REINFORCEMENT LEARNING FOR THE TRAVELING SALESMAN PROBLEM WITH DRONE

Ninh Thi Thanh Tam[1], Nguyen Duc Luu[2], Nguyen Truong Tung[2], Tran Hoang Vu[2], Nguyen Minh Quang[2], Giang Thanh Trung[2], Nguyen Thi My Binh[2,*]

DOI: http://doi.org/ 10.57001/huih5804.2025.356

ABSTRACT

Recently, the problem of transportation routing with integrated drones in IoT-based logistics has emerged as a prominent research area within the academic community. It is particularly important for IoT applications in the transportation and logistics sectors, especially in last-mile delivery. Drones take off from predefined stop points to serve customers and then land at hubs to connect with other vehicles for continued delivery, all while adhering to constraints such as flight range and payload capacity. Integrating drone and vehicle routing poses unique challenges due to the need to synchronize both drone and vehicle schedules to optimize overall delivery time or system costs. In this paper, we introduce the Traveling Salesman Problem with Drone (TSPD) to optimize delivery costs. We then develop two efficient algorithms to address the TSPD. Our Q-learning and Sarsa algorithms enable a drone, transported by a vehicle, to deliver multiple parcels to customers at different locations, with the goal of minimizing delivery costs and maximizing customer satisfaction upon parcel delivery. Through experiments on artificially generated realistic scenarios, the results demonstrate that the proposed algorithms achieve promising computational performance.

Keywords: *IoT, TSPD, SARSA, Q-learning.*

[1]Faculty of Information and Communications Technology, National Academy of Education Management, Vietnam

[2]School of Information and Communications Technology, Hanoi University of Industry, Vietnam

[*]Email: binhntm@haui.edu.vn

Received: 27/5/2025

Revised: 30/6/2025

Accepted: 28/9/2025

1. INTRODUCTION

The Truck and Drone Routing Problem (TSPD) is an extended variant of the classic Traveling Salesman Problem (TSP), which requires synchronization of both ground vehicle and drone routes to achieve optimal delivery cost and time [1]. TSPD has broad applications in logistics, last-mile delivery, robot scheduling, and supply chain planning due to its ability to reduce transportation costs, save fuel, and enhance customer experience.

Theoretically, the TSPD belongs to the class of NP-hard problems and becomes increasingly complex when additional constraints such as drone flight range, payload capacity, and time windows are introduced [2]. Scaling up the problem significantly enlarges the search space, which limits the practicality of exact methods like branch-and-bound or dynamic programming due to resource constraints. Meanwhile, heuristic and metaheuristic algorithms (e.g., genetic algorithms, ant colony optimization) must strike a balance between computational speed and solution quality [3].

The Reinforcement Learning (RL) model offers a novel approach to the TSPD by enabling an agent to autonomously explore and learn optimal policies through interaction with the environment [4]. Instead of exhaustively searching the entire solution space, RL adjusts its strategy based on experience, allowing it to quickly generate near-optimal solutions with lower computational cost. This is especially suitable for real-time scenarios and large-scale or previously unseen data. Applying RL to the TSPD not only provides an effective way to solve the drone-assisted delivery problem but also opens the door for transferring the technology to other complex optimization problems.

2. RELATED WORKS

In recent years, the integration of unmanned aerial vehicles (UAVs), particularly drones, into logistics and delivery systems has garnered increasing attention. Numerous studies have concentrated on optimizing drone and truck-drone collaborative operations. Chung et al. [1] presented a comprehensive review of the state-

of-the-art in combined drone and truck operations, identifying key challenges and outlining future research directions related to routing, scheduling, and system integration. Expanding on this work, Kong and Jiang [2] investigated the truck-drone collaborative routing problem and proposed a delivery optimization method that accounts for vehicle obstacle avoidance - an essential consideration in urban logistics environments. Likewise, Wang et al. [3] examined collaborative route planning for rural logistics, highlighting the benefits of drone-truck coordination in sparsely populated areas.

Focusing on last-mile delivery, Arishi et al. [4] employed machine learning techniques to improve the efficiency of truck-drone systems, particularly within the context of Industry 4.0. Collectively, these studies underscore the growing significance of intelligent algorithms in enhancing drone-assisted logistics operations.

Simultaneously, reinforcement learning (RL) has emerged as a robust framework for solving sequential decision-making and combinatorial optimization problems. The seminal work by Sutton and Barto [5], along with the theoretical advancements in Q-learning by Clifton and Laber [6], has laid a strong foundation for applying RL techniques to routing and delivery optimization. Notably, Bogyrbayeva et al. [7] implemented a deep reinforcement learning approach to tackle the Traveling Salesman Problem with Drone (TSP-D), demonstrating that RL-based methods can effectively generate near-optimal solutions for this NP-hard problem. Their research serves as a critical step toward the development of more scalable and efficient learning-based algorithms for drone-assisted routing.

These contributions collectively emphasize the promising potential of integrating reinforcement learning with drone logistics to address the TSP-D, thereby motivating continued research into more adaptive, robust, and efficient RL-based solutions.

3. PROBLEM STATEMENT

3.1. Preliminaries

Point

- *Definition:* A node represents either a customer or a depot, depicted as a point on a 2D plane with coordinates (x,y).

- *The distance between two points is calculated using the Euclidean distance*:

$$d(A,B) = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2}$$

- *Customer's attributes*:

+ *Demand* - weight of goods to be delivered.

+ *Time satisfaction* - time priority coefficient.

- *Depot's attributes*: located at the origin (0,0); demand = 0 and time satisfaction = 0.

Calculation of Greenhouse Gas Emissions

$$\text{TotalGHG} = \sum_{i=0}^{N+1}\sum_{j=0}^{N+1} d_{ij} * x_{ij} * ET + \sum_{i=0}^{N+1}\sum_{j=0}^{N+1} d_{ij} * y_{ij} * ED \quad (1)$$

where:

+ ET: Greenhouse gas emission factor of the truck.

+ ED: Greenhouse gas emission factor of the drone.

+ d_{ij}: Distance between customers i and j.

+ x_{ij}: Binary variable equals 1 if there is a truck route from customer i to j, and 0 otherwise.

+ y_{ij}: Binary variable equals 1 if there is a drone route from customer i to j, and 0 otherwise.

Increasing the proportion of deliveries made by drones significantly reduces this value due to their low emission factor.

Calculation of customer satisfaction

Delivery time window for the k^{th} customer:

$$\text{DeliveryTime}_k = \frac{d_{k-1,k}}{\text{velocity}_{\text{vehicle}}} + \text{waiting\_time}_{k-1} \quad (2)$$

Customer satisfaction of the k^{th} customer upon delivery:

$$\text{CustomerSatisfaction}_k = \frac{\text{TimeSatisfaction}_k}{\text{DeliveryTime}_k} \quad (3)$$

Total Satisfaction:

$$\text{TotalSatisfaction} = \sum_{k=1}^{N} \text{CustomerSatisfaction}_k \quad (4)$$

Faster delivery leads to higher customer satisfaction, reflecting a better customer experience.

3.2. Problem description

The TSPD problem can be defined as follows: Given a set of customer locations $C = \{c_1, c_2, \ldots, c_n\}$ and a distance matrix D, where d_{ij} denotes the distance between customer c_i and c_j. Each customer c_i has a specified delivery time t_i indicating the latest time the delivery should occur, along with a parcel weight requirement. A coordinated fleet of trucks and drones is used to fulfill these deliveries. After completing each delivery, the drone must return to the truck to recharge and collect the next package. Both the truck and the drone release greenhouse gases during their operations. The goal is to determine a delivery route that ensures all parcels are

delivered through the combined efforts of the truck and drone. The problem can be formally stated as:

- *Input*:

+ A dataset containing a list of points.

+ Indices of transportation vehicles including trucks and drones.

- *Output*:

+ The routes of trucks and drones.

+ The total greenhouse gas emissions and the overall customer satisfaction achieved.

- *Objective:*

+ Minimization of emissions generated:

$$\min \text{TotalGHG} = \sum_{i=0}^{N+1}\sum_{j=0}^{N+1} d_{ij} * x_{ij} * ET + \sum_{i=0}^{N+1}\sum_{j=0}^{N+1} d_{ij} * y_{ij} * ED$$

+ Maximize customer satisfaction:

$$\max \text{TotalSatisfaction} = \sum_{k=1}^{N} \text{CustomerSatisfaction}_k$$

- *Basic Constraints*:

+ Each guest is delivered exactly once by either truck or drone or both.

+ Drones can only deliver goods with the available volume of the drone and within the available range from the current location.

+ The drone must return to the truck to recharge the battery and retrieve the parcel for the next flight.

+ All parcels must be delivered to the customer.

+ Both vehicles must start and end at the depot.

4. PROPOSED METHODOLOGY

In this chapter, we present in detail the solution framework for combining SARSA and Q-learning to solve the TSPD problem, including: SarsaT for truck route optimization, SarsaD for truck-drone coordination and Q-learning with a process similar to SARSA.

4.1. Markov model and basic components

- *Environment: Set customer points C = {0, 1, ..., n}, where 0 is the depot. Each point i has coordinates (x_i,y_i), demand$_i$ and delivery lead time T_i.*

- *Agent:* Duo *of trucks and* drones. Trucks are responsible for carrying heavy goods at a speed of 50 units of distance/h; Light cargo drones with lower speeds and limited range.

- *State (s):*

+ With SarsaT: the current index of the truck.

+ With SarsaD: position pairs (i,j) correspond to trucks at i and drones at j.

- *Action (a):*

+ SarsaT: select unserved k∈C points.

+ SarsaD:

▪ The two vehicles meet.

▪ The truck continues to move independently according to the SarsaT route, not affected by the position of the drone.

▪ Drones serve customers in flight range.

- *Reward (r)*:

$$\text{reward} = \alpha * \text{DeliveryGHG} + \beta * \frac{1}{\text{CustomerSatisfaction}}$$

Where:

+ DeliveryGHG: Emissions generated when transporting goods.

+ $\text{DeliveryTime} = \frac{\text{distance}}{\text{speed}}$.

4.2. SARSA for truck (SarsaT)

- Objective: Find the optimal route for the truck when operating independently.

- State Space: The index of the current point where the truck is (0 for depot, 1, 2, 3, ... for customers).

- Agent: truck.

- Action Space: Indicators of availability points (customers who have not yet delivered).

- Update Q-table:

+ Updated formula for SARSA:

$$Q(s,a) \leftarrow Q(s,a) + \alpha * (\text{reward} + \gamma * Q(s',a') - Q(s,a)) \quad (5)$$

Where:

▪ s: Current status.

▪ a: The next action is selected from s.

▪ s': Next state (after performing action a).

▪ a': The next action is selected from s'.

▪ α: Learning rate.

▪ γ: Discount factor.

- Action Pick Strategy: Use ε-greedy strategy with ε = 0.99:

+99% of the time we will choose the action with the smallest Q value, which corresponds to the overall goal minimization.

+1% choose random actions to explore other possibilities.

- Coaching process:

+Run 2*n episodes, with n being the number of customers.

+In each training episode:

▪ Trucks start from depot.

▪ Repeat until the final state is reached (all items have been delivered to all customers):

1. Select action a (next point) based on ε-greedy.
2. Move to the next point, calculate the reward.
3. Update Q-table.
4. Status Updates s ← s′.

▪ When all customers are delivered, the truck returns to the depot.

+ Save the route with the smallest total target value after 2*n episodes.

- Space for action:

+Drones and trucks meet at one point.

+The truck moves independently to the next point in the existing route without considering the current state of the drone.

+Customer-available coordinates for drone based on hanging configuration characteristics.

- Reward:

+When two vehicles meet:

▪ The drone moves to the truck location or vice versa, depending on the available distance of the drone, giving priority to the drone to the truck's location if available.

▪ If the drone reaches the truck location:

$$\text{reward} = \text{distance}(\text{postition}_{\text{drone}}, \text{position}_{\text{truck}}) * \text{ED}$$

▪ If the truck reaches the drone location:

$$\text{reward} = \text{distance}(\text{postition}_{\text{truck}}, \text{position}_{\text{drone}}) * \text{ET}$$

+When the truck moves independently:

▪ The truck goes to the next stop on the route.

▪ $$\text{reward} = \text{distance}\begin{pmatrix}\text{recentPosition}_{\text{truck}},\\ \text{nextPosition}_{\text{truck}}\end{pmatrix} * \text{ET}$$

Input: α = 0.01, ε = 0.99, *γ = 0.9, customer's properties*

Output: Truck's route, total green house gas, total customer satisfaction

1. Initialize $Q(s, a)$, for all $s \in S^+$, $a \in A(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$
2. Loop for each episode:
3. Initialize S
4. Choose A from S using policy derived from tenNearestLocation and Q(e.g.,ε-greedy)
5. Loop for each step of episode:
6. Choose A' from S' using policy derived from tenNearestLocation and Q (e.g., ε-greedy)
7. Take action A, observe R, S'
8. $Q(S, A) \leftarrow Q(S, A) + \alpha*[R + \gamma*Q(S', A') - Q(S, A)]$
9. $S \leftarrow S'$

Figure 1. SarsaT algorithm pseudo-code

4.3. SARSA for drone (SarsaD)

- Objective: Based on the optimal route of the truck from SarsaT, learn how to integrate drones to improve the target function.

- State Space: A series of representations of the current position of trucks and drones, in the form of "truck position"|" Drone location" (example: "0|0" if both are at the depot, "1|2" if the truck is in customer 1 and the drone is in customer 2).

+When the action is a position within the available range of the drone:

▪ The drone delivers the goods to the selected point, the truck simultaneously moves to the next point.

▪ $$\text{reward} = \text{distance}\begin{pmatrix}\text{recentPosition}_{\text{truck}},\\ \text{nextPosition}_{\text{truck}}\end{pmatrix} * \text{ET} + \text{distance}\begin{pmatrix}\text{recentPostition}_{\text{drone}},\\ \text{nextPosition}_{\text{drone}}\end{pmatrix} * \text{ED}$$

- Q-table Update: Similar to SarsaT.

- Strategy to choose action: similar to SarsaT.

- Coaching process:

+Based on truck route from SarsaT.

+Run 2*n episodes:

▪ Truck and drone start from depot.

▪ Repeat until the final state is reached (all items have been delivered to all customers):

1. Choose an action for a drone from the action space according to the action selection strategy.
2. Update location, GHG, time, satisfaction level.
3. Q-table Updates.
4. Status Updates.

▪ When all the customers are delivered, they both return to the depot.

+Save the route with the smallest target value after 2*n episodes.

▪ s': Next state (after performing action a).

▪ a': The next action is selected from s'.

▪ α: Learning rate.

▪ γ: Discount factor.

From this, we can see that Q-learning algorithms can converge faster in some cases, but may not fully consider the risks.

In summary, the proposed solutions not only meet the requirements of routing optimization but also exploit the advantages of reinforcement learning in the real environment. Next, the experimental evaluation on the dataset and simulation scenario will be presented in detail, from which the performance will be compared, the results will be analyzed, and conclusions will be drawn about the feasibility of the method.

Input: $\alpha = 0.01$, $\varepsilon = 0.99$, $\gamma = 0.9$, *customer's properties, truck's route*
Output: Truck's route, Drone's route, total green house gas, total customer satisfaction
1. Initialize $Q(s, a)$, for all $s \in S^+$, $a \in A(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$, take data from SarsaD, perform getLocationForDrone.
2. Loop for each episode:
3. Initialize S
4. Choose A from S using policy derived from tenNearestLocation and Q(e.g., ε-greedy)
5. Loop for each step of episode:
6. Choose A' from S' using policy derived from tenNearestLocation and Q(e.g., ε-greedy)
7. Take action A, observe R, S'
8. $Q(S, A) \leftarrow Q(S, A) + \alpha*[R + \gamma*Q(S', A') - Q(S, A)]$
9. $S \leftarrow S'$
10. Until S is terminal

Figure 2. SarsaD algorithm pseudo-code

4.4. Q-learning

The Q-learning algorithm is similar to the SARSA algorithm in most respects because SARSA is based on Q-learning, but the difference lies in the Q_table update formula. In the Q-learning algorithm, the Q_table table is updated based on the hypothetical optimal action rather than the actual optimal action as for SARSA. Specifically, Q-learning's Q_table update formula:

Updated Formula for Q-learning:

$$Q(s,a) \leftarrow Q(s,a) + \alpha * (\text{reward} + \gamma * \max Q(s', A) - Q(s,a)) \quad (6)$$

Trong đó:

▪ s: Current status.

▪ a: The next action is selected from s.

5. EXPERIMENTAL

5.1. Description of experimental data

Data for the problem of vehicle routing combined with a drone

Author: Do Thi Ngoc Huyen

Last Updated: 17/12/2024

Data sets used: 50 cities, 100 cities, 200 cities

Values in the dataset:

+Column 1, column 2 (X, Y): coordinates in two-dimensional space.

+Column 3 (Demand): the volume of goods of the customer.

+Column 4 (Time satisfaction): maximum customer delivery lead time.

5.2. Experimental environment settings

All simulations are performed on a computer using an 11th Gen Intel(R) Core(TM) i5-11320H processor @ 3.20GHz (2.50GHz), 8GB RAM, Windows 11 64-bit operating system, using the Python programming language.

To ensure that the evaluation of the model is objective and comprehensive, each model will be trained 20 times independently, each time on a different dataset. These datasets are designed to simulate diverse scenarios, representing a variety of scenarios that can occur in real life. The purpose of this is to test the generalization of the model under a variety of conditions, without being dependent on a fixed data set.

After completing 20 training and assessments, the final result will be calculated by taking the average of all 20 runs. This method minimizes random fluctuations in data and training, which in turn provides a more accurate view of the overall performance of the model.

The evaluation indicators of the model and the parameters of the environment will be presented in detail in the table below. These metrics include both important hyperparameters that directly affect the training and performance of the model, as well as the characteristic elements of the test environment.

Table 1. Model parameters

Parameter	SARSA	Qlearning
Number of customers	50, 100, 200	
Episodes	100, 200, 400	
Learning rate (α)	0.01	
Discount factor (γ)	0.9	
Exploitation - discovery rate (ε)	0.99	

Table 2. Algorithm parameters

Attribute	Truck Parameters	Drone Parameters
Velocity (distance unit/hour)	50	43.2
Capacity (unit of goods)	1500	1
Travel range (Distance Unit)	674.3	14.4
Greenhouse gas emission factor (in GHG units per unit of distance)	Unlimited	4
Mode of operation	Independent	Depend

5.3. Experimental Results

Symbol C_x_Ep_y_Q_z: Number of Clients: x, Number of Training Episodes: y, Initialized QTable Value: z.

Effect of QTable initialization value on results obtained:

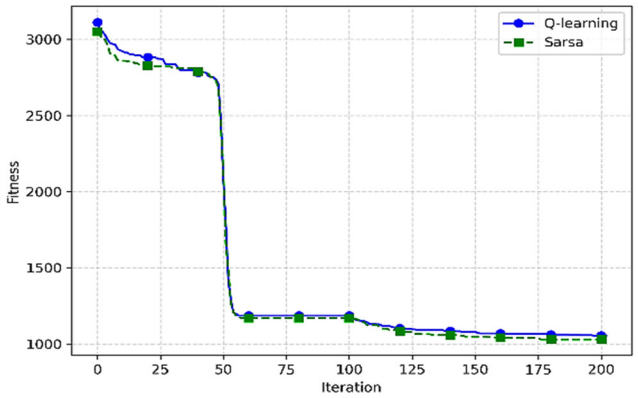
Table 3. Results obtained with C_50_Ep_100_Q_0

Algorithm	Processing time	Total emissions generated	Total customer satisfaction
SARSA	6.26 ± 0.47	102992.40 ± 4413.02	3574.99 ± 608.88
QLearning	6.51 ± 0.66	105572.30 ± 9534.12	3665.59±874.37

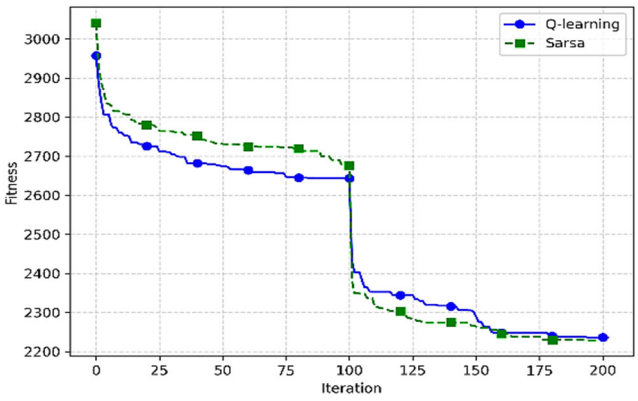
From Table 3, we can see that the SARSA algorithm has a faster processing speed than QLearning because it has a simpler update process due to the nature of on-policy, and is less volatile in learning.

Table 4. Results obtained with C_50_Ep_100_Q_1000

Processing time	Processing time	Total emissions generated	Total customer satisfaction
SARSA	6.14 ± 0.89	222871.66 ± 11466.54	1835.43 ± 608.89
QLearning	6.35 ± 0.65	223278.60 ± 20295.33	1887.57 ± 533.15



a)



b)

Figure 3. Compare the results with Qtable = 0 (a) and Qtable = 1000 (b) with the same number of episodes

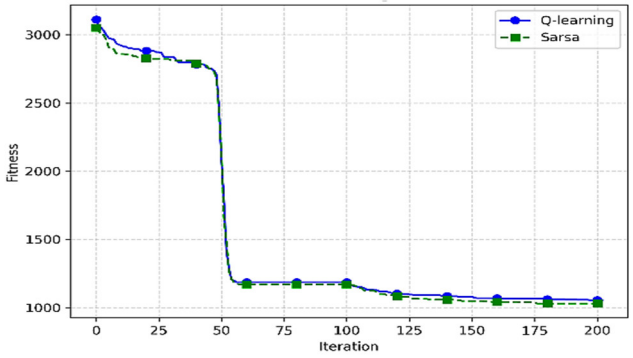
From Figure 3, we can see that, with Qtable = 0 in Figure 3a, both algorithms show that fitness drops from an initial high of 3000 to a low of 1000 after 200 iterations, but Q-learning drops faster and stabilizes at a lower level than Sarsa. With Qtable = 1000 in Figure 3b, the initial fitness is higher at about 3000 and decreases rapidly in the early stages. However, the fatigue in the later stages drops more slowly at around 2200-2300 also with 200 repetitions. With the same number of loops, initializing a Qtable = 0 value consistent with the policy objective used by both algorithms has shown superiority over initializing Qtable = 1000 in the ability to expand the search space to be able to come up with better options in later stages.

Table 5. Results obtained with C_50_Ep_100_Q_0

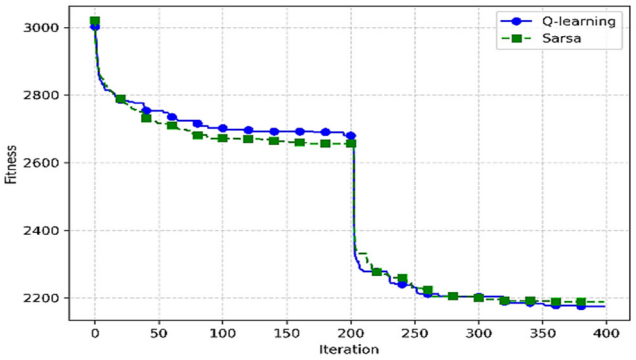
Algorithm	Processing time	Total emissions generated	Total customer satisfaction
SARSA	6.26 ± 0.47	102992.40 ± 4413.02	3574.99 ± 608.88
QLearning	6.51 ± 0.66	105572.30 ± 9534.12	3665.59 ± 874.37

Table 6. Results obtained with C_50_Ep_200_Q_1000

Algorithm	Processing time	Total emissions generated	Total customer satisfaction
SARSA	10.35 ± 0.59	218918.10 ± 4234.25	1897.23 ± 353.73
QLearning	9.78 ± 1.23	217584.49 ± 5108.60	1918.06 ± 582.65



a)



b)

Figure 4. Compare the results with Qtable = 0 (a) and Qtable = 1000 (b) with different number of episodes

From Table 5, Table 6 and Figure 4, we can see that, the processing speed of each algorithm is affected by the size of the input data, as with a set of 50 customers, both algorithms only take an average of 6 seconds, when the data increases to 100 customers, the processing time increases to 10 seconds. With Qtable = 0 in Figure 4a, both algorithms show a sharp drop in fitness after 50 cycles with 200 repetitions, it has dropped from an initial high of 3000 to a low of 1000. With Qtable = 1000 in Figure 4b, the initial fitness is higher than about 3000 and drops deeply in the early stages, but in the later stages, it is stagnant and can only drop below 2100. In short, although the number of loops has increased, the failure to find new possibilities quickly has caused the initialization of Qtable = 100 values to experience slow and high convergence, compared to the initialization of Qtable = 0 values consistent with the policy objectives used by both algorithms that have helped to find new cases in the search space Earn faster and produce better results.

Evaluation of Algorithm Convergence

Table 7. Results obtained with C_50_Ep_100_Q_0

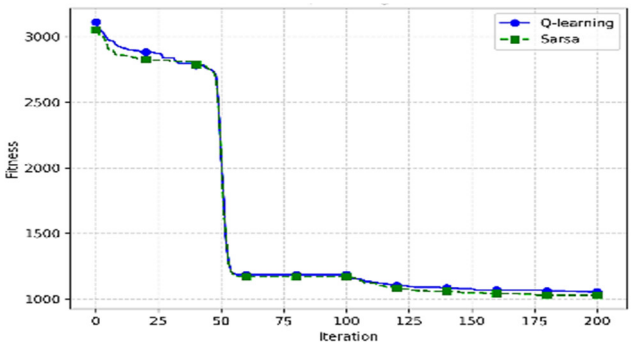
Algorithm	Processing time	Total emissions generated	Total customer satisfaction
SARSA	6.26 ± 0.47	102992.40 ± 4413.02	3574.99 ± 608.88
QLearning	6.51 ± 0.66	105572.30 ± 9534.12	3665.59 ± 874.37

Table 8. Results obtained with C_100_Ep_200_Q_0

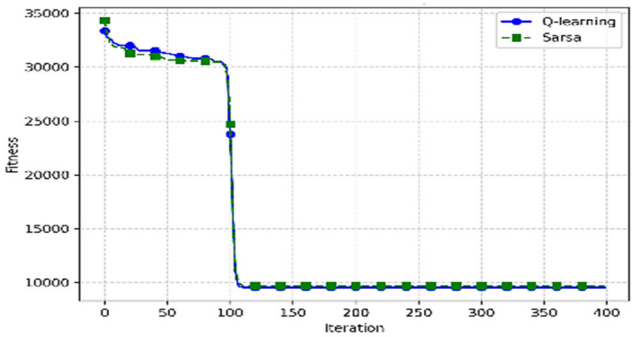
Algorithm	Processing time	Total emissions generated	Total customer satisfaction
SARSA	30.65 ± 4.13	967961.40 ± 42895.95	11617.80 ± 712.91
QLearning	32.00 ± 4.80	954218.60 ± 45682.63	11884.27 ± 1162.70

Table 9. Results obtained with C_100_Ep_400_Q_0

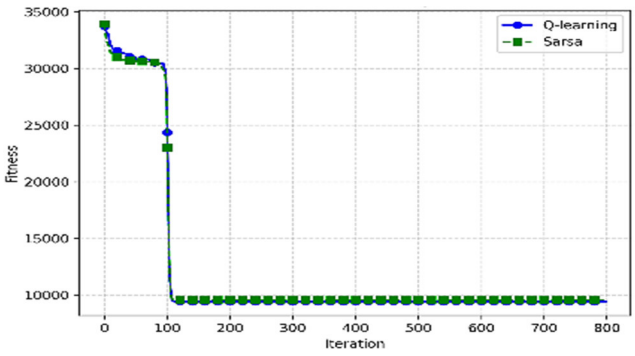
Algorithm	Processing time	Total emissions generated	Total customer satisfaction
SARSA	51.54 ± 3.66	956208.73 ± 43194.74	11944.09 ± 957.2005
QLearning	53.66 ± 2.50	941724.30 ± 55954.40	11497.11 ± 1213.29



a)



b)



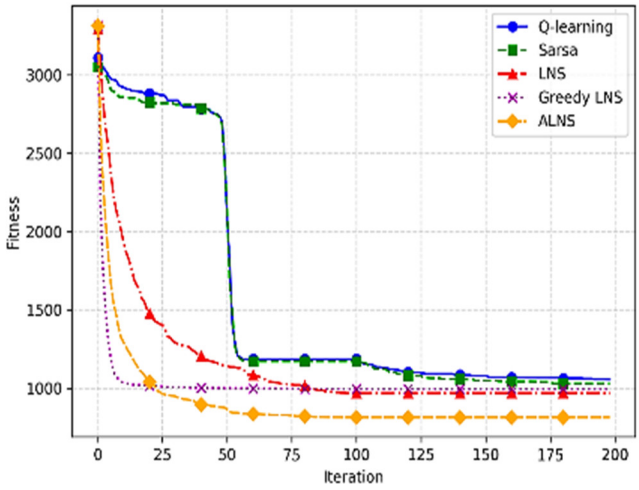
c)

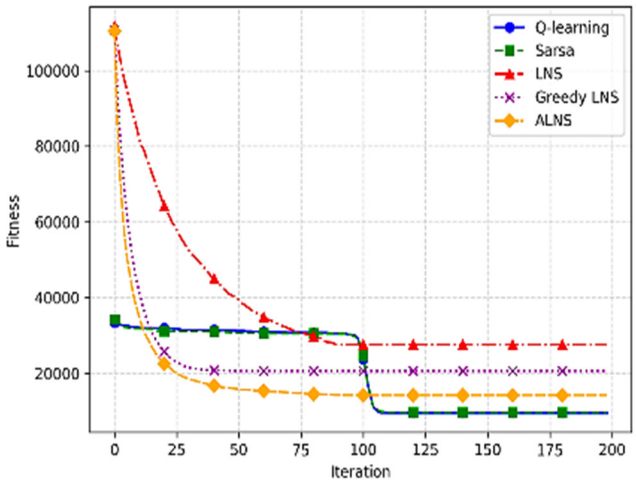
Figure 5. Convergence of algorithms through datasets a: 50, b: 100, c: 200

Based on the results presented in Tables 7, 8, and 9, as well as Figure 5, it is evident that initializing the Q-table with zeros leads to convergence within approximately twice the number of customers in training loops. This is attributed to the rapid and extensive exploration of new cases in the state space.

Comparison of SARSA and Q-learning with existing methods

- Large Neighborhood Search (LNS)
- Greedy Large Neighborhood Search (Greedy LNS)
- Adaptive Large Neighborhood Search (ALNS)





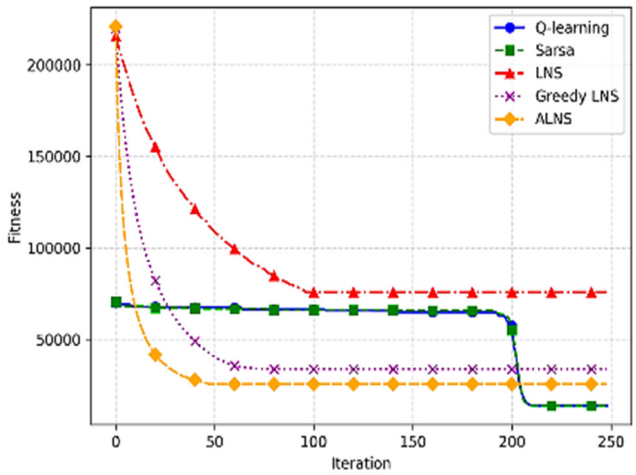


Figure 6. Comparison with 50, 100, 200 city datasets

Table 10. Runtime of local search algorithms on datasets

Dataset / Algorithm	50 cities	100 cities	200 cities
LNS	1.4s	10.81s	365.6s
Greedy LNS	1.8s	10.96s	264.76s
ALNS	0.7s	12.1s	187.36s

Table 11. Results obtained by local search algorithms on datasets

Dataset / Algorithm	50 cities	100 cities	200 cities
LNS	879.92	27605.26	64959.36
Greedy LNS	960.92	20678.4	31659.64
ALNS	764.8	27605.26	22478.92

From the presented figures and tables, it is evident that in small-scale environments, basic local search algorithms such as LNS, Greedy LNS, and ALNS demonstrate significant advantages in terms of convergence speed and solution quality (i.e., lower

fitness values) when compared to reinforcement learning approaches like Q-learning and Sarsa. These heuristic methods efficiently explore the solution space and are capable of rapidly identifying optimal solutions. However, as the problem scale increases and the environment becomes more complex, reinforcement learning models begin to exhibit superior performance. Owing to their dynamic balance between exploration and exploitation, algorithms like Q-learning and Sarsa progressively refine their performance, effectively overcoming the local optima issues that heuristic methods typically face. Consequently, in large and diverse search spaces, reinforcement learning approaches show enhanced robustness and deliver improved solution quality.

In summary, reinforcement learning methods such as Q-learning and Sarsa demonstrate strong potential when addressing large and complex search spaces. While their initial convergence may be slower compared to heuristic approaches, these models progressively enhance both accuracy and stability over successive iterations, ultimately yielding highly optimal solutions for large-scale combinatorial problems.

6. CONCLUSION

This study investigated the use of reinforcement learning to address the Truck and Drone Delivery Problem, aiming to enhance delivery efficiency, reduce operational costs, and improve customer satisfaction. By applying Q-learning and SARSA algorithms, the research demonstrated that reinforcement learning is well-suited to solving complex routing challenges, offering notable advantages over conventional methods. The findings indicate that Q-learning achieves faster convergence and superior performance on larger datasets, whereas SARSA provides more consistent and stable results. Both algorithms yielded near-optimal solutions, with performance deviations remaining within acceptable bounds when compared to heuristic-based local search methods. Future research directions include incorporating real-world constraints such as traffic conditions, drone energy limitations, and dynamic terrain features. Additionally, adopting more advanced reinforcement learning techniques and exploring multi-agent frameworks could further enhance model effectiveness, bridging the gap between theoretical optimization models and their practical applications in logistics

REFERENCES

[1]. Chung, Sung Hoon, Bhawesh Sah, Jinkun Lee, "Optimization for drone and drone-truck combined operations: A review of the state of the art and future directions," *Computers & Operations Research,* 123: 105004, 2020.

[2]. Kong Fanhui, Bin Jiang, "Delivery optimization for collaborative truck-drone routing problem considering vehicle obstacle avoidance," *Computers & Industrial Engineering,* 198: 110659, 2024.

[3]. Wang Yong, et al. "Research on truck-drone collaborative route planning for rural logistics delivery services," *Scientific Reports,* 14.1 (2024): 31815.

[4]. Arishi Ali, Krishna Krishnan, Majed Arishi, "Machine learning approach for truck-drones based last-mile delivery in the era of industry 4.0," *Engineering Applications of Artificial Intelligence*, 116: 105439, 2022.

[5]. Sutton R.S., Barto A.G., *Reinforcement Learning: An Introduction*, 2nd ed. MIT Press, 2018.

[6]. Clifton J., Laber E., "Q-learning: Theory and applications," *Annual Review of Statistics and Its Application*, 7(1), 279-301, 2020.

[7]. Bogyrbayeva A., Yoon T., Ko H., Lim S., Yun H., Kwon C., "A deep reinforcement learning approach for solving the traveling salesman problem with drone," *Transportation Research Part C: Emerging Technologies*, 148, 103981, 2023.