi trọng số C_{ijx} càng cao thì giá trị hàm thích nghi càng thấp. Ngược lại, khi trọng số C_{ijx} càng thấp thì giá trị hàm thích nghi càng cao nên xác suất để thành phố x chọn thành phố j càng lớn.

Từ hàm thích nghi ở công thức (1.3), ta tiến hành xây dựng công thức để tìm vị trí tối ưu cục bộ của cá thể x tại chiều j+1 như sau:

$$pb_{ix}^{j+1} = \begin{cases} pb_{ix}^{j+1} \; nếu \; F(p_{ix}^{j+1}) \geq F(p_{ix}^{j}) \\ pb_{ix}^{j} \; ngược \; lại \end{cases} \quad (1.4)$$

Vị trí tối ưu cục bộ tại vị trí j+1 được cập nhật nếu giá trị của hàm thích nghi tại vị trí j+1 lớn hơn hoặc bằng giá trị của hàm thích nghi tại vị trí trước nó, ngược lại nó vẫn giữa lại giá trị của hàm thích nghi trước đó.

Sau khi xác định được vị trí tối ưu cục bộ của từng các thể, chúng tôi tiến hành xác định vị trí tối ưu cục bộ lớn nhất của các cá thể (Pbest) và vị trí tối ưu toàn cục của cả đàn (Gbest) như sau:

Sau khi mỗi cá thể x (x=1…Y) tìm kiếm được thành phố lân cận trên không gian M_x chiều, mỗi cá thể tìm được vị trí tối ưu nhất ($Pbest_x$) bằng cách phân tích các vị trí tối ưu trên mỗi chiều và được xác định:

$$Pbest_x = \max_{x=1..Y,\, j=1..M_x} \{pb_{ix}^{j}\} \quad (1.5)$$

Vị trí tối ưu toàn cục của cả đàn (Gbest) phụ thuộc vào vị trí tối ưu cục bộ lớn nhất của từng cá thể và được xác định như sau:

$$Gbest = \max_{x=1..Y} \{Pbest_x\} \quad (1.6)$$

Mỗi cá thể dựa vào vận tốc hiện tại và khoảng cách từ $Pbest_x$ đến $Gbest$ để thay đổi vị trí và điều chỉnh tốc độ của nó như sau [5] :

$$v_x^{j+1} = v_x^j + c_1 z_1 \left(Pbest_x - pos_x^j\right) + c_2 z_2 \left(Gbest - pos_x^j\right) \quad (1.7)$$

$$pos_x^{j+1} = pos_x^j + v_x^j \quad (1.8)$$

Trong đó:

- pos_x^j: vị trí hiện tại của cá thể thứ x tại chiều j.
- c_1, c_2: hệ số gia tốc.
- z_1, z_2: là số ngẫu nhiên giữa 0 và 1.
- v_x^j: vận tốc của cá thể x tại chiều j.

2.2 Thuật toán TSPPSO

Áp dụng mô mình toán học ở Phần 2, chúng ta xây dựng thuật toán TSPPSO áp dụng cho bài toán người du lịch như sau :

Đầu vào: Tập cá thể Y, c_1,c_2, số đỉnh và tập chứa trọng số giữa các đỉnh.

Đầu ra: Một đường đi gần tối ưu đi qua các đỉnh.

Phương pháp:

1 Khởi tạo vị trí, vận tốc của các thể ;

2 **While** chưa tìm được đường đi qua các đỉnh **DO**

Begin

3 Tính giá trị thích nghi như công thức *(1.4);*

4 Tính Pbest như công thức *(1.5);*

5 Tính Gbest như công thức (1.6);

6 Tính vận tốc và cập nhật lại vị trí như công thức (1.7) và (1.8);

7 **End**

8 Hiển thị ra đường đi gần tối ưu qua các đỉnh

3. MÔ PHỎNG VÀ ĐÁNH GIÁ THUẬT TOÁN

Thuật toán TSPPSO được cài đặt trên ngôn ngữ C#, sử dụng tập dữ liệu chuẩn Pr76Dataset.xml và thư viện TspPsoDemo. Các tham số của thuật toán TSPPSO được xác định như sau:

Trường hợp 1. Mô phỏng thuật toán PSO từ tập dữ liệu Pr76Dataset.xml

Dữ liệu đầu vào:

- Số lần lặp: 20000
- Số cá thể: 50.. 200
- Hệ số gia tốc cục bộ (c_1): 1.4
- Hệ số gia tốc toàn cục (c_2): 1.4
- Số đỉnh: 76 đỉnh và trọng số được lấy từ tập dữ liệu chuẩn Pr76Dataset.xml

Các kết quả của Bảng 1 và Bảng 2 được lấy trung bình cộng của 5 lần chạy thử nghiệm.

Bảng 1. Kết quả thực nghiệm về thời gian và độ lệch đường đi khi thay đổi số cá thể

Số cá thể	**Thời gian thực hiện (giây)**	**Độ lệch đường đi so với đường đi tốt nhất (%)**
50	25.5	9.971
100	50.3	3.714
150	76.7	2.480
200	102	2.172

Kết quả thực hiện của thuật toán TSPPSO:

Thuật toán khởi tạo các giá trị: hệ số gia tốc cục bộ C_1=1.4, hệ số gia tốc toàn cục C_2=1.4, số cá thể thay đổi từ 50 đến 200, số đỉnh 76 và cho kết quả ở Bảng 1.

Dựa vào kết quả thực nghiệm ở Bảng 1, ta thấy khi số cá thể càng lớn thì thời gian thực hiện càng cao nhưng ngược lại độ lệch đường đi so với đường đi tối ưu càng thấp. Thời gian tính toán của thuật toán TSPPSO tăng dần tỉ lệ thuận với số lượng cá thể (khi số lượng cá thể ít thì thời gian thực hiện nhanh, với 50 cá thể thời gian thực hiện là 25.5 giây, nhưng khi số cá thể là 200 thì thời gian thực hiện lên đến 102 giây).

Khi số cá thể lớn thì số lần lặp sẽ tăng lên, nên thời gian thực hiện của thuật toán sẽ lớn. Khi số cá thể càng lớn thì sẽ tìm ra nhiều vị trí tối ưu cục bộ ($Pbest_x$), vì vậy, xác suất tìm được vị trí tối ưu toàn cục của cả đàn (Gbest) sẽ cao.

Thuật toán TSPPSO giải bài toán TSP cho kết quả chính xác cao khi thực hiện trên bộ dữ liệu có số lượng đỉnh và số cá thể lớn. Khi số đỉnh và số cá thể càng lớn, thì vị trí tối ưu cục bộ sẽ tăng lên, xác xuất để chọn đường đi ngắn nhất cho vị trí toàn cục sẽ cao. Vì vậy, khi số đỉnh và số cá thể càng lớn thì độ lệch độ dài đường đi tìm được với độ dài đường đi tốt nhất sẽ thấp.

Trường hợp 2. So sánh thuật toán TSPPSO với thuật toán di truyền

Năm 2015, Luca Benci (https://github.com/7lb/TSP) đã sử dụng thuật toán di truyền để giải bài toán người du lịch. Bài báo này sử dụng thuật toán di truyền của Luca Benci để cài đặt và chạy thử nghiệm. Sau đó so sánh thời gian thực hiện của 2 thuật toán. Kết quả thử nghiệm được thể hiện ở Bảng 2 và Hình 1.

Dữ liệu đầu vào của thuật toán TSPPSO:

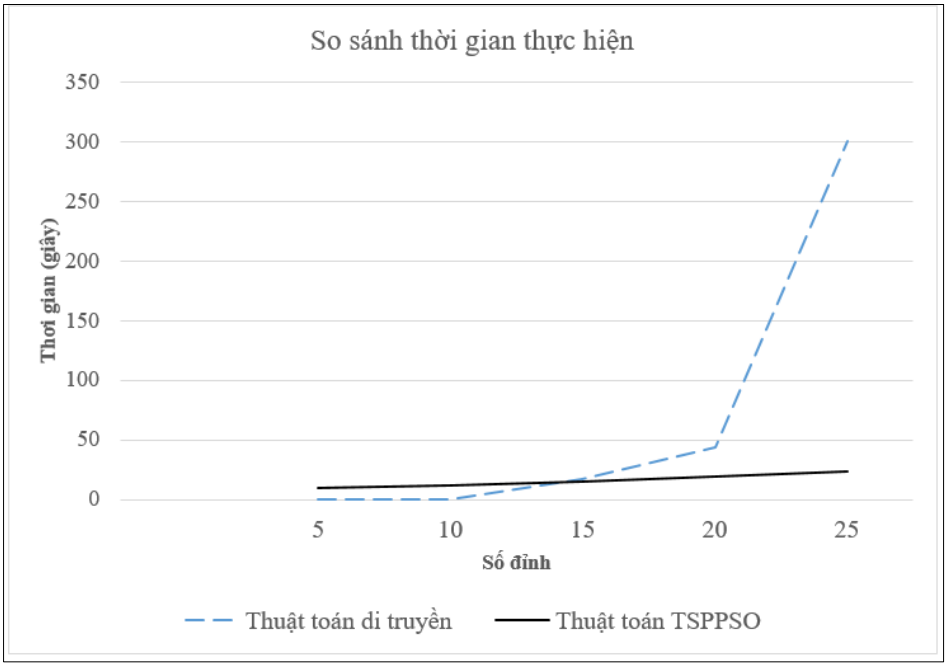
- Số lần lặp: 20000
- Số cá thể: 200
- Hệ số gia tốc cục bộ (c_1): 1.4
- Hệ số gia tốc toàn cục (c_2): 1.4
- Số đỉnh thay đổi từ 5 đến 30 đỉnh

Dữ liệu đầu vào của thuật toán di truyền:

- mutRate = 0.03;
- elitism = 6;
- popSize = 200;
- numCities = 5 ..30;

Bảng 2. Bảng so sánh thời gian thực hiện của 2 thuật toán khi thay đổi số đỉnh từ 5 đến 25.

Số đỉnh	5	10	15	20	25
Thời gian thực hiện của thuật toán di truyền	0.009	0.12	17.5	43.4	300
Thời gian thực hiện của Thuật toán TSPPSO	9.15	12	14.8	18.8	23.56



Hình 1. Biểu đồ so sánh thời gian thực hiện của 2 thuật toán khi số đỉnh thay đổi từ 5 đến 25

Phân tích và đánh giá 2 thuật toán:

Từ kết quả của Bảng 2 và Hình 1 cho thấy:

Khi số đỉnh nhỏ (từ 5 đến 15) thì thuật toán di truyền sẽ tối ưu về thời gian hơn vì khi số đỉnh càng nhỏ thì tốc độ hội tụ của thuật toán nhanh. Trong khi đó thuật toán TSPPSO cũng phải chạy đủ số lần lặp và duyệt qua tất cả các cá thể để tìm giải pháp tối ưu cục bộ, vì vậy thời gian thực hiện của thuật toán sẽ cao hơn thuật toán di truyền.

Khi số đỉnh lớn (từ 20 đến 25) thì tốc độ hội tụ của thuật di truyền rất chậm, vì vậy khi số đỉnh lớn thì thời gian thực hiện của thuật toán rất lớn. Thuật toán TSPPSO vẫn duy trì số cá thể và số lần lặp nên thời gian không tăng đột biến như thuật toán di truyền.

Khi số đỉnh lớn (từ 30 đến 35) thì tốc độ hội tụ của thuật toán di truyền rất chậm, có nhiều trường hợp thuật toán không đi đến điểm hội tụ được làm cho thuật toán bị lặp vô hạn. Trong khi đó, thuật toán TSPPSO được thử nghiệm trên 76 đỉnh, thời gian thực hiện 102 giây và độ lệch đường đi so với đường đi tốt nhất gần 2.2% (Bảng 1) nên kết quả này có thể chấp nhận được

4. KẾT LUẬN

Bài báo này tập trung vào nghiên cứu khung thuật toán chung PSO, sau đó phân tích và xây dựng mô hình toán học áp dụng PSO vào để giải bài toán TSP. Từ mô hình toán học đã xây dựng bài báo tiến hành cài đặt thực nghiệm và so sánh với thuật toán di truyền. Thông qua việc phân tích, đánh giá các kết quả thực nghiệm, đối sách trên cùng một bộ dữ liệu cho thấy khi số đỉnh lớn thì kết quả của thuật toán TSPPSO có sự cải tiến đáng kể về thời gian so với thuật di truyền đang sử dụng.

TÀI LIỆU THAM KHẢO

[1]. Trần Ngọc Hà (2017), Các bài toán tối ưu tổ hợp và tính toán mềm, Luận án tiến sĩ, Dại học Quốc gia Hà Nội

[2]. Đỗ Như An (2017), Ứng dụng thuật toán nhánh cận để giải một số bài toán tối ưu liên quan đến chu trình hamilton dựa trên bài toán TSP, tạp chí Đại học Đà Lạt, tập 7, số 2, trang 205-2016

[3]. Chunhua Fu, Lijun Zhang, Xiaojing Wang, and Liying Qiao (2018), Solving TSP problem with improved genetic algorithm , AIP

[4]. Abid Hussain,Yousaf Shad Muhammad,M. Nauman Sajid, Ijaz Hussain, Alaa Mohamd Shoukry and Showkat Gani, Genetic Algorithm for Traveling Salesman Problem with Modified Cycle Crossover Operator, Computational Intelligence and Neuroscience Volume 2017, Article ID 7430125, 7 page

[5]. Omar M.Sallabi, Younis EI-Haddad (2009), An Improved Genetic Algorithm to Solve the Traveling Salesman Problem. Proceedings of world academy of science, engineering and technology, volume 40, APRIL 2009, ISSN: 2070-3740

[6]. J.Kennedy and R.Eberhart (1995). Particle swarm optimization. In Proceedings of IEEE International Conference on Neural Networks, pages 1942 -1948. IEEE.

A STUDY ON PSO GENERAL ALGORITHM FRAMEWORK FOR SOLVING TSP PROBLEM

Nguyen Hoang Ha

Faculty of Information Technology, University of Sciences, Hue University

Email: nhha@husc.edu.vn

ABSTRACT

The Traveling Salesman Problem (TSP) is a classical combinatorial optimization problem. It belongs to a class of NP-hard problems that cannot be solved in polynomial time. In fact, these problems are often solved by heuristic methods that obtained outcomes are near-optimal solutions. Nowadays, there are many heuristic methods used to deal with this problem such as the branch and bound method, heuristic ACO (Ant Colony Optimization), Genetic Algorithm (GA). However, in case of the problem with large-scale data, these methods are inefficient. In this paper, based on studying of the PSO general algorithm framework, we constructed a mathematical model and applied to solve TSP problems having the large number of vertices with optimal execution time.

Keywords: TSP, PSO, Traveling Salesman Problem, Metaheuristic PSO.

Nguyễn Hoàng Hà sinh ngày 22/11/1976 tại Quảng Nam. Năm 1999, ông tốt nghiệp đại học ngành Công nghệ Thông tin tại trường Đại học Khoa học, ĐH Huế. Năm 2005, ông nhận bằng thạc sỹ Khoa học Máy tính tại Trường Đại học Khoa học, ĐH Huế. Năm 2017, ông tốt nghiệp tiến sĩ chuyên ngành Khoa học máy tính tại trường Đại học Khoa học, ĐH Huế. Hiện ông công tác tại Trường Đại học Khoa học, Đại học Huế.

Lĩnh vực nghiên cứu: Xử lý song song và phân tán, tính toán lưới và tính toán đám mây.