# PHƯƠNG PHÁP MỚI GIẢI BÀI TOÁN NGƯỜI BÁN HÀNG SỬ DỤNG THUẬT TOÁN RUNNER – ROOT

*A NEW METHOD FOR SOLVING TRAVELING SALESMAN PROBLEM USING RUNNER-ROOT ALGORITHM*

***NGUYỄN THỊ QUYÊN(*)***

***TÓM TẮT:*** *Bài toán người bán hàng (Travelling Salesman Problem - TSP) là bài toán tìm đường đi ngắn nhất giữa nhiều thành phố cho người bán hàng nhằm tiết kiệm thời gian và chi phí. Đây là bài toán tối ưu rời rạc phức tạp, đòi hỏi phải có các phương pháp giải hiệu quả. Bài viết trình bày một phương pháp mới dựa trên thuật toán Runner - Root (RRA) để tìm đường đi ngắn nhất cho TSP. Trong đó, RRA là thuật toán được phát triển dựa trên ý tưởng về sự nhân giống của các loại thực vật bò lan. Hiệu quả của RRA cho bài toán TSP được kiểm chứng trên TSP 14 thành phố. Dựa trên kết quả tính toán cho thấy, phương pháp đề xuất RRA là một trong những công cụ đáng được xem xét cho bài toán TSP.*

***Từ khóa:*** *thuật toán runner (RRA) – root; bài toán người bán hàng (TSP); đường đi ngắn nhất.*

***ABSTRACT:*** *The traveling salesman problem (TSP) is the problem of finding the shortest route between many cities for sellers to save time and costs. This is a complex discrete optimization problem that requires effective solutions. The article presents a new method based on the runner-root algorithm (RRA) to find the shortest route to the TSP. In which, RRA is an algorithm developed based on the idea of propagation of creeping plants. The effectiveness of RRA for the TSP is verified on the TSP of 14 cities. The calculation results show that the proposed RRA method is one of the tools worth considering for the TSP.*

***Key words:*** *runner-root algorithm; traveling salesman problem; shortest route.*

## 1. ĐẶT VẤN ĐỀ

Bài toán TSP là bài toán tối ưu nổi tiếng nhằm tìm đường đi ngắn nhất giữa các thành phố cho người bán hàng, bao gồm thành phố bắt đầu và thành phố kết thúc cùng các thành phố đi qua nhưng tất cả chúng chỉ xuất hiện một lần trong đường đi. Bài toán TSP đã được ứng dụng trong cắt giấy, đi dây máy tính, định tuyến, lập lịch, mạng xã hội [2], [6]. Kể từ khi được đề xuất lần đầu vào năm 1970 [10], bài toán TSP đã được giải bằng nhiều phương pháp khác nhau bao gồm các phương pháp cổ điển như phương pháp nhánh và cắt [15], nhánh và biên [14], phương pháp Lagrangian [7] và các phương pháp dựa trên các thuật toán tối ưu như giải thuật di truyền [3], tối ưu bầy đàn (Particle Swarm Optimization - PSO) [16], tối ưu đàn kiến [4], thuật toán tìm kiếm cuckoo [11], thuật toán tìm kiếm hài hòa [1] và thuật toán đàn ong nhân tạo [12]. Nhìn chung, sử dụng các phương pháp cổ điển có thể thu được giải pháp tối ưu nhưng quá trình tính toán thường mất thời gian, Vì vậy, đòi hỏi phải có phương pháp giải hiệu quả rút ngắn thời gian hơn. Các thuật toán tối ưu có thể thu được lời giải bài toán với thời gian tương đối ngắn. Do đó, sử dụng phương

(*) ThS. Trường Đại học Văn Lang, *quyen.nt@vlu.edu.vn, Mã số: TCKH25-04-2021*

pháp mới cho bài toán TSP luôn là một vấn đề cần thiết.

Thuật toán runner – root (RRA) được phát triển dựa trên ý tưởng nhân giống qua thân của một số loài cây như cỏ nhện, dâu tây hay cây mẫu tử [8]. Để giải bài toán tối ưu, mỗi giải pháp được xem như một cây mẹ. Cây mẹ sinh ra các cây con qua thân của nó để khai thác nguồn tài nguyên. Các cây con sinh ra ở nơi có nguồn tài nguyên phong phú sẽ phát triển mạnh và tiếp tục sinh ra các cây con khác ngược lại chúng sẽ chết đi. Thuật toán RRA đã thể hiện được nhiều ưu điểm trong việc tìm giải pháp tối ưu cho các hàm toán chuẩn [8]. Ngoài ra, thuật toán RRA cũng đã được áp dụng thành công trong một số lĩnh vực như kỹ thuật điện [9], dự báo sản xuất [5]. Tính cấp thiết của bài viết là trình bày chi tiết các bước áp dụng thuật toán RRA để tìm đường đi ngắn nhất cho bài toán TSP hiệu quả hơn với thời gian ngắn hơn.

## 2. NỘI DUNG

### 2.1. Mô hình bài toán TSP

Cho $N_{city}$ thành phố và tọa độ của mỗi thành phố, chiều dài đường đi giữa các thành phố $path = [c_1, c_2, \dots, c_{N_{city}}]$ được xác định như sau: $L(path) = \sum_{i=1}^{N_{city}-1} d(c_i, c_{i+1}) + d(c_n, c_1)$ (1)

Trong đó, $d(c_i, c_{i+1})$ là khoảng cách Euler giữa hai thành phố được xác định như sau: $d(c_i, c_{i+1}) = \sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2}$(2)

Trong đó, $(x_i, y_i)$ và $(x_{i+1}, y_{i+1})$ lần lượt là tọa độ của hai thành phố i và j.

### 2.2. Thuật toán RRA cho bài toán TSP

Thuật toán RRA đã được áp dụng cho các hàm toán chuẩn. Tuy nhiên, để áp dụng thành công RRA cho bài toán TSP, các bước thực hiện của thuật toán cần được điều chỉnh [8]. Vì vậy, trong mục này chi tiết áp dụng thuật toán RRA cho bài toán TSP được mô tả như sau:

*Bước 1 - Khởi tạo:* Để tìm đường đi ngắn nhất cho bài toán TSP, mỗi đường đi khả thi được xem như một cây mẹ. Trong đó, mỗi biến của cây mẹ đại diện cho một thành phố như trong biểu thức: $path_{mother,i} = [c_1, c_2, \dots, c_j, \dots\ c_{Nc}]$ (3)

Trong đó, $path_{mother,i}$ là đường đi thư i trong quần thể N đường đi. $c_j$ là thành phố thứ j trong đường đi. $Nc$ là số thành phố.

Để bắt đầu quá trình tìm kiếm, mỗi đường đi được khởi tạo như sau: $path_{mother,i} = path_{low} + rand.(path_{high} - path_{low})$ (4)

Trong đó, $path_{low}$ và $path_{high}$ là véc tơ giới hạn của các biến trong đường đi.

Do mỗi thành phố được biểu diễn dưới dạng số nguyên dương, nên các phần tử trong đường đi $path_{mother,i}$ sẽ được sắp xếp theo thứ tự tăng dần. Khi đó, vị trí của các biến trong mảng đã được sắp xếp sẽ được quy ước là thành phố tương ứng với biến đó. Chẳng hạn như bài toán tìm đường đi ngắn nhất qua 4 thành phố, giá trị của $path_{mother,i} = [2.3, 5.6, 4.3, 1.8]$ thì đường đi tương ứng sẽ là [2, 4, 3, 1]. Sau khi được khởi tạo, các đường đi sẽ được đánh giá hàm mục tiêu như mô tả ở biểu thức (1) để xác định chiều dài của mỗi đường đi. Khi đó, đường đi có chiều dài ngắn nhất ($L_{best}$) sẽ được xem như là đường đi tốt nhất $path_{best}$.

*Bước 2: Tạo ra quần thể đường đi mới.* Ý tưởng mỗi cây mẹ sinh ra một cây con qua thân của nó được thể hiện trong thuật toán RRA trong quá trình tạo ra các giải pháp mới như sau:

$$path_{daughter,k} = \begin{cases} path_{best} & , k = 1 \\ path_{mother,k} + runner.rand, & k = 2, \dots, N \end{cases} (5)$$

Trong đó, $path_{daughter,k}$ là đường đi mới thứ k được tạo ra từ đường đi $path_{mother,k}$. $runner$ là khoảng cách lớn nhất từ biến hiện tại đến biến mới được đặt bằng 200.

Các đường đi mới sẽ được đánh giá hàm mục tiêu như mô tả ở biểu thức (1) để xác định chiều dài của mỗi đường đi $L_{daughter,k}$. Dựa trên chiều dài của mỗi đường đi mới, giá trị $path_{best}$ được cập nhật lại.

*Bước 3: Cập nhật đường đi tốt nhất.* Sau hai vòng lặp liên tiếp mà đường đi tốt nhất thu được không có sự cải thiện về chiều dài đường đi, thì thủ tục cập nhật đường đi tốt nhất được thực hiện. Trong đó, sẽ có 2. $Nc$ đường đi mới được tạo ra bằng cách điều chỉnh lần lượt từng biến của $path_{best}$ như sau:

$$path_{1,d} = M_{u,d}.runner.path_{best} + path_{best} \quad (6)$$

$$path_{2,d} = M_{u,d}.root.path_{best} + path_{best} \quad (7)$$

Trong đó, $path_{1,d}$ và $path_{2,d}$ là đường đi thứ d được tạo ra với d = 1, 2,…, $Nc$. $M_{u,d}$ là dòng thứ d của ma trân đơn vị có kích thước $(Nc \times Nc)$. $root$ là hằng số có giá trị nhỏ hơn so với $runner$, nó thường được chọn bằng ½ giá trị của $runner$.

Các đường đi mới được tính toán hàm mục tiêu. Nếu tồn tại đường đi có chiều dài ngắn hơn $path_{best}$ nó sẽ được chọn để thay thế cho $path_{best}$.

*Bước 4: Lựa chọn đường đi cho thế hệ tiếp theo.* Quần thể cây mẹ cho vòng lặp sau được chọn từ quần thể cây con. Khi đó, xác suất để cây con $path_{daughter,k}$ được chọn để trở thành một cây mẹ ở vòng lặp tiếp theo được xác định như sau:

$$p_k = \frac{f(path_{daughter,k})}{\sum_{j=1}^{N} f(path_{daughter,j})} \quad (8)$$

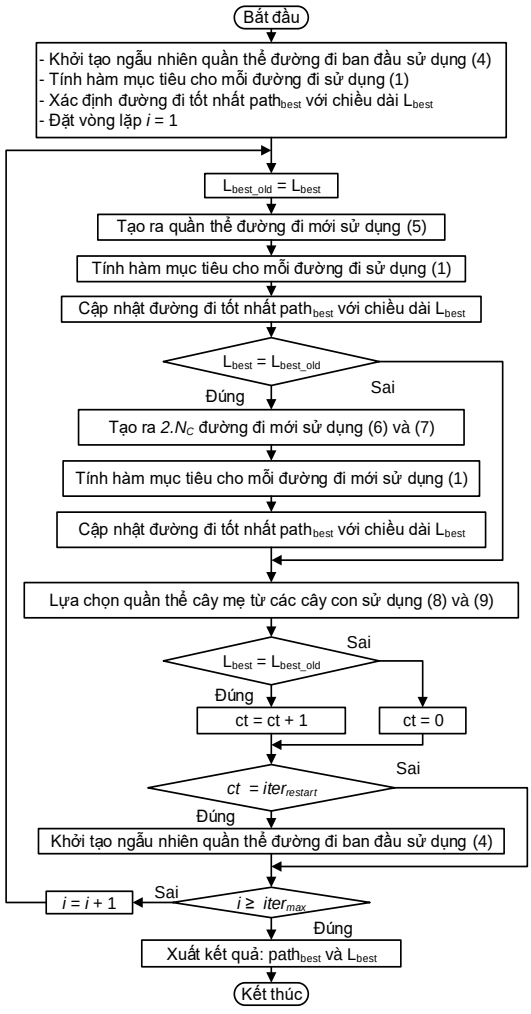
Trong đó, $f(path_{daughter,k})$ là giá trị thích nghi của cây con $path_{daughter,k}$ được tính như sau:

$$f(path_{daughter,k}) = \frac{1}{\sigma + L_{daughter,k} - L_{best}} \quad (9)$$

Trong đó, $\sigma$ là hằng số nhỏ nhằm loại trừ phép chia cho 0 trong trường hợp giá trị của $L_{daughter,k}$ và $L_{best}$ bằng nhau.

*Bước 5: Khởi động lại quần thể cây mẹ.* Trong trường hợp sau một số vòng lặp định trước ($iter_{restart}$) mà đường đi tốt nhất vẫn không được cải thiện, thì thuật toán RRA sẽ khởi tạo ngẫu nhiên lại quần thể cây mẹ bằng cách sử dụng biểu thức (4) nhằm gia tăng cơ hội khai phá không gian tìm kiếm.

*Bước 6: Kiểm tra điều kiện dừng.* Thuật toán RRA cho bài toán TSP hoạt động dựa trên số lượng vòng lặp tối đa. Nếu số vòng lặp hiện tại chưa đạt tới giá trị lớn nhất đặt trước ($iter_{max}$), thuật toán sẽ quay trở lại bước 2 để tiếp tục quá trình tìm kiếm. Ngược lại, thuật toán sẽ được dừng lại và đường đi tốt nhất $path_{best}$ được xem như là lời giải cho bài toán. Toàn bộ các bước thực hiện của thuật toán RRA cho bài toán TSP được trình bày trong lưu đồ ở Hình 1.




**Hình 1.** Lưu đồ thuật toán RRA cho bài toán TSP

## 2.3. Kết quả tính toán

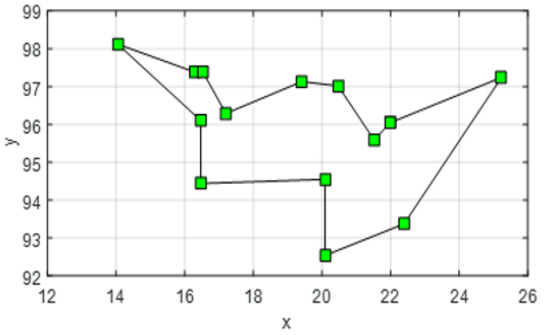
Trong phần này, phương pháp RRA đề xuất được sử dụng để tìm đường đi ngắn nhất cho một bài toán TSP có 14 thành phố được gọi là Burma14 [13]. Trong đó, tọa độ của mỗi thành phố được cho trong Bảng 1. Phương pháp đề xuất RRA được lập trình từ phần mềm Matlab 2016a chạy trên máy tính cá nhân CPU core i5 2x2.4GH, 8GB RAM. Thông số RRA

bao gồm kích thước quần thể N và số vòng lặp lớn nhất $iter_{max}$ được chọn lần lượt bằng 20 và 50. Trong khi đó, giá trị của $iter_{restart}$ được chọn bằng ½ giá trị của $iter_{max}$. Phương pháp đề nghị được chạy 100 lần độc lập để đánh giá hiệu quả của phương pháp.

Kết quả tính toán cho bài toán được trình bày ở Bảng 2. Trong đó, chiều dài đường đi ngắn nhất thu được từ RRA cho Burma14 là 30.8785 tương ứng với đường đi ngắn nhất là {13, 7, 5, 12, 6, 4, 3, 14, 2, 1, 10, 9, 11, 8} được mô tả ở Hình 2. Kết quả này hoàn toàn tương tự với kết quả thu được từ phương pháp PSO [16]. Tuy nhiên, giá trị trung bình của hàm mục tiêu thu được bởi RRA trong 100 lần chạy bé hơn so với phương pháp PSO [16]. Đặc tuyến hội tụ lớn nhất, nhỏ nhất và trung bình trong 100 lần chạy độc lập ở Hình 3a cho thấy đường đặc tuyến hội tụ trung bình rất gần với đường hội tụ nhỏ nhất. Giá trị hàm mục tiêu thu được trong 100 lần chạy ở Hình 3b cho thấy RRA có thể tìm được đường đi tốt nhất trong hầu hết các lần chạy. Điều này cho thấy, sự ổn định và độ tin cậy cao của thuật toán RRA cho bài toán TSP.

**Bảng 1.** Tọa độ các thành phố của bài toán Burma14

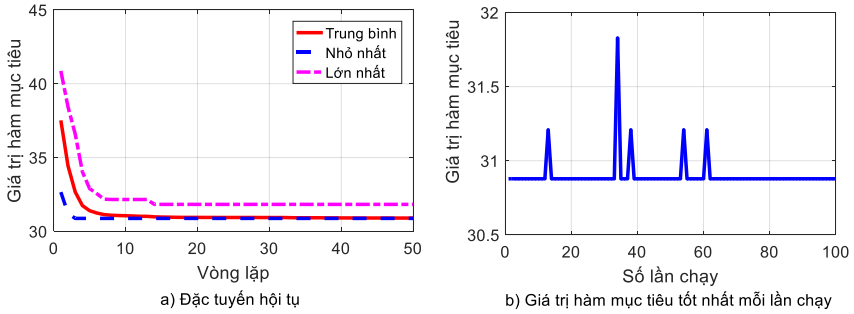
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **x** | 16.47 | 16.47 | 20.09 | 22.39 | 25.23 | 22.00 | 20.47 |
| **y** | 96.10 | 94.44 | 92.54 | 93.37 | 97.24 | 96.05 | 97.02 |
| **x** | 17.20 | 16.30 | 14.05 | 16.53 | 21.52 | 19.41 | 20.09 |
| y | 96.29 | 97.38 | 98.12 | 97.38 | 95.59 | 97.13 | 94.55 |




**Hình 2.** Đường đi tốt nhất thu được từ RRA

**Bảng 2.** Kết quả tính toán cho bài toán TSP Burma14

| Phương pháp | Đường đi tốt nhất | $L_{max}$ | $L_{min}$ | $L_{mean}$ | STD | Thời gian tính toán (s) |
|---|---|---|---|---|---|---|
| RRA | 13, 7, 5, 12, 6, 4, 3, 14, 2, 1, 10, 9, 11, 8 | 31.8283 | 30.8785 | 30.9012 | 0.1140 | 0.2098 |
| PSO [16] | 13, 7, 5, 12, 6, 4, 3, 14, 2, 1, 10, 9, 11, 8 | 31.022 | 30.8785 | 30.9245 | - | - |




**Hình 3.** Đặc tuyến hộ tụ trung bình và giá trị hàm mục tiêu trong 100 lần chạy

## 3. KẾT LUẬN

Bài viết trình bày ứng dụng của thuật toán RRA cho bài toán TSP. Hiệu quả của RRA được kiểm chứng trên bài toán TSP có 14 thành phố. Kết quả thu được bởi RRA trong 100 lần chạy độc lập được so sánh với phương pháp PSO đã thực hiện. RRA là một phương pháp hiệu quả và tin cậy trong việc tìm lời giải cho bài toán TSP. Trong tương lai, chúng tôi sẽ sử dụng thuật toán RRA cho các bài toán TSP có nhiều thành phố hơn cũng như áp dụng phương pháp đề xuất vào một số ứng dụng thực tế.

## TÀI LIỆU THAM KHẢO

[1] Boryczka U. and Szwarc K. (2019), *The Harmony Search algorithm with additional improvement of harmony memory for Asymmetric Traveling Salesman Problem*, 122, Elsevier, Expert Systems with Applications.

[2] Campuzano G., Obreque C., and Aguayo M. M. (2020), *Accelerating the Miller–Tucker–Zemlin model for the asymmetric traveling salesman problem*, 148, Elsevier, Expert Systems with Applications.

[3] Dong X. and Cai Y. (2018), *A novel genetic algorithm for large scale colored balanced traveling salesman problem*, 95, Elsevier, Future Generation Computer Systems.

[4] Ebadinezhad S. (2020), *DEACO: Adopting dynamic evaporation strategy to enhance ACO algorithm for the traveling salesman problem*, 92, Elsevier, Engineering Applications of Artificial Intelligence.

[5] Goli A., Moeini E., Shafiee A. M., Zamani M., and Touti E. (2020), *Application of Improved Artificial Intelligence with Runner-Root Meta-Heuristic Algorithm for Dairy Products Industry: A Case Study*, 29(5), World Scientific Publishing, International Journal on Artificial Intelligence Tools.

[6] Hu Y., Yao Y., and Lee W. S. (2020), *A reinforcement learning approach for optimizing multiple traveling salesman problems over graphs*, 204, Elsevier, Knowledge-Based Systems.

[7] Laporte G. (1992), *The traveling salesman problem: An overview of exact and approximate algorithms*, 59 (2), Elsevier, European Journal of Operational Research.

[8] Merrikh-Bayat F. (2015), *The runner-root algorithm: A metaheuristic for solving unimodal and multimodal optimization problems inspired by runners and roots of plants in nature*, 33, Elsevier, Applied Soft Computing.

[9] Nguyen T. T., Nguyen T. T., Truong A. V., Nguyen Q. T., and Phung T. A. (2017), *Multi-objective electric distribution network reconfiguration solution using runner-root algorithm*, 52, Elsevier, Applied Soft Computing.

[10] Oliver I. M., Smith Dj. And Holland J. R. (1987), *Study of permutation crossover operators on the traveling salesman problem*, The Massachusetts Institute of Technology, Genetic algorithms and their applications: proceedings of the second International Conference on Genetic Algorithms.

[11] Ouaarab A., Ahiod B., and Yang X. S. (2014), *Discrete cuckoo search algorithm for the travelling salesman problem*, 24, Springer, Neural Computing and Applications.

[12] Pandiri V. and Singh A. (2019), *An artificial bee colony algorithm with variable degree of perturbation for the generalized covering traveling salesman problem*, 78, Elsevier, Applied Soft Computing.

[13] Reinelt G. (1991), *TSPLIB. A traveling salesman problem library*, 3(4), The Institute for Operations Research and the Management Sciences, ORSA journal on computing.

[14] Salman R., Ekstedt F., and Damaschke P. (2020), *Branch-and-bound for the Precedence Constrained Generalized Traveling Salesman Problem*, 48(2), Elsevier, Operations Research Letters.

[15] Yuan Y., Cattaruzza D. Ogier M. and Semet F. (2020), *A branch-and-cut algorithm for the generalized traveling salesman problem with time windows*, 286, Elsevier, European Journal of Operational Research.

[16] Yuan Z., Yang L., Wu Y., Liao L., and Li G. (2007), *Chaotic particle swarm optimization algorithm for traveling salesman problem*, IEEE, Proceedings of the IEEE International Conference on Automation and Logistics.

Ngày nhận bài: 26-8-2020. Ngày biên tập xong: 11-01-2021. Duyệt đăng: 22-01-2021