A Q-Learning based Routing Algorithm for Internet of Vehicles, using vehicle trajectory prediction

Pratik Tripathy, Praneeth Chandra, Nuthan Chingeetham, Dr. Prasenjit Chanak
Department of Computer Science and Engineering, IIT (BHU) Varanasi

***Abstract*—In the rapidly evolving landscape of the Internet of Vehicles (IoV), where vehicular networks are becoming increasingly interconnected and intelligent, novel routing protocols are essential to ensure efficient and reliable communication. The protocol employs Q-learning to select the optimal neighboring vehicle for data packet forwarding, considering parameters such as stability, continuity, and cache. Nodes broadcast INFO packets to share neighbor characteristics, enabling informed decision-making. Each vehicle queries its Q-table to choose the best neighbor for forwarding. Predicted distances are incorporated to calculate final discounted Q values, selecting the neighbor with the maximum Q value using a Greedy approach. Vehicle trajectory prediction using the ARIMA time series model helps determine vehicle location at future time steps, aiding in estimating the distance between current and neighboring nodes when the data packet is expected to arrive. The suggested approach performs better than the other routing algorithms, according to simulation findings, in terms of packet delivery ratio, average end-to-end delay, and routing overhead ratio.**

***Index Terms*— Routing, IoV, Trajectory Prediction, Q-Learning, ARIMA model**

I. Introduction

THE Internet of Vehicles (IoV) represents a revolutionary paradigm shift in transportation, fostering intelligent traffic management, enhanced safety features, and improved driver experiences. IoV permits the real-time exchange of information regarding accidents, hazards, and road closures, allowing vehicles to respond promptly and avoid collisions. Traffic management systems can employ IoV data to optimize traffic flow, reduce congestion, and improve travel time. Furthermore, IoV paves the way for cooperative driving applications where cars share information on their position and purpose, facilitating smoother traffic flow and possibly expediting the development of autonomous vehicles.[1]

IoV is a dynamic communication ecosystem that allows cars to communicate data with their surroundings. The system is based on vehicle-to-vehicle communication, resulting in a dynamic ad-hoc network where cars share information and collaborate for diverse purposes. However, guaranteeing effective and dependable data routing in such a dynamic environment poses major hurdles. This ecosystem encompasses various communication modes, including vehicle-to-vehicle (V2V), vehicle-to-person (V2P), and vehicle-to-cloud (V2C) communication, vehicle-to-infrastructure (V2I),. Collectively, these communication forms are referred to as Vehicle-to-Everything (V2X).

Improved routing protocols in IoV might transform India's traffic system, which is overcrowded and has high accident

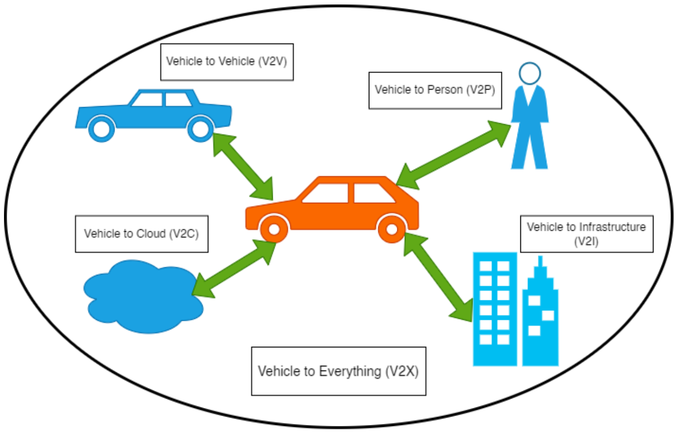

Fig. 1. Various modes of communication in IoV.

rates. A better routing system might address this issue by dynamically selecting the optimum routes based on real-time characteristics such as traffic, road closures, and vehicle behavior. Improved information sharing can lead to smoother traffic flow, shorter travel times, and significantly lower fuel consumption and emissions in Indian cities.

Unlike static networks, routing in IoV is a complex task owing to the network's dynamic nature. IoV uses a variety of routing methods to manage the constantly changing terrain formed by moving cars. These protocols are classified depending on the information they require: map-based routing, geographic location-based routing, position-based routing, path-based routing, and topology-based routing. Map-based protocols utilize preloaded digital maps containing information like traffic flow and road closures to guide data routing. Geographic location-based and position-based protocols utilize the vehicle's present location to identify the next hop for data packets. Path-based protocols establish and maintain specific paths between vehicles based on historical data or real-time information. Topology-based protocols require knowledge of the entire network's structure, which is impractical in IoV due to its constantly changing topology.

The main contributions of this paper are summarized as follows:

1) The nodes that receive the INFO packets sent by the current node are considered to be its neighbours. An optimal route from the source to desired destination nodes is determined by sending a data packet from source node, selecting the intermediate (neighbouring) nodes for forwarding using the proposed Q-learning model until it

reaches the destination node

2) The stability factor,the continuity factor and the resource factor of all the neighboring nodes is calculated so as to determine the reward value of the Q-learning model for forwarding the data packet to that node. The reward table with the corresponding reward values is filled.
3) The Bellman equation is used to train the Q learning model to get the corresponding Q values which is use to fill the Q table.
4) The ARIMA time series model is used to predict the future location of a particular vehicle at the time instant when the data packet is assumed to arrive at the next hop node
5) The Distance factor is calculated to direct the data packet toward the destination node by using the distances calculated from predicted locations.
6) The Enhanced Q values is found integrating the initial Q values with the distance factor which is then used to select the next hop node greedily.

II. Literature review

The dynamic nature of Internet of Vehicles (IoV), a key component of Intelligent Transportation Systems (ITS), necessitates efficient routing protocols to improve road safety, traffic flow, and driver experience. Geographic routing protocols, like Greedy Perimeter Stateless Routing (GPSR), are popular for their simplicity and effectiveness in IoV. GPSR uses vehicle GPS data to forward packets to their destination. However, its usefulness decreases in sparse networks with weak connectivity. This protocol is popular for its simplicity and effectiveness but suffers from reduced usefulness in sparse networks with weak connectivity. Its reliance on vehicle GPS data for packet forwarding limits its performance in such scenarios.

Fadil *et al.* explored the emergence of the Internet of Vehicles (IoV) as an upgraded version of VANET, with a focus on the problems and prospects for urban implementation. It investigated the technological constraints of IoV, showed known solutions, and proposed future approaches based on reviewed research papers. [2]

ECRDP, or Efficient Clustering Routing for Urban VANETs using Density Peaks Clustering (DPC) and Particle Swarm Optimization (PSO), was suggested by Khalid *et al.*. In a simulated urban setting, ECRDP enhanced routing performance by improving cluster head selection and clustering based on link dependability. This resulted in higher levels of stability, throughput within and between clusters, and a decrease in average latency. While ECRDP improves routing performance by enhancing cluster head selection and clustering based on link dependability, its effectiveness may be limited in highly dynamic urban environments where clusters frequently change [3].

Jafarzadeh *et al.* proposed a multi-path transmission method for video streaming in VANETs that employs an adaptive junction-based routing protocol with QoS support. It introduced algorithms based on ant colony optimization and fuzzy logic to optimize routing decisions, as well as TCP and UDP protocol adaptations for better video stream performance. [4]

Karbab *et al.* suggested using Multi-Agent Reinforcement Learning (MARL) for routing optimization in VANETs, which addressed issues such as node velocity and varying density. It also included a Fuzzy Logic system for evaluating link quality, which results in improvements in routing metrics such as delivery ratio, end-to-end delay, and traffic overhead.[5]

Wang *et al.* offered a geographic routing method based on vehicle location analysis, which took into account implicit factors influencing vehicle movement. It surpassed previous methods in terms of packet delivery ratio, average end-to-end delay, and routing overhead ratio in IoV and ITS simulations.[6]

Karim *et al.* discussed the obstacles and potential associated with integrating Internet of Vehicle (IoV) technologies in metropolitan areas. It explored the technical limits and potential solutions for IoV, an improved version of Vehicular Ad hoc Network (VANET) that is intended for safe driving. [7]

Wang *et al.* examined the security and privacy issues in Internet of Vehicles (IoV) networks, emphasizing the importance of robust communication protocols and solutions. It provides latest advances and solutions, including the usage of blockchain for enhanced security, in a thorough and understandable manner. [8]

Shon *et al.* focused on advancements in in-vehicle networking, autonomous network security, and safety in the context of IoT and cars. It addressed the issues posed by cybersecurity threats as well as the requirement for safe and dependable automotive systems. [9]

Ajaz *et al.* examined the evolution of routing protocols from VANET to the Internet of Vehicles (IoV), categorizing them according to parameters and analyzing difficulties and future prospects. Its goal was to guide and motivate IoV researchers in designing effective routing methods. [10]

Finally, routing protocols are critical for ensuring reliable and efficient communication between vehicles in IoV. Geographic, position-based, and Q-learning-based routing protocols offer various approaches to address routing challenges. Integrating vehicle trajectory prediction with routing algorithms further improves IoV performance by optimizing route selection based on predicted vehicle movements.

III. Proposed Methodology

A Q-learning is proposed based routing algorithm integrated with vehicle trajectory prediction by analysing the neighbouring nodes and finding which one of them would be best for the data forwarding. The vehicle trajectory is predicted using ARIMA time series model.

A. System Model

The system model for the routing algorithm is defined as follows.

1) Grid-based Map Representation: The network area is modeled as a two-dimensional grid with each cell representing a unit of space. The overall dimensions of the grid map (e.g., number of rows and columns) should be chosen based on the expected number of vehicles and the desired simulation area.

Each cell in the grid can be assigned a state value indicating whether it is occupied by a vehicle or is empty space.

2) Vehicle Description: Vehicles within the network act as agents responsible for carrying and forwarding data packets. Each vehicle has a unique identifier (ID) to distinguish it from others in the network. The vehicle's location within the grid map is represented by its cell coordinates (row and column indices). A speed value is assigned to each vehicle. This speed can be a fixed value or incorporate a random ”jitter” factor to mimic real-world variations in vehicle movement. A Communication range is defined to determines the maximum distance within which a vehicle can establish communication with neighboring vehicles. The topology can be imagined as in Figure 2.

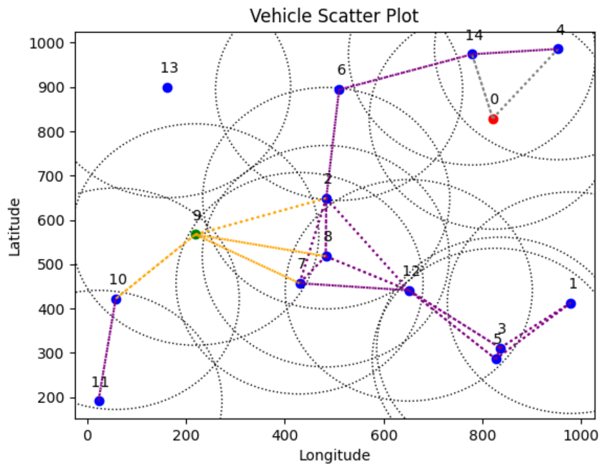

Fig. 2. Topology of 15 vehicles in which, 0 is source and 9 is destination.

3) Additional Description: Additionally, the model incorporates various limitations of a vehicle, such as

- Number of Neighbors: It the count of nearby vehicles within the communication range that can potentially be selected as forwarding nodes for data packets or for collaborative actions such as cooperative sensing or merging in traffic. This can result in prioritizing vehicles with sufficient number of neighbors to forward packet.
- Resource Utilisation: A resource limit is implemented to represent the onboard storage capacity for holding data packets. This can influence routing decisions, prioritizing vehicles with sufficient space to temporarily store packets before forwarding.

B. Packet Delivery Process

The packet delivery process within this methodology is the core decision-making mechanism for routing data packets in the dynamic vehicle network.

1) Packet Definition and Initialization: When a vehicle needs to send a data packet to a destination, it creates a new packet that contains the information like Source (The ID of the vehicle originating the packet), Destination (The ID of the intended recipient vehicle), Current Location (The current cell coordinates of the sending vehicle), Time Taken (A timer variable initialized to 0, which will track the total time elapsed during transmission) and Previous Hop (This will be used to keep track of the previous forwarding vehicle in the packet's journey, initially set to None).

2) Neighbor Discovery and Information Exchange: The sending vehicle identifies its current set of neighboring vehicles. This can be achieved by maintaining a communication range and considering all vehicles within that radius. Each vehicle in the network periodically broadcasts INFO packets, to it's neighbors, containing crucial information for routing decisions. These INFO packets includes Vehicle ID (Unique identifier of the broadcasting vehicle), Location (Current cell coordinates of the broadcasting vehicle), Stability Factor, Continuity factor, Resource Utilisation, Number of Neighbors, etc.

Algorithm 1 Routing Protocol

```
Require: $T, Dataset, packet, env$
Ensure: $packet.at == env.destination$
  $t \Leftarrow 0$
  while $t < T$ do
    $vehicle \Leftarrow dataset[t]$
    if $packet.at == env.destination$ then
      $break$
    end if
    $UpdateVehiclesLocation(env, vehicles)$
    $UpdateRewardTable(env, vehicles)$
    $Q \Leftarrow QLearning(env, src, dest)$
    $df \Leftarrow ARIMAGetDf(env, src, dest)$
    $Q' \Leftarrow sigmoid(Q) * sigmoid(df)$
    $packet.at \Leftarrow argmax(Q'[packet.at])$
    $t \Leftarrow t + 1$
  end while=0
```

C. Q-Learning for Action Selection

The vehicle sending the data packet utilizes the Q-learning algorithm to make an informed decision about the ”best” neighboring node to forward the packet. Q-learning is a reinforcement learning technique where the agent (vehicle) learns through trial and error by interacting with its environment (network) and receiving rewards. The core of Q-learning is the Q-table, which stores the estimated Q-value for each possible action in a given state. The Q-value represents the expected future reward associated with taking that action. In this context, the state is the vehicle or node that has the data packet. The action space consists of the set of neighboring vehicles to which the data packets can be forwarded to.

Algorithm 2 ARIMA Get DF Table

```
**Require:** $env, src, destination, vehicles, numEpisodes$
**Ensure:** $DFTable$ is returned
  $env.vehicles \Leftarrow ARIMA(1, 1, 0).forecast(vehicles)$
  while cur in vehicles do
    while nei in cur.neighbor do
      $df_table[cur][nei] \Leftarrow angle(cur, nei, destination) * exp(-dist(nei, destination)/env.Range)$
    end while
  end while=0
```

The States of Q-Learning are the vehicles $V_1, V_2, ...V_n$ and Actions are also the vehicles $V_1, V_2, ...V_n$. Hence the Q-table is an $n * n$ table. A Reward table is created which gives the reward for each pair of State and Action. Hence the Reward table is an $n * n$ table. Initially, the Q-table is filled with zeros. The epsilon-greedy Q-learning model is used involving adding an exploration-exploitation strategy to the basic Q-learning algorithm. The epsilon value determines the balance between exploration (taking random actions to discover new states) and exploitation (selecting the best-known action based on the current Q-values). During training, at each time step, the agent (in this case, a vehicle) chooses a random number between 0 and 1.In the event that this value is smaller than epsilon, the agent investigates by acting arbitrarily. If not, it chooses the action for the current state that has the highest Q-value in order to exploit.

Algorithm 3 Q Learning

```
**Require:** $env, src, destination, numEpisodes$
**Ensure:** $QTable$ is returned
  $episode \Leftarrow 0$
  $Q \Leftarrow 0$
  $alpha \Leftarrow getAlpha()$
  $epsilon \Leftarrow getEpsilon()$
  $gamma \Leftarrow getGamma()$
  while episode ¡ numEpisodes do
    $state \Leftarrow env.reset$
    $done \Leftarrow false$
    while not done do
      if $random() < epsilon$ then
        $action \Leftarrow env.getRandomAction()$
      else
        $action \Leftarrow argmax(Q[state])$
      end if
      $nextState, reward, done \Leftarrow env.step(action)$
      $Q[state][action] \Leftarrow Q[state][action] + alpha * (reward + gamma * max(Q[next_state]) - Q[state][action])$
      $state \Leftarrow nextState$
    end while
    $episode \Leftarrow episode + 1$
  end while=0
```

Based on the observed rewards and the Q-values in the table, the agent updates the Q-values using the Bellman Equation.

$$Q(s, a) = (1 - \alpha) * Q(s, a) + \alpha * (r + \gamma * \max_{a'} Q(s', a')) \tag{1}$$

where:
$Q(s, a)$ is the Q-value for state s and action a,
α is the learning rate,
r is the immediate reward,
γ is the discount factor,
s' is the next state,
a' is the next action,
$\max_{a'} Q(s', a')$ represents the maximum Q-value achievable in the next state s'

Algorithm 4 Q Learning

```
**Require:** $env, src, destination, numEpisodes$
**Ensure:** $QTable$ is returned
  $episode \Leftarrow 0$
  $Q \Leftarrow 0$
  $alpha \Leftarrow getAlpha()$
  $epsilon \Leftarrow getEpsilon()$
  $gamma \Leftarrow getGamma()$
  while episode ¡ numEpisodes do
    $state \Leftarrow env.reset$
    $done \Leftarrow false$
    while not done do
      if $random() < epsilon$ then
        $action \Leftarrow env.getRandomAction()$
      else
        $action \Leftarrow argmax(Q[state])$
      end if
      $nextState, reward, done \Leftarrow env.step(action)$
      $Q[state][action] \Leftarrow Q[state][action] + alpha * (reward + gamma * max(Q[next_state]) - Q[state][action])$
      $state \Leftarrow nextState$
    end while
    $episode \Leftarrow episode + 1$
  end while=0
```

To select the next hop, the sending vehicle queries the Q-table for Q-values corresponding to each neighboring node in the current state.

D. Reward Calculation

The Reward table used in the Q-learning algorithm is created which is used for making routing decisions. Whenever a vehicle receives an INFO packets from it's neighbour it updates it's Reward table based on the newly received information along with the previous information. Reward table stores the reward value associated with sending a data packet to a specific neighboring vehicle. The reward should capture the desirability of selecting a particular neighbor for forwarding based on various factors, each having a value from 0 (bad) to 1 (good):

- Stability factor: It is defined as the change of the relative distance between sender node and adjacent node. This metric reflects the reliability of the connection between the sender and the neighbor.Different node survival periods are caused by their relative stability, and neighbors with short link survival times might quickly cause an inter-node link to fail.

$$c = |dist_t(cur, nei) - dist_{t-1}(cur, nei)| \leq speed_{avg} \tag{2}$$

$$sf_t = \begin{cases} 1 - \frac{|dist_t(cur,nei) - dist_{t-1}(cur,nei)|}{speed_{avg}} & \text{c} \\ 0 & \text{not c} \end{cases} \tag{3}$$

$$sf = a_1 * sf_t + (1 - a_1) * sf_{t-1} \tag{4}$$

- Continuity factor: It is the relative number of neighbors of the current node. This factor assesses the neighbor's ability to further forward the packet towards the destination. A node with few or no neighbors may be chosen at random by a stable node, which could result in routing failure. More neighbors mean a node is more connected and increases the likelihood that packets will be sent to a next-hop that is nearer the destination node.

$$cf_t = \frac{neighbors(nei)}{neighbors_{max}} \tag{5}$$

$$cf = a_2 * cf_t + (1 - a_2) * cf_{t-1} \tag{6}$$

- Resource Factor: It is the resources utilisation of a vehicle. A lower resource utilisation value indicates the neighbor has space to buffer the data. A node with lower resource utilisation is preferred.

$$rf_t = 1 - resourceUtil(nei) \tag{7}$$

$$rf = a_3 * rf_t + (1 - a_3) * rf_{t-1} \tag{8}$$

Based on these three factors with calculate the reward for a particular vehicle forwarding to a particular neighbour as a linear combination.

$$reward = b_1 * sf + b_2 * cf + b_3 * rf \tag{9}$$

Algorithm 5 Update Reward Table

```
Require: $env, vehicles$
Ensure: $env.rewardTable$ is updated
  while cur in vehicles do
    while nei in cur.neighbors do
      $sf \Leftarrow getSF()$
      $cf \Leftarrow getCF()$
      $rf \Leftarrow getRF()$
      $reward \Leftarrow b1 * sf + b2 * cf + b3 * rf$
      $env.rewardtable[cur][nei] \Leftarrow reward$
    end while
  end while=0
```

E. Vehicle trajectory prediction with ARIMA

Since the network is dynamic with moving vehicles, relying solely on the current network state might not be optimal for routing decisions. To address this, the methodology incorporates ARIMA (Autoregressive Integrated Moving Average) for trajectory prediction. The sending vehicle utilizes the ARIMA model to predict the future location of each neighboring node at the time the packet would likely reach them. This prediction considers historical movement patterns of the neighboring vehicles. By predicting future locations, the sending vehicle can estimate the potential future state of the network after the packet is forwarded.

The chosen ARIMA model, ARIMA (1,1,0), is specifically selected for its suitability in predicting short-term trends. In a dynamic vehicle network, this translates to effectively forecasting the movement patterns of neighboring vehicles for a limited time horizon (roughly the time it takes for a packet to reach them). Also the order is kept low so as to reduce the training time.

- Autoregressive (AR) Term (Order 1): This component analyzes the historical positions of a vehicle. It considers the most recent change in the vehicle's location (difference between its current and immediate past position) to predict the next change. This captures the vehicle's ongoing movement direction and speed.
- Integrated Term (Order 1): The data used for prediction might exhibit non-stationarity, meaning its statistical properties (like mean or variance) might change over time. The first differencing step removes any trends in the data, making it stationary and suitable for ARIMA modeling.
- Moving Average (MA) Term (Order 0): Since we're focusing on short-term predictions, ARIMA (1, 1, 0) doesn't incorporate a moving average component. This simplifies the model and emphasizes the immediate past movement patterns for the prediction.

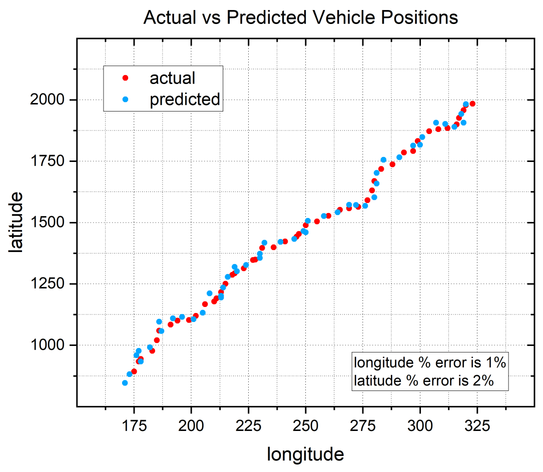

Fig. 3. Vehicle actual and predicted positions.

By combining these elements, ARIMA (1, 1, 0) effectively forecasts the future locations of neighboring vehicles, enabling the sending vehicle to make informed routing decisions based on their predicted positions and the potential future network state. The final predictions is done on both the latitude and longitude of a vehicle separately. The results of prediction for a car moving is given by Figure 3.

$$\Delta X_t = X_t - X_{t-1} \tag{10}$$

$$\Delta X_t = \alpha_1 \Delta X_{t-1} + \epsilon_t \tag{11}$$

and

$$\Delta Y_t = Y_t - Y_{t-1} \tag{12}$$

$$\Delta Y_t = \alpha_1 \Delta Y_{t-1} + \epsilon_t \tag{13}$$

Distance factor table: This table stores the distance factor between any two vehicles in the grid map. This information is crucial for determining the neighbour to send vehicle to. The distance table is a two-dimensional array similar to the Q table, where each cell represents a value which determines which neighbor to send the data packet, combining both the closeness of the neighbour to destination and direction of neighbour to the destination. It utilizes the future position of vehicles calculated above by the ARIMA model.

$$df_t = \cos(\theta) * e^{-\frac{dist(nei,des)}{R}} \tag{14}$$

where θ = the angle made by the neighboring node between current node and destination node

F. Enhanced Q value

Both the information regarding the quality of nodes (given by Q table) and location information of the vehicle (given by distance factor table) is combined

$$Q' = \text{sigmoid}(Q) * \text{sigmoid}(df_{\text{table}}) \tag{15}$$

1) Selecting the Next Hop based on Enhanced Q-Values: Both the information received from Q values (node quality) and distance factor according to the formula is combined. The sending vehicle then selects the neighboring node with the highest Enhanced Q-value as the next hop for packet forwarding. This approach prioritizes the path with the highest potential reward for successful delivery.

$$\text{next\_hop\_node} = \text{argmax}(Q'[i]) \tag{16}$$

2) Packet Transmission and Timer Update: Once the next hop is chosen, the sending vehicle transmits the data packet to the neighboring vehicle.The packet's timer is updated based on the distance to the next hop (using the precomputed distance table) and the transmission speed of the network. Once the data packet reaches the destination the transmission is over.

Algorithm 6 Update Vehicles Location

```
Require: $env, vehicles$
Ensure: $env.location$ is updated
  while cur in vehicles do
    $env.location[cur.id] \Leftarrow cur.location$
  end while=0
```

IV. Simulation Setup

The proposed algorithm is implemented using Python 3 on an Intel Core-i7 12700H processor, and evaluated its performance against ICAR, RPUV, and RAVP. The simulation area was 2000m x 2000m, with vehicles equipped with 802.11p wireless radios. The various parameters used are given in Table 1. Any packets reaching the destination after 1 minute is considered as expired. The following metrics are used in the performance comparison[11]:

- Packet Delivery Ratio: The ratio of the number of packets delivered by the source node within a specific time frame to the total number of packets successfully received by all destination nodes.
- Average End-to-End Delay: The sum of the transmission, processing, and propagation delays for each hop node over the pertinent link is the average end-to-end delay.
- Routing Overhead Ratio: The total number of packets that were forwarded, less the number of packets that were successfully received by the destination, and divided by the number of packets successfully received by the destination.

TABLE I
Simulation Parameters

Description	values
Neighbour distance	250m
Number of vehicles	100, 150, 200 ... 450
Maximum time	0.5, 1, 1.5 ... 4.0 day
Grid size	2000m x 2000m
Vehicle Average speed	10m/s
ARIMA Look back	5
Packet acceptable time	1min
Packet size	512kb
Transmission rate	2Mbps
Maximum neighbors	8
Q-learning alpha	0.9
Q-learning gamma	0.95
Q-learning epsilon	0.0
ARIMA Look back	5
a1, a2, a3	0.8, 0.8, 0.8
b1, b2, b3	0.2, 0.5, 0.2

V. Results

The following results are obtained for the three parameters.

A. Packet Delivery Ratio

As simulation time increased, the delivery ratio of four algorithms decreased gradually. This decline was attributed to the accumulation of discarded data packets, which increased network load and congestion over time. Despite this trend, QLVP exhibited a superior packet delivery ratio compared to other algorithms. QLVP's strength lies in its ability to determine the best neighbor along the direction of the destination node, enhancing its robustness and ultimately leading to better performance in maintaining high delivery ratios over time, as shown in Figure 4.

As the number of vehicles increases, the continuity between vehicles improves, enhancing network connections and increasing information transmission efficiency. However, beyond

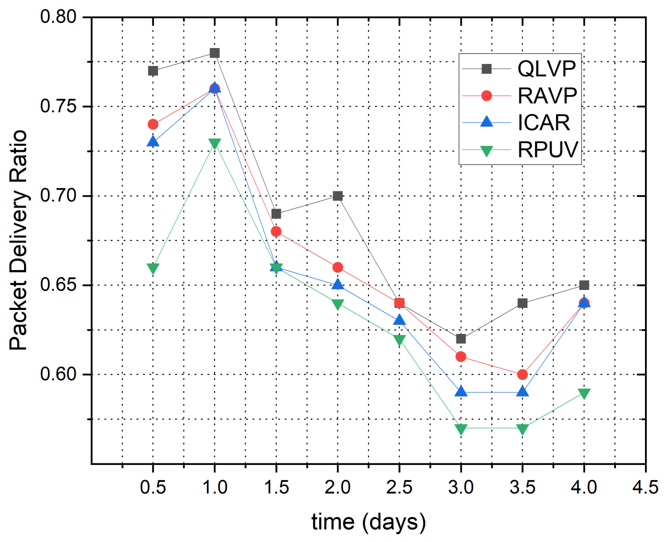

Fig. 4. Result of Packet Delivery Ratio for different Time(day).

a certain threshold, the delivery ratio gradually decreases due to increased congestion. QLVP outperforms other algorithms in this scenario by considering neighbor information and prioritizing better quality neighbors. This approach ensures that QLVP adapts to the increasing number of vehicles, maintaining a higher delivery ratio compared to other algorithms as the network expands, as shown in Figure 5.

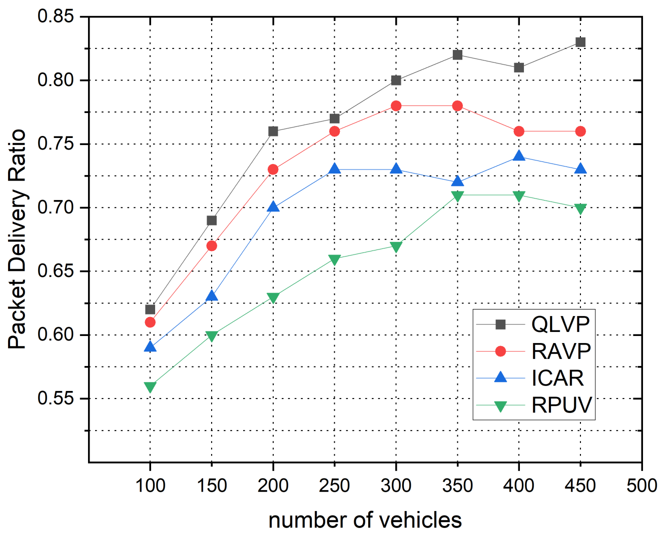

Fig. 5. Result of Packet Delivery Ratio for different Number of Vehicles.

B. Average End-to-End Delay

As simulation time increases, the number of nodes and network load also increase, leading to higher delays in all the algorithms. Initially, QLVP performs better, likely due to its efficient neighbor-based approach. However, as time progresses, QLVP's delay gradually increases. This is attributed to the algorithm's increased computations and jumps as the network expands and becomes more complex. Despite its initial advantage, QLVP's performance diminishes over time compared to other algorithms, as shown in Figure 6.

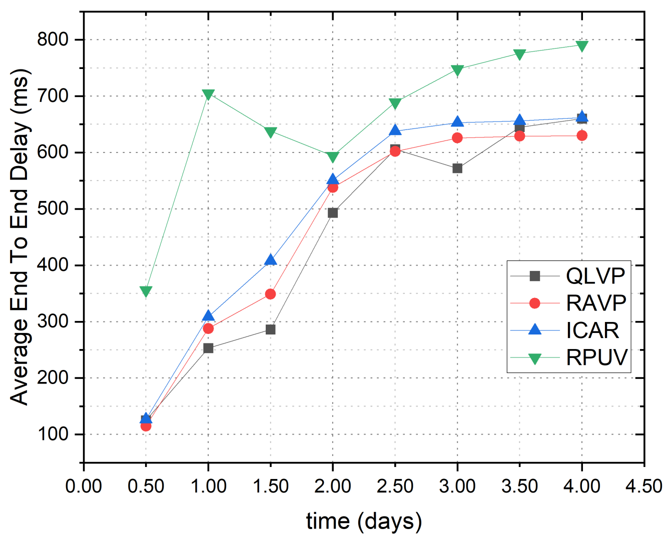

Fig. 6. Result of Average End-to-End Delay for different Time(day).

The average end-to-end delay of each algorithm decreases gradually as the number of vehicles increases. This is because as the vehicle density increases, the distribution becomes more intensive, leading to stronger network connectivity and accelerated data packet transmission speeds. Consequently, delays are reduced due to improved network conditions. However, QLVP experiences a slight slowdown as its computation time is directly influenced by the number of vehicles. Despite this, the overall trend shows that increasing vehicle numbers contribute to reduced delays in all algorithms, as shown in Figure 7.

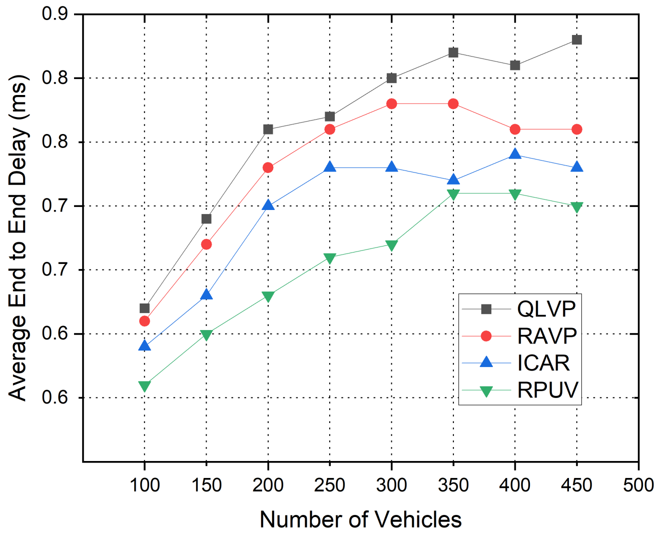

Fig. 7. Result of Average End-to-End Delay for different Number of Vehicles.

C. Routing Overhead Ratio

With an increase in simulation time, the routing overhead typically decreases. In the case of QLVP, the routing overhead remains consistently low. This is because vehicles in QLVP only send INFO packets to their immediate neighbors periodically, regardless of the simulation time. This approach ensures that the routing overhead in QLVP is not significantly affected by the duration of the simulation, contributing to its efficient performance over time, as shown in Figure 8.

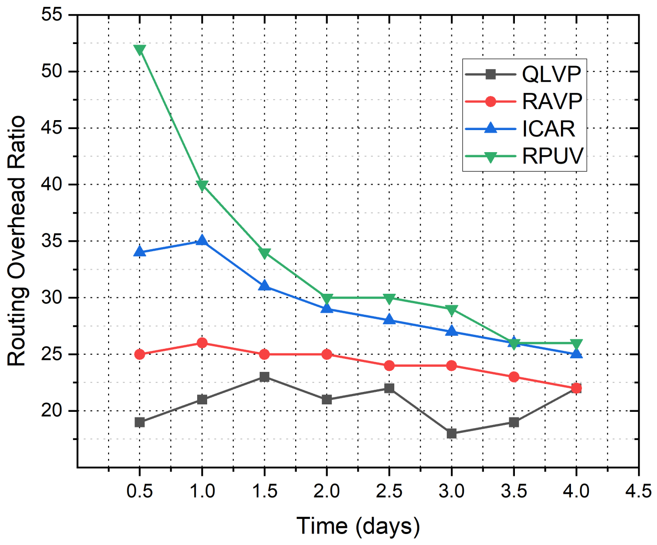

Fig. 8. Result of Routing Overhead Ratio for different Time(day).

As the number of vehicles increases, the routing overhead gradually increases due to more vehicles participating in message passing and route propagation. This increase in vehicle density also leads to a higher frequency of route propagation failures, further contributing to overhead. In contrast, QLVP maintains lower overhead by broadcasting INFO packets only to neighboring vehicles, rather than to all vehicles. This targeted approach reduces the number of vehicles involved in message passing, minimizing congestion and ensuring more efficient routing. Consequently, QLVP outperforms other algorithms by effectively managing overhead and maintaining better pathfinding performance, even as the number of vehicles increases, as shown in Figure 9.

VI. Conclusion

This algorithm for IoV routing integrates Q Learning and ARIMA forecasting. It assesses neighbor quality with Q Learning and predicts vehicle trajectories using ARIMA. By merging these approaches, it selects the optimal neighbor for packet forwarding, preempting congestion. This integration enhances routing decisions, decreasing packet loss, boosting efficiency, and optimizing resource allocation in IoV networks. Through experimentation and comparison with existing algorithms, the effectiveness of the proposed method in reducing travel time and improving overall network efficiency is proved.

For future work, the plan is to focus on improving the accuracy of ARIMA forecasting by optimizing the model parameters and exploring other forecasting techniques. Moreover, the aim is to enhance the algorithm's processing time to enable real-time routing decisions. Obtaining real-time, high-quality datasets will also be crucial for validating and refining the algorithm in practical IoV environments.

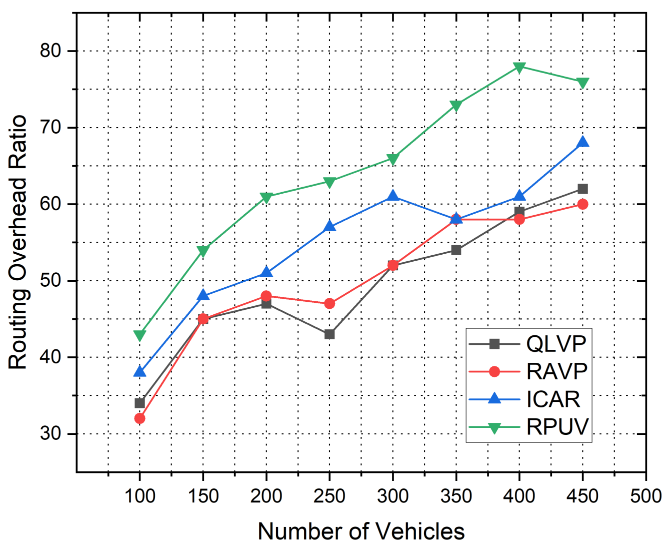

Fig. 9. Result of Routing Overhead for different Number of Vehicles.

References

[1] C. Chen, L. Liu, T. Qiu, J. Jiang, Q. Pei, and H. Song, "Routing with traffic awareness and link preference in internet of vehicles," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 10, pp. 6705–6714, 2020. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9154533

[2] J. A. Fadhil, "A survey of challenges and solutions for internet of vehicles (iov)," *2020 IEEE 9th International Conference on Advanced Computing Technologies (ICACT)*, 2020. [Online]. Available: https://doi.org/10.1109/ACIT50332.2020.9300095

[3] K. Kandali, L. Bennis, O. E. Bannay, and H. Bennis, "An intelligent machine learning based routing scheme for vanet," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 4, pp. 4470–4483, 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9829551

[4] O. Jafarzadeh, M. Dehghan, H. Sargolzaey, and M. M. Esnaashari, "A new qos adaptive multi-path routing for video streaming in urban vanets integrating ant colony optimization algorithm and fuzzy logic," *Wireless Personal Communications*, vol. 118, no. 1, pp. 337–359, 2021. [Online]. Available: https://link.springer.com/article/10.1007/s11277-021-08142-7

[5] E. M. B. Karbab, S. Cherkaoui, and R. Zagrouba, "A model-based reinforcement learning protocol for routing in vehicular ad hoc network," *Wireless Personal Communications*, vol. 119, no. 1, pp. 377–401, 2021. [Online]. Available: https://link.springer.com/article/10.1007/s11277-021-09166-9

[6] L.-L. Wang, J.-S. Gui, X.-H. Deng, and Z.-F. Kuang, "Routing algorithm based on vehicle position analysis for internet of vehicles," *IEEE INTERNET OF THINGS*, vol. 7, no. 12, 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9106336

[7] S. M. Karim, A. Habbal, S. A. Chaudhry, and A. Irshad, "Architecture, protocols, and security in iov: Taxonomy, analysis, challenges, and solutions," 2022. [Online]. Available: https://doi.org/10.1155/2022/1131479

[8] Y.-C. Wang and Y.-W. Huang, "Knowledge development trajectory of the internet of vehicles domain based on main path analysis," *Sensors*, vol. 23, no. 13, p. 6120, 2023. [Online]. Available: https://www.mdpi.com/1424-8220/23/13/6120

[9] T. Shon, "In-vehicle networking/autonomous vehicle security for internet of things/vehicles," vol. 10, no. 6, p. 637, 2021. [Online]. Available: https://www.mdpi.com/2079-9292/10/6/637

[10] F. Ajaz, M. Naseem, G. Ahamad, Q. R. Khan, S. Sharma, and E. Abbasi, *Routing Protocols for Internet of Vehicles: A Review*. Springer Singapore, 2021, pp. 95–103. [Online]. Available: https://link.springer.com/chapter/10.1007/978-981-33-4412-9_5

[11] L. Wang, J.-s. Gui, X. Deng, and Z. Kuang, "Routing algorithm based on vehicle position analysis for internet of vehicles," in *2020 International Conference on Computer Communications (INFOCOM WKSHP)*, vol. 7, no. 12. IEEE, 2020. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9106336