JTE HCMUTE
ISSN: 2615-9740

JOURNAL OF TECHNICAL EDUCATION SCIENCE
Ho Chi Minh City University of Technology and Education
Website: https://jte.edu.vn
Email: jte@hcmute.edu.vn

# Application of Optimization Algorithms in Reliable Pathfinding Models for Smart Cities

**Lap Duy Le[1*], Van Long Nguyen[1], Thanh Tuan Vu[2]**
[1]*Ho Chi Minh City University of Technology and Education, Vietnam*
[2]*FPT Software Company Limited, Vietnam*

[*]Corresponding author. *Email: 2391301@student.hcmute.edu.vn*

**ARTICLE INFO**

Received: 17/03/2025
Revised: 15/04/2025
Accepted: 10/06/2025
Published: 28/08/2025

**KEYWORDS**

Optimization Algorithms;
Pathfinding;
Smart Cities;
RAO-3;
Q-Learning.

**ABSTRACT**

This article implements and compares two potential algorithms in urban traffic management: the RAO-3 optimization algorithm and the Q-Learning reinforcement learning algorithm. The research is conducted on the dynamic traffic density optimization system platform, which collects real-time traffic data from IoT devices and processes it through fog computing infrastructure. The implementation of RAO-3 and Q-Learning on this rich dataset can be considered a groundbreaking contribution, helping to identify a more optimal algorithm for traffic flow and routing based on current conditions. The core idea of the research is to manually create a set of sample data while also extracting data from the dynamic traffic density optimization system, then testing this dataset with the RAO-3 and Q-Learning algorithms. The results indicate that Q-Learning outperforms RAO-3 in terms of efficiency and accuracy. This serves as a foundation for future advancements in smart city technology, emphasizing the role of integrating advanced technology in promoting more sustainable, efficient, and safer urban environments.

Doi: https://doi.org/10.54644/jte.2025.1847

Copyright © JTE. This is an open access article distributed under the terms and conditions of the Creative Commons Attribution-NonCommercial 4.0 International License which permits unrestricted use, distribution, and reproduction in any medium for non-commercial purpose, provided the original work is properly cited.

## 1. Introduction

As urban areas develop, people's lives improve, and with a growing population, the average household now owns at least two motorbikes and often a car, driving higher transportation demand. This surge presents major challenges: congestion, traffic accidents, infrastructure strain, environmental pollution, and resource depletion [1]. These issues increasingly affect quality of life. To help address them, researching and applying optimal route planning algorithms is essential, particularly as local governments pursue smart city initiatives.

In Vietnam, several notable studies have been conducted. One used Dijkstra's algorithm to find the shortest path between two non-empty subsets of vertices in a graph [2]. Another focused on parallelizing Dijkstra's algorithm to compute shortest paths from a single vertex to all others in large graphs using multiple processors [3], though optimal data partitioning is still needed to reduce inter-processor communication. Additionally, another study modified the Bellman-Ford algorithm to handle extended graphs with variable edge weights, solving the problem despite high complexity and limited large-scale testing [4].

Internationally, numerous studies since the 20th century have driven major advancements. One study [5] compared eight shortest path algorithms on directed graphs, analyzing performance, implementation, and data structures, grouping them by linked lists or priority queues. Study [6] provides a theoretical evaluation of shortest path and maximum flow algorithms, aiding future research. Study [7] proposes an efficient algorithm for finding multiple non-overlapping shortest paths between two points by combining a shortest path tree with a heap to generate and search subgraphs. Study [8] offers a more

practical approach, applying the A-star algorithm to pathfinding in dynamic, obstacle-filled environments.

Finding optimal paths is also critical in many other fields. In supply chain management, study [9] proposed an optimized warehouse picking route using the A-star algorithm. In transportation, study [10] introduced a new method combining a genetic algorithm with a reward-penalty strategy, achieving significant improvements. In safety and risk management, study [11] integrated Dijkstra's algorithm with IoT technologies to provide real-time optimal evacuation routes. Another transportation study [12] combined Dijkstra's algorithm with the Cuckoo Search and Whale Optimization algorithms to improve path efficiency in fixed road networks.

Most large enterprises now use third-party mapping services like Google Maps, OpenStreetMap, HERE Maps, and Waze, which integrate various pathfinding algorithms to optimize speed, performance, and accuracy. However, many businesses aim to develop their own mapping systems to lower costs and tailor features to their specific needs.

Overall, these studies have been well analyzed, but most remain academic and theoretical, with limited real-world application. This raises the question of how to implement pathfinding algorithms effectively in practice. A key application is providing reliable routing for smart city transportation. Our research explores two algorithms - RAO-3 optimization and Q-Learning reinforcement learning - to address optimal route planning in complex traffic environments. Unlike prior studies focused on theory and performance, this work tests the algorithms in realistic simulations to assess practical applicability and offer insights for intelligent transportation systems. It also supports the development of autonomous solutions for businesses and local governments building smart cities.

This research focuses on urban traffic networks, considering factors such as roadways, intersections, traffic density, congestion, accidents, weather, and military events or local events that affect traffic flow. It targets optimal pathfinding algorithms - specifically RAO-3 and Q-Learning - to identify effective solutions for urban transportation. A dynamic traffic density optimization system is used to extract experimental data, leveraging IoT and fog computing to collect and process real-time information. By applying these algorithms, the system improves vehicle coordination, reduces congestion, and enhances traffic management in smart cities.

The following content of this research is divided into four sections. Section 2 focuses on constructing and modeling the optimal pathfinding problem, introducing and implementing the algorithms to solve this problem. Additionally, this section presents the model of a dynamic traffic density optimization system, which is used to extract real-world map data for experimentation. Section 3 provides detailed information on the experimental process and analyzes the obtained results. Section 4 discusses the effectiveness of the tested algorithms. Finally, section 5 presents the concluding summary of the research.

## 2. Proposed Solution

### *2.1. Constructing the Optimal Pathfinding Problem*

Optimal pathfinding is a fundamental problem in computer science, aiming to determine the shortest path from a given point to any other within a graph or network. Though classical, it remains an active research area, with ongoing efforts to enhance algorithm efficiency and accuracy to tackle more complex, real-world problems.

To identify an optimal pathfinding method for dynamic traffic management in smart cities, it is essential to validate algorithms that balance simplicity and computational efficiency - key for real-time applications requiring rapid decisions. The problem can be described as follows: Given a directed graph $G$ representing the urban traffic network:

$$G = (V, E, W) \quad (1)$$

JTE
HCMUTE
ISSN: 2615-9740

JOURNAL OF TECHNICAL EDUCATION SCIENCE
Ho Chi Minh City University of Technology and Education
Website: https://jte.edu.vn
Email: jte@hcmute.edu.vn

Where $V$ is the set of network nodes (representing major intersections, road entry, and exit points), $E$ is the set of edges in the graph (representing road segments), and $W$ is the cost associated with the edges (representing the length and traffic density of the road segment). The objective of this problem is to determine an algorithm $A$ that can find a path $P = \{v_0, v_1, \ldots, v_k\}$ where $v_0 \in V$ is the starting node and $v_k \in V$ is the destination node. The path $P$ must satisfy the condition:

$$Minimize\ C(P) = \sum_{i=0}^{k-1} W(v_i, v_{i+1}) \tag{2}$$

Where $C(P)$ is the cost of the path $P$ and $C(P)$ must be minimized; additionally, each edge $(v_i, v_{i+1}) \in E$.

### *2.2. Algorithms for Solving the Optimal Pathfinding Problem*

#### *2.2.1. The RAO-3 Optimization Algorithm*

The RAO-3 algorithm [13] is a simple, non-metaphorical optimization method that does not rely on analogies to nature or biology and is parameter-free. It improves solutions by interacting with the best and worst candidates in the population and updates values using random coefficients.

The working principle of this algorithm is as follows: Let $f(x)$ be the objective function to be minimized (or maximized). This function $f(x)$ is defined based on the specific problem at hand. At iteration $i$, assume there are "$m$" design variables (adjustable variables) and "$n$" candidate solutions. Let $f(x)_{best}$ be the best candidate solution, and $f(x)_{worst}$ be the worst candidate solution. If $X_{j,k,i}$ represents the value of the $j^{th}$ variable for the $k^{th}$ candidate solution in iteration $i$, then $X_{j,k,i}$ value will be updated as follows:

$$\boldsymbol{X'_{j,k,i} = X_{j,k,i} + r_{1,j,i}\left(X_{j,best,i} - \left|X_{j,worst,i}\right|\right) + r_{2,j,i}\left(\left|X_{j,k,i}\ \textbf{or}\ X_{j,l,i}\right| - \left(X_{j,l,i}\ \textbf{or}\ X_{j,k,i}\right)\right)} \tag{3}$$

Where $X_{j,best,i}$ is the value of variable $j$ for the best candidate solution in iteration $i$; $X_{j,worst,i}$ is the value of variable $j$ for the worst candidate solution in iteration $i$; $r_{1,j,i}$ and $r_{2,j,i}$ are two random numbers in the range [0, 1]; $X'_{j,k,i}$ is the updated value of $X_{j,k,i}$ using $r_{1,j,i}$ and $r_{2,j,i}$; "$X_{j,\boldsymbol{k},i}$ *or* $X_{j,\boldsymbol{l},i}$": If the fitness value of solution $k$ is better than that of $l$, then $X_{j,\boldsymbol{k},i}$ is chosen; otherwise $X_{j,\boldsymbol{l},i}$ is chosen (i.e., the better solution is selected); "$X_{j,\boldsymbol{l},i}$ *or* $X_{j,\boldsymbol{k},i}$" If the fitness value of solution $l$ is worse than that of $k$, then $X_{j,\boldsymbol{l},i}$ is chosen; otherwise $X_{j,\boldsymbol{k},i}$ is chosen (i.e., the worse solution is selected).

Figure 1 provides a visual representation of the working principle of the algorithm:

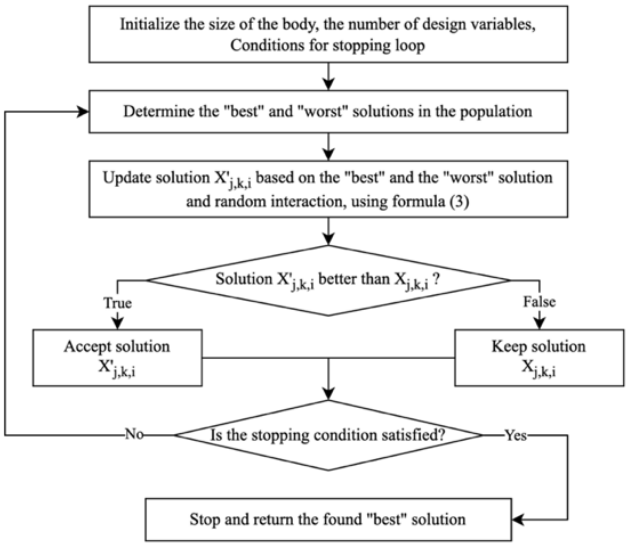



**Figure 1.** *Flowchart of the RAO-3 Algorithm Operation*

*2.2.2. Q-Learning Reinforcement Learning Algorithm*

The Q-Learning algorithm [14] is a reinforcement learning algorithm that enables agents to learn optimal policies in Markov decision processes. It aims to find the best action for each state by maximizing the expected cumulative reward through trial-and-error interactions with the environment.

The working principle of this algorithm is as follows: Q-Learning works by using a Q-table to store Q-values, which estimate the expected future rewards for taking specific actions in given states. These values are initialized (randomly or as zero), and learning proceeds over multiple episodes using parameters such as learning rate (α), discount factor (γ), and a chosen learning strategy (e.g., epsilon-greedy, Softmax, Boltzmann, Random, etc.). At each step, the agent selects an action, receives a reward, and transitions to a new state, updating the Q-values accordingly. The Q-value is then updated using the following formula:

$$\boldsymbol{Q^{new}(s_t, a_t) = (1 - \alpha) \cdot Q(s_t, a_t) + \alpha \cdot \left(r_t + \gamma \cdot max_a Q(s_{t+1}, a)\right)} \quad (4)$$

Where $s_t$ is the state at time $t$; $a_t$ is the action taken at time $t$; α is the learning rate; $r_t$ is the reward received when performing action $a_t$ in state $s_t$ at time $t$; γ is the discount factor; $max_aQ(s_{t+1}, a)$ is the maximum Q-value currently stored in the Q-table for the next state $s_{t+1}$.

Figure 2 provides a visual representation of the working principle of the algorithm:

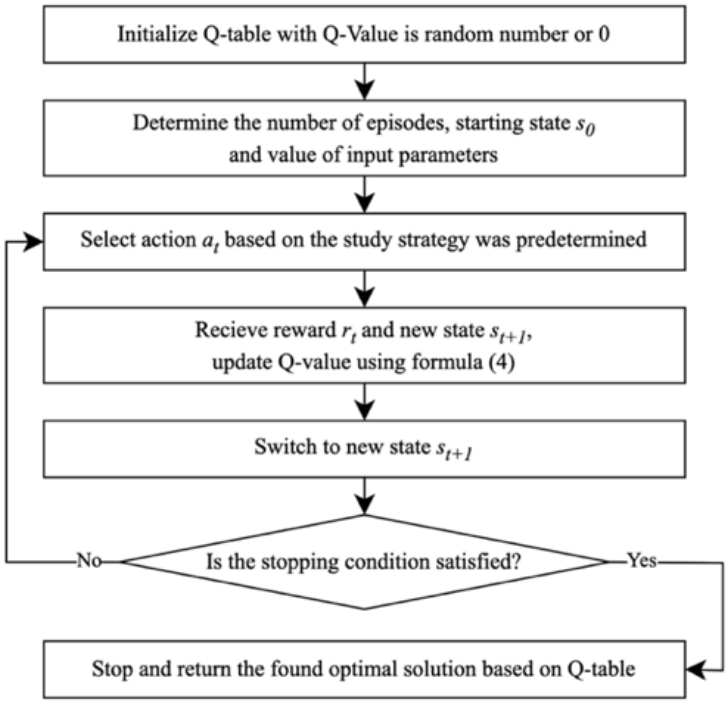


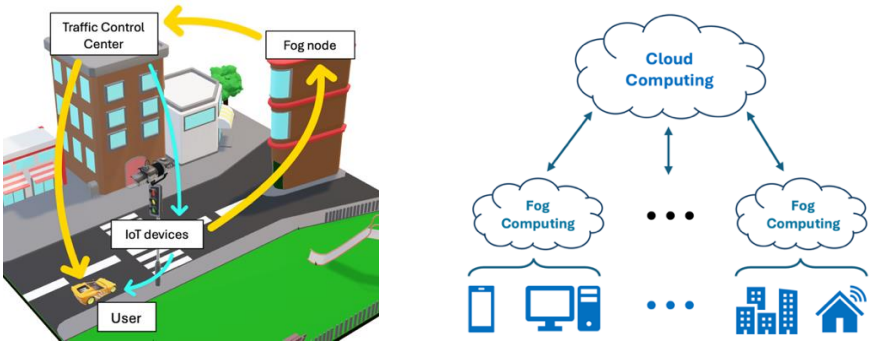

**Figure 2.** *Flowchart of the Q-Learning Algorithm Operation*

### *2.3. The Dynamic Traffic Optimization System*

This research utilizes the Dynamic Traffic Optimization System (DTOS) [15] to extract real-world traffic data for experiments using the RAO-3 and Q-Learning algorithms. The extracted data serves as the basis for analysis, evaluation, and comparison of these two algorithms.

The Dynamic Traffic Optimization System (DTOS) is designed to build a database of optimal routes within urban traffic networks, addressing key traffic challenges through advanced pathfinding. DTOS integrates three components: OpenStreetMap (OSM), IoT devices, and fog computing. The model of this system and three key components is visually represented in Figure 3.

JTE HCMUTE
ISSN: 2615-9740
JOURNAL OF TECHNICAL EDUCATION SCIENCE
Ho Chi Minh City University of Technology and Education
Website: https://jte.edu.vn
Email: jte@hcmute.edu.vn




*(a)* *(b)*

**Figure 3.** *DTOS Model in a Smart City: (a) Visually model; (b) Three key components of the model*

Traditional pathfinding methods assume static networks, failing to reflect the dynamic nature of urban traffic shaped by incidents and fluctuations. This study explores RAO-3 and Q-Learning algorithms, integrating real-time data for adaptive route optimization. The approach improves responsiveness to current traffic, advancing real-time traffic management.

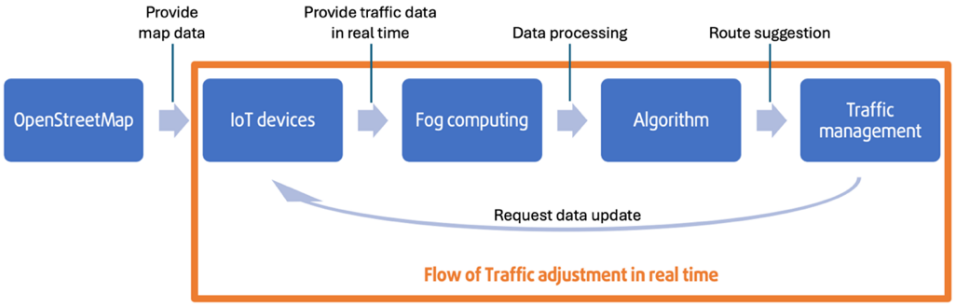



**Figure 4.** *DTOS Operational Process*

Figure 4 illustrates the real-time traffic optimization process. This study will extract distance matrix data in the first step of this process, specifically as follows:

**Step 1**. Identify the Area for Data Extraction. The system queries APIs from OpenStreetMap and Overpass Turbo[1] to retrieve map data, extracting key nodes such as intersections, entry/exit points, and adjacent nodes along routes. These nodes are used to calculate distances and initialize a distance matrix, which forms the basis for traffic flow analysis using algorithm (5):

$$\boldsymbol{X(i,j)} = \begin{cases} \boldsymbol{XX(i,j)}, & \boldsymbol{XX(i,j) \neq -1}. \\ \boldsymbol{Inf}, & \boldsymbol{XX(i,j) = -1}. \end{cases} \quad (5)$$

where $X$ is the distance matrix; $XX$ is the input data matrix of size $N \times N$ obtained after querying the APIs and computing distances between key network nodes; $N$ is the number of key network nodes (intersections, road entry/exit points); $i$, $j$ are natural numbers representing the rows and columns, iterating from $1$ to $N$.

**Step 2.** Export the Distance Matrix. The matrix $X$ is exported and saved as an Excel file.

### *2.4. Algorithms Implementation*

*2.4.1. RAO-3 Algorithm*

| **Input:** Starting node *S*, destination node *T*, weight matrix |
|---|

[1] Overpass API. https://wiki.openstreetmap.org/w/index.php?title=Overpass_API

**Output:** Optimal path or empty (if no path is found)

```
vertex_amount ← Number of rows or columns in the weight matrix
best_solution ← empty                          // Initialize the best solution
for (i < run_times):                           // Iterate for a predefined number of times
    Initialize a population with p_size random individuals
    for (j < max_iter):                        // Iterate for a predefined number of times
        Find the current best (best) and worst (worst) solutions
        for (solution_k < p_size):
            rand_solution ← a random solution from the population
            Compute new_solution based on the best solution, worst solution, solution_k,
            and rand_solution using formula (3)
            Trim each element of new_solution.path to ensure values remain within the range
            [0, vertex_amount – 1]
            Replace invalid values in new_solution.path
            Compute new_solution.cost from new_solution.path
            if (new_solution.cost < solution_k.cost):
                solution_k ← new_solution
                if (new_solution.cost < best_solution.cost):
                    best_solution ← new_solution
return best_solution.path
```

*2.4.2. Q-Learning Algorithm*

**Input:** Starting node *S*, destination node *T*, weight matrix

**Output:** Optimal path or empty (if no path is found)

```
max_cost ← Longest path length
episode ← Number of learning iterations
list_states ← List of all nodes
Initialize Q_table with all Q_values set to 0
dead_ends ← empty                              // Stores dead-end nodes
for (i < episode):
    state ← Starting node S
    visited ← empty                            // Stores visited nodes
    while (state != Destination node T):
        list_actions ← List of feasible actions (Unvisited nodes that are not dead ends and
        do not match the current state)
        if (list_actions.length = 0):
            dead_ends.add(state)
            break while
        α' ← α / (1 + episode)
        if (random(0,1) > ε * e^(-0.1*episode)):
            action ← random(list_actions)      // Exploration
```

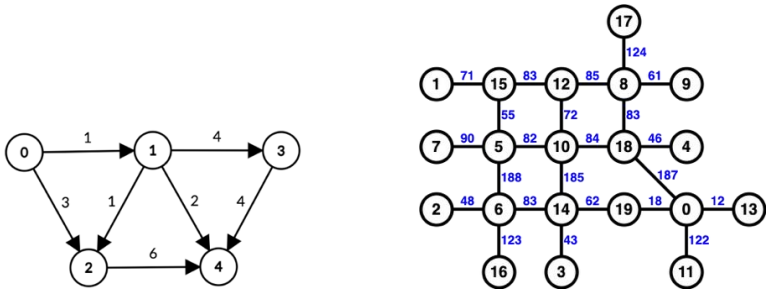
```
        else:
            action ← max_Q_value(list_actions)      // Exploitation
        if (action = Destination node T):
            reward ← 10 * max_cost
        else:
            reward ← state.action.cost * (-1)
        Update Q-value using formula (4) with learning rate α'
        visited.add(state)
        state ← action
return Q_table
Retrieve the optimal learned path from Q_table
```

## 3. Experiments and results

### *3.1. Preparation of experimental data*

#### *3.1.1. Manually created data*

This research generates two sample datasets consisting of 5 network nodes and 20 network nodes. Figures 5 provide a visual representation of the graphs corresponding to these two datasets.




*(a)* *(b)*

**Figure 5.** *Manually created sample datasets 1 and 2:*
*(a) 5-network-node graph (directed); (b) 20-network-node graph (bidirectional)*

#### *3.1.2. Data collected using the DTOS*

This research utilizes the DTOS to extract road network data from Linh Chieu Ward, Thu Duc City, Ho Chi Minh City. This sample dataset consists of 524 network nodes, with data collected in April 2025.

### *3.2. Experimental results*

#### *3.2.1. Sample dataset 1: 5 network nodes*

The initial parameter settings are as follows: $S$ = 0; $T$ = 4; Optimal expected result: [0, 1, 4] = 3. Table 1 presents the results after executing the RAO-3 algorithm on Sample dataset 1, which consists of 5 network nodes.

**Table 1.** *Experimental results – Sample dataset 1 – RAO-3*

| Variables | Iteration | Result Found |
|---|---|---|
| max_iter = 10<br>p_size = 10 | 1 (0,1s) | [0, 1, 4] = 3 |
| | 2 (0,1s) | [0, 1, 4] = 3 |
| | 3 (0,1s) | [0, 1, 4] = 3 |

Table 2 presents the results after executing the Q-Learning algorithm on Sample dataset 1, which consists of 5 network nodes.

**Table 2.** *Experimental results – Sample dataset 1 – Q-Learning*

| Parameters | Iteration | Result Found |
|---|---|---|
| episode = 100<br>$\alpha = 0.1; \gamma = 0.9$<br>$\varepsilon = 0.9$ | 1 (0,2s) | [0, 1, 4] = 3 |
| | 2 (0,1s) | [0, 1, 4] = 3 |
| | 3 (0,1s) | [0, 1, 4] = 3 |

*3.2.2. Sample dataset 2: 20 network nodes*

The initial parameter settings are as follows: $S = 0$; $T = 15$; Optimal expected result: [0, 19, 14, 6, 5, 15] = 406. Table 3 presents the results after executing the RAO-3 algorithm on Sample dataset 2, which consists of 20 network nodes.

**Table 3.** *Experimental results – Sample dataset 2 – RAO-3*

| Variables | Iteration | Result Found |
|---|---|---|
| max_iter = 200<br>p_size = 500 | 1 (5,8s) | [0, 18, 10, 12, 15] = 426 |
| | 2 (4,4s) | 0, 18, 10, 12, 15] = 426 |
| | 3 (4,6s) | Result not found |
| max_iter = 200<br>p_size = 1000 | 1 (8,4s) | [0, 18, 10, 12, 15] = 426 |
| | 2 (8,5s) | Result not found |
| | 3 (8,5s) | [0, 19, 14, 10, 12, 15] = 421 |

Table 4 presents the results after executing the Q-Learning algorithm on Sample dataset 2, which consists of 20 network nodes.

**Table 4.** *Experimental results – Sample dataset 2 – Q-Learning*

| Parameters | Iteration | Result Found |
|---|---|---|
| episode = 100<br>$\alpha = 0.1; \gamma = 0.9$<br>$\varepsilon = 0.9$ | 1 (0,1s) | [0, 19, 14, 6, 5, 15] = 406 |
| | 2 (0,2s) | [0, 18, 10, 12, 15] = 426 |
| | 2 (0,1s) | [0, 18, 10, 12, 15] = 426 |

*3.2.3. Sample dataset 3: Linh Chieu Ward, Thu Duc City, Ho Chi Minh City (524 network nodes)*

The initial parameter settings are as follows: $S = 15$; $T = 133$; Optimal expected result: [15, 132, 128, 418, 130, 133] = 235,95.

The results obtained after applying the RAO-3 algorithm to Sample Dataset 3, which consists of 524 network nodes, indicate that no viable paths were found. For detailed variable values and runtime information, please refer to the "Data Availability Statement" at the end of this article.

Table 5 presents the results after executing the Q-Learning algorithm on Sample dataset 3, which consists of 524 network nodes.

**Table 5.** *Experimental results – Sample dataset 3 – Q-Learning*

| Parameters | Iteration | Result Found |
|---|---|---|
| episode = 100 | 1 (0,1s) | [15, 132, 134, 130, 133] = 291.18 |

| | | |
|---|---|---|
| α = 0.1; γ = 0.9<br>ε = 0.9 | 2 (0,1s) | [15, 2, 22, 123, 125, 418, 130, 133] = 416.72 |
| | 3 (0,1s) | Result not found |
| episode = 1000<br>α = 0.1; γ = 0.9<br>ε = 0.9 | 1 (0,3s) | [15, 110, 9, 183, 270, 269, 516, 518, 370, 510, 12, 135, 4, 133] = 946.29 |
| | 2 (0,4s) | [15, 132, 134, 130, 133] = 291.18 |
| | 3 (0,4s) | [15, 2, 22, 263, 333, 519, 25, 509, 227, 226, 225, 279, 223, 224, 194, 52, 102, 433, 107, 172, 173, 174, 175, 415, 103, 106, 256, 258, 63, 267, 282, 233, 464, 221, 222, 11, 512, 220, 230, 78, 505, 79, 456, 20, 506, 167, 3, 289, 264, 68, 31, 325, 324, 272, 323, 322, 417, 321, 359, 69, 74, 76, 73, 72, 163, 179, 57, 56, 136, 135, 4, 133] = 4648.6 |
| episode = 5000<br>α = 0.1; γ = 0.9<br>ε = 0.9 | 1 (1,6s) | [15, 110, 9, 183, 270, 269, 516, 518, 370, 134, 130, 133] = 735.97 |
| | 2 (1,8s) | [15, 110, 9, 181, 516, 269, 270, 29, 30, 244, 341, 340, 243, 377, 444, 437, 438, 161, 160, 159, 158, 156, 12, 135, 4, 133] = 1496.64 |
| | 3 (1,9s) | [15, 132, 134, 130, 133] = 291.18 |

## 4. Discussion

This research implemented and compared two algorithms, RAO-3 and Q-Learning, for optimal pathfinding across three datasets of increasing scale. Results show that in smaller networks, the likelihood of finding an optimal path is high, but accuracy drops significantly in larger networks.

For small and medium-sized datasets (5 and 20 nodes), the RAO-3 algorithm finds results close to the optimal solution. However, as the dataset grows to 524 nodes, it struggles and fails to find the optimal path. RAO-3 also shows instability by often failing to find any path in large networks. This limitation may stem from its reliance on random solution initialization, which makes it less effective with complex networks. In contrast, the Q-Learning algorithm performs better across all datasets, including large networks. It consistently finds optimal or near-optimal paths, showing greater stability and an ability to handle large-scale data. Regarding scalability, RAO-3 remains limited as increasing nodes or population size does not improve outcomes, highlighting the need for further enhancement. Q-Learning, on the other hand, scales better because its reinforcement learning approach allows it to adapt and learn from the environment. However, its runtime can increase significantly with more learning episodes. RAO-3 also proves inadequate for optimal pathfinding due to its random initialization, while Q-Learning offers more flexibility. It can adaptively tune parameters such as ε (exploration) and α (learning rate), enabling it to learn and improve with each iteration.

Overall, RAO-3 is a relatively new approximation-based search algorithm that generates random solutions without utilizing the graph structure (such as edge weights), which significantly limits its efficiency. In contrast, Q-Learning identifies the optimal path through interaction with the environment. Its key advantage lies in its capacity to learn and adapt to changes in the environment (e.g., variations in edge weights). However, its performance depends heavily on the initial parameter settings and their adjustment during the learning process. Moreover, as a machine learning algorithm, Q-Learning operates with inherent accuracy limitations and rarely achieves absolute precision.

Both algorithms have their own strengths and limitations. However, it is evident that Q-Learning has a clear advantage in solving pathfinding problems in large networks due to its adaptive learning capabilities. While RAO-3 shows potential, it requires further improvements to be more effective in complex and large-scale data scenarios.

Table 6 presents a comparison between the two algorithms across various aspects to further clarify their suitability for the optimal pathfinding problem.

**Table 6.** *Comparison of RAO-3 and Q-Learning Algorithms in the Optimal Pathfinding Problem*

| Criterias | RAO-3 Algorithm | Q-Learning Algorithm |
|---|---|---|
| Convergence ability | Not guaranteed due to the algorithm's randomness | Converges if trained sufficiently; higher convergence likelihood than RAO-3 |
| Convergence stability | Low due to randomness in each run | High as it relies on learned results to produce outcomes |
| Accuracy | High for small networks and low for large networks | High for small networks, with slightly reduced accuracy for large networks |
| Scalability | Limited to small-scale networks | Can be scaled to large-scale networks |
| Execution time | Slow due to restarting from scratch when input changes | Fast as it relies on previously learned results |
| Initial efficiency | Depends on the number of iterations and population size | Depends on initial parameter settings and how they change during learning |
| Efficiency with updated data | Requires re-running the algorithm from scratch | Can continue learning from previous knowledge, gradually updating solutions based on new data |
| Computational complexity | Proportional to the number of iterations and population size | Proportional to the state and action space |
| Implementation | Simple | Requires tuning parameters appropriately to prevent divergence |
| Biggest limitation | Depends on randomness | Depends on parameter values |
| Adaptability to dynamic environments | Low, as every change requires rerunning the algorithm from scratch | High, as the algorithm gradually adapts through repeated learning |

## 5. Conclusion

This research has implemented and compared two algorithms, RAO-3 and Q-Learning, for the optimal pathfinding problem using three datasets of increasing size. The experimental results reveal clear differences in performance, stability, scalability, and flexibility between the two algorithms.

The research findings confirm that the Q-Learning algorithm is a superior solution for the optimal pathfinding problem in large-scale and highly complex networks. Its ability to learn and adapt from the environment allows it to identify near-optimal or optimal paths, even when faced with increasing network complexity. The Q-Learning algorithm has demonstrated exceptional stability, maintaining consistent performance across multiple runs and different datasets. This confirms its adaptive capability and its ability to improve with each iteration, making it a suitable choice for real-world applications requiring high accuracy. On the other hand, the RAO-3 algorithm remains effective in smaller networks but encounters significant challenges as the network size and complexity increase. This suggests the need for improvements to enhance its performance in more complex scenarios. Although RAO-3 has potential, its instability in larger networks is a notable weakness that needs to be addressed. Further research on optimization techniques is required to enhance its efficiency for large-scale applications.

Overall, this research not only provides deep insights into the performance of the two algorithms in pathfinding problems but also opens up new directions for optimizing and improving RAO-3 to enhance its ability to handle large-scale data. At the same time, further research on improving the Q-Learning algorithm presents a promising direction for further exploration and enhancement.

## Conflict of Interest

The authors declare no conflict of interest.

## Data Availability Statement

The data supporting the findings of this study are available at "drive.google.com/open?id=1tBgCLMcvP4IGVooU3PQYKEA-Xt0emczK" or can be provided upon reasonable request to the corresponding author.

## REFERENCES

[1] P. C. Tho, T. T. P. Anh, N. T. Cuong, and P. N. Phuong, "Organization and control of urban areas of Vietnam oriented towards green and sustainable development," *UD-JST*, vol. 11, no. 96.2, pp. 183–189, Nov. 2015.
[2] T. Q. Chien and T. N. Tuan, "Algorithms of the problem of finding the shortest paths from a set of nodes to another set of nodes," 2004.
[3] D. N. Lau and N. T. Viet, "Parallelizing algorithm Dijkstra's finding the shortest paths from a vertex to all vertices," *HUEUNI-JNS*, vol. 74B, no. 5, pp. 81–92, 2012.
[4] T. Q. Chien, "Revised Bellman-Ford algorithm finding shortest path on extended networks," *UD-JST*, vol. 11, no. 96.1, pp. 84–87, Nov. 2015.
[5] G. Giorgio and P. Stefano, "Shortest path algorithms," *Ann. Oper. Res.*, vol. 13, no. 1, pp. 1–79, 1988.
[6] E. L. Lawler, "Shortest path and network flow algorithms," *Ann. Discrete Math.*, pp. 251–263, 1979.
[7] D. Eppstein, "Finding the k shortest paths," *SIAM J. Comput.*, vol. 28, no. 2, pp. 652–673, 1998.
[8] M. El Khaili, "Path planning in a dynamic environment," *Int. J. Adv. Comput. Sci. Appl.*, vol. 5, no. 8, pp. 86–92, 2014.
[9] P. T. M. Phuong, P. T. Yen, and N. T. N. Trang, "Research on optimizing order-picking routes in warehouse operation," *JMST*, vol. 80, no. 80, pp. 100–103, Dec. 2024.
[10] W. Zhu, W. Cai, and H. Kong, "Optimal path planning based on ACO in intelligent transportation," *Int. J. Cogn. Comput. Eng.*, 2025.
[11] H. T. Thien, T. V. Le, and S. Bouzefrane, "iTEVAC: An enhanced trusted evacuation system leveraging fog computing and IoT," *J. Inf. Telecommun.*, vol. 8, no. 4, pp. 417–451, 2024.
[12] Y. Li, D. Pan, C. Xing, J. Huang, and Q. Zhang, "Research on road test route design method of ADAS system based on fault database and Dijkstra-CWOA," in *Proc. 7th Int. Conf. Robot., Control Autom. Eng. (RCAE)*, 2024, pp. 281–287.
[13] R. V. Rao, "Rao algorithms: Three metaphor-less simple algorithms for solving optimization problems," *Int. J. Ind. Eng. Comput.*, pp. 107–130, 2020.
[14] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, pp. 279–292, 1992.
[15] T. V. Le, D. L. Le, and H. T. Tran, "Dynamic traffic optimization system: Leveraging IoT and fog computing for enhanced urban mobility with the RAO algorithm," in *Proc. Int. Conf. Green Technol. Sustain. Dev.*, 2024, pp. 73–84.

**Lap Duy Le** graduated from Ho Chi Minh City University of Technology and Education, Vietnam in 2018 with a major in Software Engineering. He is currently pursuing a master's degree in computer engineering at the same university. His research interests include the application of IoT to everyday practical problems.
Email: 2391301@student.hcmute.edu.vn. ORCID: https://orcid.org/0009-0001-9317-7758

**Van Long Nguyen** graduated from Ho Chi Minh City University of Technology and Education, Vietnam in 2000 with a major in Electric Engineering. He is currently pursuing a Master's degree in Computer Engineering at the same university. His research interests incde the application of IoT to everyday practical problems.
Email: longnv@hcmute.edu.vn. ORCID: https://orcid.org/0009-0009-2992-5748

**Thanh Tuan Vu** is currently holding the position of Head of Recruitment at FPT Software Academy. He has 12 years of experience in the fields of Training and Recruitment.
Email: tuanvt5@fpt.com. ORCID: https://orcid.org/0009-0004-5068-871X