IET Communications

WILEY

IET The Institution of Engineering and Technology

ORIGINAL RESEARCH OPEN ACCESS

Graph Neural Network-Based Task Offloading and Resource Allocation for Scalable Vehicular Networks

Menghan Shao[1] | Rongqing Zhang[2] | Liuqing Yang[3]

[1]Internet of Things Thrust, The Hong Kong University of Science and Technology (Guangzhou), Guangzhou, China | [2]School of Computer Science and Technology, Tongji University, Shanghai, China | [3]Internet of Things Thrust and Intelligent Transportation Thrust, The Hong Kong University of Science and Technology (Guangzhou), Guangzhou, China

Correspondence: Rongqing Zhang (rongqingz@tongji.edu.cn)

Received: 23 April 2025 | **Revised:** 12 June 2025 | **Accepted:** 9 July 2025

Funding: This work was supported by Guangdong Provincial Project under Grant Number 2023ZT10X009, Guangzhou Municipal Science and Technology Project under Grant Number 2023A03J0011, Natural Science Foundation of China under Grant Numbers U23A20339, 62271351, Department of Education of Guangdong Province Key Project under Grant Number 2023ZDZX1037.

ABSTRACT

Intelligent vehicles require extensive data processing to enhance safety and improve driver comfort. With limited onboard computing resources, these vehicles often offload tasks to nearby vehicles or servers for auxiliary processing to meet real-time response requirements. However, the complexity and highly dynamic nature of the vehicular environment render the design of effective offloading strategies. While existing approaches can adapt to changes in environmental parameters within vehicular networks, they are fundamentally limited by their inability to process variable-dimensional environmental information and make decisions that scale with network size. Traditional methods typically rely on fixed-size input representations and static computational frameworks, which are inherently unsuitable for the dynamic and scalable nature of real-world vehicular networks that require adaptive responses to varying network sizes. As a result, existing alternatives lack feasibility to highly dynamic real-world vehicle networks that require adaptive responses to varying network sizes. To alleviate this limitation, we develop an original approach to address the task offloading and resource allocation problem with a scalable size, via a framework based on a graph neural network (GNN). Leveraging its neighbour aggregation mechanism, GNN effectively adapts to varying-scale topologies in dynamic vehicular networks, ensuring robust performance regardless of network size. To evaluate our proposed approach, we conducted extensive simulations to analyse its performance. The experimental results demonstrate that our method provides a more scalable and real-time capable solution, surpassing existing approaches by seamlessly handling dynamic network size variations.

1 | Introduction

In recent decades, vehicles have been transformed from mere transportation into intelligent mobility systems through the integration of various sensors and advanced algorithms. These innovations enable self-perception, autonomous decision-making, and self-learning capabilities, significantly enhancing vehicular intelligence. Vehicular services require high computational intensity and ultra-low latency due to the need to process substantial sensor data volumes and execute intricate algorithms for decision-making and control, as well as the requirement for immediate response to diverse traffic scenarios to avert accidents [1, 2].

The vehicle's own on-board unit (OBU) has constrained computing resources due to its size and cost, making it insufficient to

meet the high computational demands. To address this limitation, vehicles can offload their tasks to other computing platforms such as the central cloud, edge servers, or nearby vehicles to aid in computation. Although task offloading is a promising solution, its implementation presents significant challenges due to the highly dynamic nature of vehicular networks, which are characterised by rapid topology changes and unpredictable vehicle movements [3].

The implementation of intelligent transportation systems, such as autonomous driving and road condition monitoring, requires substantial data processing and computation. However, the limited computational capacity of individual vehicles makes it impractical to handle such complex tasks independently. Offloading tasks to a remote cloud data centre is another potential approach, but it is often hindered by high data transmission latency and constrained backhaul bandwidth. In this context, edge computing, particularly when integrated with road side units (RSUs), has emerged as a promising solution for providing the necessary computational assistance [4, 5]. However, RSUs have restricted geographical coverage and cannot be densely deployed due to their high infrastructure and maintenance costs. To overcome these limitations, researchers have proposed offloading computational tasks to surrounding vehicles with idle resources, aiming to improve resource utilisation and reduce computational latency [6, 7]. Despite their mobility, vehicles moving in the same direction exhibit minimal communication delays, making them viable candidates for efficient task offloading.

Making decisions in highly dynamic vehicular networks is inherently challenging. An effective offloading strategy must not only select the optimal device for task execution but also dynamically allocate computational resources in response to real-time demands. These decisions are complicated by multiple dynamic factors, including unpredictable topology changes, time-varying communication quality, and fluctuating resource availability. Among these, the evolving network topology is particularly critical yet often overlooked. As vehicles continuously join and leave the network, the number of active nodes fluctuates, requiring an offloading strategy that is both adaptive to rapid changes and scalable across different operational timeframes. Moreover, the stringent delay requirements of vehicular tasks further complicate decision-making, as timely responses must be ensured despite the network's dynamic nature.

Existing methods struggle to address these challenges. Exact algorithms, while theoretically optimal, are computationally expensive and impractical for real-time decision-making. Heuristic approaches, though computationally feasible, lack adaptability to the highly dynamic vehicular environment [8]. Recently, machine learning (ML)-based methods have been explored to handle these dynamic conditions, with deep neural networks (DNNs) emerging as a popular choice [9–11]. However, DNNs face inherent limitations in managing the ever-changing network topology, as they rely on fixed structures. Current approaches attempt to address this issue by padding input data with zeros or applying output masks, but these methods introduce inefficiencies and degrade performance in dynamic scenarios. To overcome these limitations, we propose a flexible task offloading scheme based on graph neural network (GNN). GNNs are well-suited for modelling the dynamic topology of vehicular networks, as they can adapt to varying network sizes and structures, offering a more flexible and efficient solution.

We consider a scalable vehicular network scenario where multiple task vehicles and service vehicles collaborate to achieve an optimal task offloading and resource allocation strategy, aiming to minimise the average delay and enhance the responsiveness of vehicular applications. Simulation results demonstrate that the proposed method effectively adapts to network scale variations, striking a balance between decision-making inference time and task processing efficiency. Our contributions are as follows:

- Scalable vehicular network modelling: We model the task offloading and resource allocation problem in a vehicular network characterised by a dynamic topology and varying network sizes. Unlike static or fixed-scale networks, vehicular networks experience continuous topology changes and variations in the number of active nodes, making offloading decisions significantly more complex. By addressing the scalability of the network, we account for the inherent flexibility required in real-world vehicular scenarios, ensuring that our solution adapts well to fluctuating numbers of task and service vehicles.
- Graph-based network representation: To effectively represent the highly dynamic and heterogeneous nature of vehicular networks, we construct a graph-based model where nodes represent vehicles and edges capture the time-varying communication links between them. Unlike traditional DNN-based offloading methods, which often rely on fixed or handcrafted feature representations, our graph-based abstraction dynamically reflects real-time network states. This allows for a richer encoding of topological attributes such as link connectivity and resource availability. By leveraging this structured representation, we enable more informed, topology-aware offloading decisions that are inherently adaptable to network fluctuations.
- GNN-based task offloading and resource allocation (G-TORA) framework: We introduce a GNN-driven approach to model and optimise task offloading in vehicular networks. GNNs inherently support variable-sized input graphs and can generalise across different network scales, making them exceptionally suited for vehicular environments where the topology evolves rapidly. This structural adaptability empowers G-TORA to maintain high performance under dynamic conditions without retraining or extensive feature engineering. Simulation results demonstrate that G-TORA consistently meets stringent latency requirements while scaling efficiently to large networks, offering a unique and robust solution to the challenges of dynamic topology and scalability.

The remainder of this article is organised as follows: Section 2 summarises the literature related to task offloading and resource allocation in vehicular networks. Section 3 presents the scalable task offloading problem formulation, with the GNN-based

framework introduced in Section 4. In Section 5, we evaluate the performance of our solution. Finally, Section 6 gives the conclusion and future work.

2 | Related Works

Task offloading and resource allocation in vehicular networks refer to the process of transferring computationally intensive tasks from resource-constrained vehicles to more capable service vehicles or edge servers. The aim is to enhance the computational capabilities of vehicles, improve resource utilisation efficiency, and enable the seamless execution of resource-demanding applications. In the literature, two main categories of algorithms have been extensively studied for task offloading in the vehicular networking domain: heuristic-based approaches and learning-based techniques. This section provides an overview of the current research status in these two categories, highlighting the key challenges encountered in designing an efficient and reliable task offloading mechanisms for vehicular networks.

2.1 | Meta-Heuristic Based Task Offloading in Vehicular Networks

In [12], a hybrid adaptive particle swarm optimisation algorithm was proposed for efficient task scheduling and resource allocation across three cloud layers, improving service quality for on-road users. In [13], Hameed et al. proposed a method that considers vehicle mobility and latency requirements while balancing edge computing system load. They combined deep neural networks with particle swarm optimisation to jointly optimise decision variables for meeting latency demands in vehicular applications. An online and offline multi-armed bandit algorithm was introduced in [14] to address offloading in heterogeneous vehicular edge environments, focusing on base station selection to minimise congestion using offloading history for delay prediction. Considering the frequent disconnections due to the rapid mobility of vehicles in vehicular networks, Barbosa et al. [15] proposed a bee colony-based task offloading algorithm for vehicular edge computing. This algorithm optimises task scheduling in 5G networks by leveraging contextual parameters and wireless access. While heuristic algorithms offer near-optimal solutions efficiently, they struggle to adapt to the fast-changing vehicular environment, requiring frequent re-execution and significant computational overhead, limiting real-time decision-making.

2.2 | Learning-Based Task Offloading in Vehicular Networks

Learning-based approaches are more adaptable to dynamic environments compared to traditional optimisation methods or heuristics, as they can learn and adjust to changing conditions over time. In particular, reinforcement learning (RL) has been widely adopted for task offloading and resource allocation in vehicular networks [10, 16], capitalising on its ability to effectively adapt to the constantly evolving network conditions. Due to the dynamic nature of vehicles, it is almost impossible to model the changes in network conditions accurately. Consequently, most existing RL methods in the field of vehicular networking have adopted a model-free approach.

Q-learning is widely used for its simplicity. In [17], a hierarchical edge computing model for mobile edge-cloud environments was developed, using Q-learning for joint optimisation of offloading decisions, resource allocation, and bandwidth to minimise system costs. Similarly, Asheralieva et al. [18] introduced a Q-learning framework combined with game theory for computation offloading in MEC networks operated by multiple service providers. To handle vehicular mobility, Liang et al. [19] employed a Q-network-based RL algorithm, improving decision-making in a semi-Markov decision process without needing knowledge of vehicular request distributions. Yan et al. [20] proposed a deep Q-learning-based solution for optimising both network selection and driving strategies to improve traffic flow and minimise collisions. As RL problems grow in complexity, deep deterministic policy gradient (DDPG) methods have been introduced. Peng et al. [21] used DDPG to optimise spectrum, computing, and storage resources for different vehicular applications. Similarly, Wang et al. [22] applied DDPG to optimise UAV-assisted edge computing systems, addressing user scheduling and task offloading under time-varying channel conditions.

Multi-agent reinforcement learning (MARL) offers a decentralised approach where each vehicle acts as an intelligent agent, optimising decisions based on partial observations. MARL can accelerate task learning through communication and parallelisation, making it suitable for the dynamic nature of vehicular networks. Budhiraja et al. [23] used a soft actor-critic approach to reduce delay and energy consumption in vehicular networks. To address network unavailability caused by the high-speed mobility of vehicles, Wu et al. [24] employed MARL to learn the dynamic communication states between vehicles and edge nodes, enabling timely processing of vehicular tasks. Tehrani et al. [25] proposed a federated DRL approach in which base stations collaborate by sharing model weights instead of data. Ma et al. introduced a platoon-assisted vehicular edge computing system [26], employing MARL to address a multi-task offloading problem using a multi-leader, multi-follower Stackelberg game. Hou et al. [27] introduced a distributed task offloading strategy using counterfactual multi-agent learning to balance tasks and resources within heterogeneous fog computing vehicular networks. Although these works account for the dynamic nature of the environment, they have overlooked the scalability of the number of vehicles in vehicular networks, often assuming a fixed network size across different time slots. The RL methods they employ are also typically based on DNNs, which have a fixed network structure that limits their ability to adapt to changes in network scale.

Vehicle dynamics pose scalability and real-time challenges for task offloading and resource allocation strategies. Although there has been substantial research on task offloading in vehicular networks, they have not taken into account the scalable network dimension and the topological information between vehicles. To bridge this gap, we propose a GNN-based task offloading and resource allocation framework that optimises offloading decisions in the network by leveraging its flexible input and output dimensions and powerful capacity for extracting topological features.

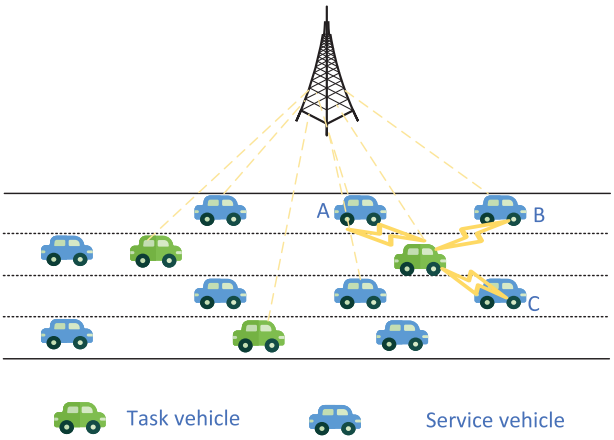

FIGURE 1 | Vehicular network architecture.

3 | System Model and Problem Formulation

3.1 | System Model

We consider a vehicular network architecture shown in Figure 1. The network consists of one BS that covers a one-way road, allowing neighbouring vehicles to communicate with each other. The task scheduling time is discretised into several time slots. During each time slot, the status of the vehicular network can be considered constant, albeit changing across slots [28]. The BS is responsible for making task offloading decisions consistently, leveraging its longer communication range compared to the vehicular communication range. When a vehicle enters the coverage area of a BS, it sends a message to the BS containing its task state, position, and computational capability. Consequently, the agent deployed in the BS can collect the vehicle information on the road within its coverage area and make decisions for vehicles. During a specific period, several vehicles lack the computational capability to execute certain onboard applications. We refer to these vehicles as task vehicles. Conversely, there are vehicles with idle resources available to assist in computations, and we classify them as service vehicles.

In a time slot, there are M_t task vehicles and N_t service vehicles are distributed on a BS coverage area. Different from other works that assume the number of vehicles in the system is fixed across different time slots, we believe the number of vehicles in one BS coverage area changes with time, which is more practical in the real world. The set of task vehicles and service vehicles are denoted by $\mathbb{M}_t = \{1, 2, \ldots, M_t\}$ and $\mathbb{N}_t = \{1, 2, \ldots, N_t\}$, respectively. Each task vehicle is connected to one or multiple service vehicles nearby through V2V wireless links and can offload its tasks to the corresponding service vehicles. Each task vehicle has one task per slot and the task profile is defined by a triple tuple $< d_i, c_i, f_i >$, where d_i represents the input data size, c_i is the computational requirements, f_i is used to represent the local computing capacity, which are measured in the number of CPU cycles. The computing capacity of the service vehicle is denoted as f_j, and the available bandwidth is B. The vehicular network can be modelled as a graph $G_t = (V_t, E_t)$, where nodes represent task and service vehicles and edges represent the channel information between vehicles.

3.2 | Communication Model

In this part, we describe the communication model for vehicles, in which task and service vehicles are connected via vehicle-to-vehicle wireless channels. Assuming that during the data transmission process of each computing task, the wireless channel state between the task and service vehicles remains unchanged. Based on Shannon's theorem, we can calculate the transmission rate between the task and service vehicles

$$R_{i,j} = B \log_2 \left(1 + \frac{P_i H_{i,j}^2}{\sigma^2} \right) \tag{1}$$

where B is the bandwidth between vehicles, $P_{i,j}$ is the transmit power of task vehicle i, $H_{i,j}$ is the channel gain between task vehicle i and service vehicle j, and σ^2 represents the background noise of the Gaussian distribution.

3.3 | Computation Model

There are two ways to perform computing tasks: one is to compute locally, where the task vehicle itself completes the computing tasks; the other is to offload the tasks to nearby service vehicles, utilising their more powerful computing capabilities to complete the tasks.

1. Local computing: For local computing, the task generated by the vehicle is executed on the local processor, and then the delay only depends on the local processor frequency f_i, which can be given as

$$T_i^{\text{loc}} = \frac{c_i}{f_i} \tag{2}$$

2. Offloading computing: If the task is offloaded to the service vehicle, there are three steps for the task to completion. The task vehicle first sends the task to the service vehicle through a wireless channel, and then the service vehicle performs the computation. After the computation, the result is sent to the task vehicle.
   The transmission delay from the task vehicle to the service vehicle depends on the data size of the task and transmission rate.

$$T_{i,j}^{\text{trans}} = \frac{d_i}{R_{i,j}} \tag{3}$$

   After receiving the task, the service vehicle will allocate its remaining computing resources to assist in the computation of the task. The computation latency of the task performed on service vehicle j can be calculated by,

$$T_{i,j}^{\text{edge}} = \frac{c_i}{f_{i,j}} \tag{4}$$

   where $f_{i,j}$ is the computation frequency allocated by service j to task vehicle i. After the execution, the service vehicle returns the processing result to the corresponding task vehicle. According to the research [29, 30], due to the data size of computing results being much smaller than the input data, the return delay of tasks can be ignored. In summary, for

the vehicular network, the response time of the set of tasks generated at slot t can be expressed as

$$T_t = \sum_{i \in \mathbb{M}_t} \left(x_{i,0} T_i^{\text{loc}} + \sum_{j \in \mathbb{N}_t} x_{i,j}^t \left(T_{i,j}^{\text{trans}} + T_{i,j}^{\text{edge}} \right) \right) \quad (5)$$

3.4 | Problem Formulation

In the proposed vehicular network system, we consider that BS makes offloading decisions for multiple task vehicles within the coverage area across different time slots. Different from the existing work that assumes the size of the network topology is fixed all the time, we believe that the number of task vehicles and service vehicles of the entire network will change in different slots, such as the addition and deletion of nodes. During each time slot, task vehicles may offload their computational tasks to corresponding service vehicles and service vehicles will allocate their idle resources to task vehicles. Our goal is to minimise the average response delay during this period by designing an effective task-offloading and computing resource allocation strategy. The total response delay of tasks in the period is

$$T(G_t, \mathbb{X}) = \sum_{t \in \mathcal{T}} \sum_{i \in \mathbb{M}_t} \left(x_{i,0}^t T_i^{\text{loc}} + \sum_{j \in \mathbb{N}_t} x_{i,j}^t \left(T_{i,j}^{\text{trans}} + T_{i,j}^{\text{edge}} \right) \right) \quad (6)$$

The optimisation problem is formulated as follows:

$$\min T(G_t, \mathbb{X}) \quad (7)$$

$$\text{s.t.}, C_1 : x_{i,0}^t + \sum_{j \in \mathbb{N}_t} x_{i,j}^t = 1, \quad \forall i \in \mathbb{M}_t$$

$$C_2 : \sum_{i \in \mathbb{M}_t} f_{i,j}^t \le f_j, \quad \forall j \in \mathbb{N}_t$$

$$C_3 : x_{i,j}^t \in \{0, 1\}, \quad \forall i \in \mathbb{M}_t, \quad \forall j \in \mathbb{N}_t$$

Constraint C_1 ensures that each task must be executed locally or on the service vehicle. $C2$ refers to the constraint that the total computation resources allocated by each service vehicle to task vehicles must not exceed its maximum computational resource limit. In constraint C_3, $x_{i,j}^t = 1$ means that task vehicle i will offload its to vehicle j, while $x_{i,j}^t = 0$ means not.

The optimisation problem is NP-hard due to the combinatorial nature of task offloading decisions and the continuous resource allocation variables. Once the computation offloading action at time slot t is determined, that is, $x_{i,j}^t$ as a deterministic value, the original problem simplifies into a convex optimisation problem. Therefore, we focus on the offloading variables in the next section. Exact algorithms are slow due to the large search space, and traditional DNNs lack adaptability to the dynamic, scalable nature of vehicular networks. To overcome these limitations, we propose a GNN-based approach that exploits the graph structure of vehicular networks. GNNs inherently capture the topological relationships between vehicles, making them well-suited for dynamically changing network sizes and structures.

4 | GNN-Based Task Offloading and Resource Allocation Framework

In this section, we present a comprehensive overview of the GNN-based framework for task offloading and resource allocation. First, we outline the overall workflow of the framework. Next, we describe how the GA is utilised to generate task offloading labels for the GNN. Finally, we detail the process of training the GNN model using the vehicular network topology graph and the generated labels to enable inference.

4.1 | Workflow of the Framework

The dynamic nature of vehicular networks is particularly reflected in the changing number of vehicles within a BS's coverage area as vehicles enter and exit. This constant fluctuation alters the scale of the optimisation problem, as the number of available task and service vehicles varies over time, adding complexity to the task offloading and resource allocation process. Therefore, it is crucial to design an offloading scheme that can make real-time decisions and adapt to varying vehicular topology sizes.

We propose a GNN-based framework integrating GA and GNN to solve the scalable task offloading and resource allocation problem, which optimises task offloading in vehicular networks by leveraging GA for data labelling and GNN for decision-making. The workflow of the proposed framework is shown in Figure 2. Our framework will be structured in three stages: Firstly, the data preparation stage, we will sample network parameter instances and use GA to address the task offloading problem, in order to use the results as labels for subsequent training of GNN. Secondly, the offline training stage entails the conversion of each problem instance from the first stage into a graph, followed by the utilisation of the data labels obtained in the first stage for training a GNN network for decision-making. Finally, the third stage involves the online inference phase, where the status of the vehicular network is acquired and the trained GNN network is deployed at the BS for making direct task offloading decisions.

Overall, the framework consists of three main stages:

- Data preparation stage: Firstly, the genetic algorithm plays a crucial role in the algorithm by facilitating the labelling of training samples. This is essential for the subsequent training of the GNN. The genetic algorithm employs evolutionary principles to iteratively improve the labelling of training samples, thereby enhancing the quality and relevance of the data used for GNN training.
- Offline training stage: Following the acquisition of labelled training samples, the GNN undergoes supervised learning in the offline phase. This phase is vital for enabling the GNN to learn and extract complex patterns and features from the labelled data, ultimately enhancing its ability to make informed task offloading decisions.
- Online inference stage: Once the GNN has completed its training, it can be deployed at the base station to autonomously make task offloading decisions. Leveraging the knowledge and patterns learned during training, the GNN can

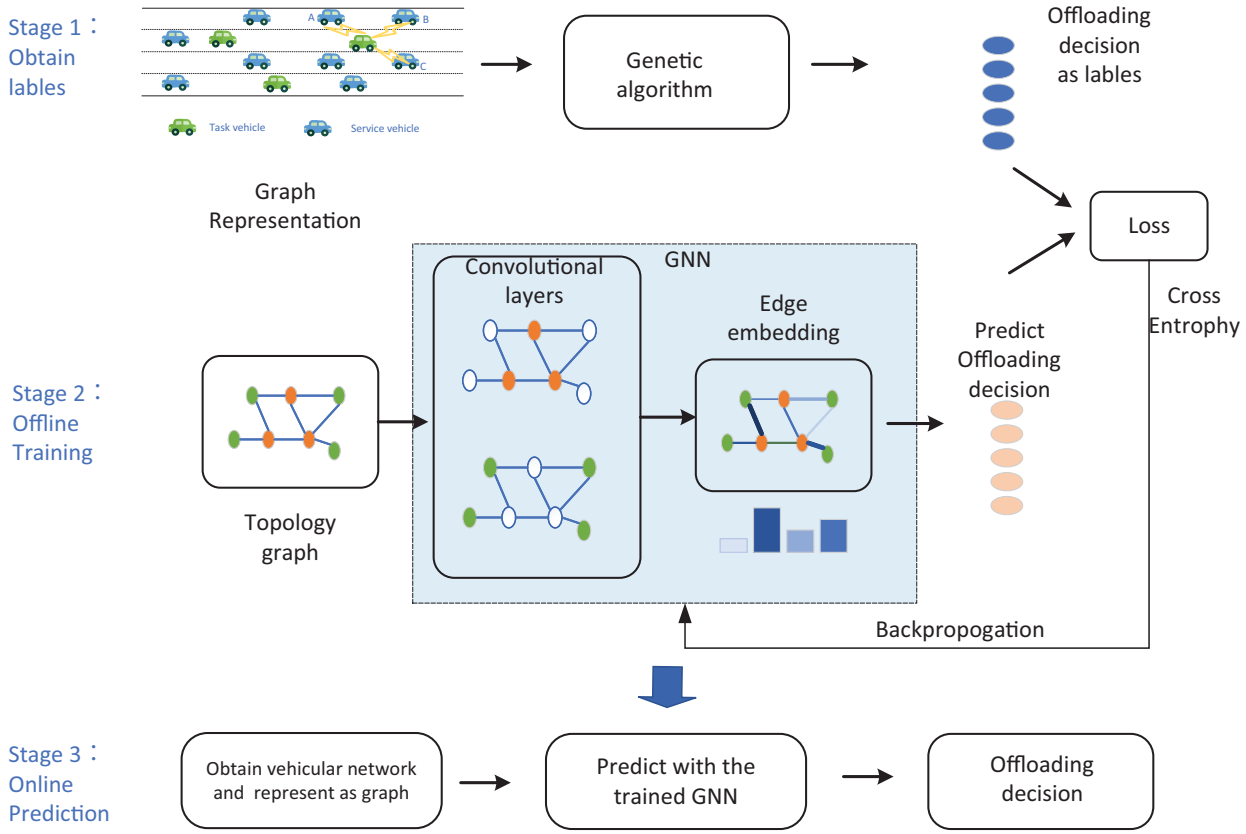

FIGURE 2 | GNN-based task offloading framework in vehicular networks.

effectively evaluate and determine the most efficient and optimal offloading strategies, contributing to improved network performance.

Then, we introduce each module of the learning-based framework in detail. We first discuss how to use GA to address the task offloading problem. Then, we provide a detailed explanation of how to transform vehicular network information into a graph that can be input into GNN, as well as how to use the labels obtained from the GA algorithm to train GNN to assist in making task offloading decisions.

4.2 | Label Generation Using Genetic Algorithm

To effectively train a GNN for making offloading decisions, we first need to sample instances from the vehicular network and generate corresponding offloading decisions as training labels. One possible approach is to perform an exhaustive search, exploring all possible offloading combinations. However, this would involve examining a vast number of decision variable combinations, which cannot be solved in polynomial time, making it impractical for large-scale networks.

To address this challenge, we employ a low-complexity heuristic algorithm to find approximate solutions to the offloading problem. Specifically, we integrate GA within the GNN-based framework, as GA is capable of efficiently searching for near-optimal solutions in complex problem spaces.

In this approach, the offloading decisions are encoded into chromosomes, where each chromosome represents a potential solution. The search space of each chromosome corresponds to the entire solution space of the offloading problem. Initially, we randomly generate offloading decisions to form the chromosomes of the initial population. These chromosomes are then evaluated using a fitness function, where the fitness is inversely proportional to the task response delay; in other words, chromosomes that yield lower response delays are considered fitter.

Based on the fitness scores, we apply selection to choose chromosomes with higher adaptability for the next generation. Subsequently, crossover and mutation operations are performed to introduce diversity and explore the solution space. This evolutionary process continues through multiple generations, iterating until a predefined stopping criterion is met. At the end of this process, the chromosome with the highest fitness will be selected as the optimal offloading decision.

The detailed steps of the GA are outlined in Algorithm 1. The network instances and offloading decisions obtained through this process will be used as training data for the GNN in the subsequent training phase.

4.3 | Training GNN With Vehicular Network Data

In a real dynamic vehicular network, the continuous movement of vehicles leads to substantial changes in the network topology, including variations in communication links between vehicles

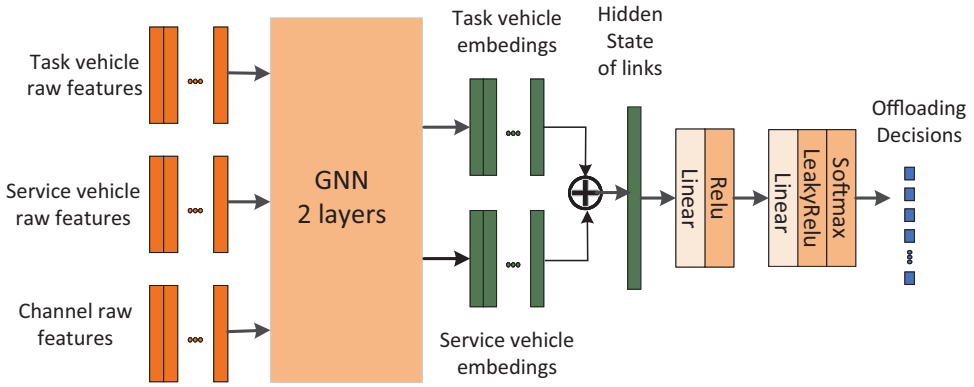

FIGURE 3 | GNN-based network architecture.

ALGORITHM 1 | Genetic algorithm for task offloading

Input: Task vehicle information, service vehicle information, and channel information for different time slots.

Output: Suboptimal task offloading decisions for vehicles.

1: Initialise the population by generating random offloading decisions $\boldsymbol{Q}^k = \{x_1^k, x_2^k, \ldots, x_M^k\}$ for each chromosome, where $k = 1, 2, \ldots, pop\_size$.
2: **for** each generation **do**
3: **for** each chromosome $\boldsymbol{Q}^k$, $k = 1, 2, \ldots, pop\_size$ **do**
4: Calculate the total response delay for $\boldsymbol{Q}^k$ based on Equation (6).
5: Compute fitness function for each chromosome, $\text{Eval}(\boldsymbol{Q}^k) = \beta(1-\beta)^{k-1}$.
6: **end for**
7: Select a new population using roulette wheel selection based on fitness scores.
8: Perform crossover on selected chromosomes with crossover probability P_c.
9: Perform mutation on chromosomes with mutation probability P_m.
10: **end for**
11: Return the best chromosome $\boldsymbol{x}^*$ as the optimal offloading decision.

and the number of vehicles in one base station coverage area. While DNN-based networks, such as RL, are capable of addressing the dynamic nature of vehicular network states, their lack of scalability restricts them from processing fixed-dimensional information, rendering them unable to accommodate the flexible topological changes inherent in vehicular networks. GNN demonstrates strong adaptability and scalability. In this section, we explain how to transform the information from vehicular networks into a graph and how to train a GNN network to assist in offloading computations (See Figure 3).

Graph data is suitable for representing the vehicular task network. We will first transform the tasks and vehicular environment into a graph G so that they can be input into GNN for training. In the GNN model of the vehicular network, the vertices of the graph denote the vehicles within the network, which are primarily categorised into two essential types of nodes: task nodes and computing nodes. The features of task nodes are defined by two key attributes: the size d_i of the task and the computational requirement c_i of the task nodes. The feature of computing nodes is denoted by the computational capability f_j of the vehicle. This indicator reflects the potential ability of the service vehicles in processing tasks. The edges of the graph represent the communication links between vehicles, and their features are characterised by the transmission rate between nodes. This quantitative measure is crucial, as it determines the efficiency of information transmission between vehicles, thus affecting the delay performance of the entire vehicular network system.

At a specified temporal snapshot t, the BS collects the information on vehicles, and the network status is captured by G_t. These states and tasks are transformed into inputs suitable for processing by the GNN. The GNN is employed to discern the hidden relational state among nodes within the network. The process commences with node feature updates, which are executed by aggregating information disseminated from neighbouring nodes. After this aggregation, a fusion of both node and neighbour features is conducted to extract the latent node features. Then, the node features and edge features are combined to encapsulate the relationship dynamics between the task and service vehicles that are interconnected by the edges in the problem. Such features are then utilised as inputs for the multi-layer perception (MLP) which is responsible for decision-making processes. The details of node embedding and the training process is as follows. Firstly, the GNN learns the node embedding in the next state by aggregating features of neighbour nodes.

$$m_{i,j}^k = \text{AGG}(h_i^k, h_j^k), A_{i,j} \neq 0 \tag{8}$$

h_i^k and h_j^k is the hidden state of node i and j in the k convolution layer, and $m_{i,j}^k$ denotes the extracted message from neighbour vehicle i to node j, and AGG refers to an aggregation network that extracts features from neighbour nodes. We can update the node features using the following equation:

$$h_i^{k+1} = \sigma\left(W^{k+1} \cdot C_k\left(h_i^k, \alpha_{i,j} m_{i,j}, j \in \mathcal{N}(i)\right)\right) \tag{9}$$

where C_k is the concatenation operation, which combines the aggregated neighbour information with the features of the

node itself to obtain the new embedding. k is the number of convolution layers in the GNN. Since we want to aggregate information within the second-order neighbourhood of task vehicles, the number of layers is set to 2. W is a trainable neural network used to update node features from the previous input features h_i to output features h_i^{k+1}. The attention value $\alpha_{i,j}$ evaluates the influence of adjacency node j on the current node i and can be calculated as follows:

$$\alpha_{i,j} = \frac{\exp\left(\text{LeakyReLU}\left(\left[W_1\varsigma_i \| W_2\varsigma_j\right]\right)\right)}{\sum_{k\in\mathcal{N}(i)}\exp\left(\text{LeakyReLU}\left([W_1\varsigma_i \| W_2\varsigma_k]\right)\right)}, \tag{10}$$

After obtaining the new graph embedding, we need to perform further inference to get the hidden states of the edges to assist the offloading decision. We combine the features between the nodes to get the embedding of the edges and use MLP for offloading prediction.

$$x_{i,j} = \text{MLP}(C(h_i^k, h_j^k, h_{i,j})) \tag{11}$$

Intuitively, we can classify the edges between task vehicles and service vehicles to determine the offloading action. To train the GNN, we compute the cross-entropy loss between the predicted x and the offloading decision obtained by GA. Subsequently, the parameters of the GNN are updated through backpropagation. This iterative process is repeated until the algorithm converges to a stable solution. After the GNN is trained, it can be deployed on the BS for online reasoning (See Algorithm 2).

ALGORITHM 2 | GNN-based task offloading for vehicular network

Input: Vehicular network instances information and offloading labels gained from GA

Output: A well-trained GNN for task offloading

1: Transform the original vehicular network information into graph data, including nodes, edges, and attributes, and store them in the train loader.
2: Initialise the GNN model with the specified number of layers, attention mechanisms, and other hyperparameters.
3: Initialise the optimiser and loss function.
4: **for** each epoch **do**
5: Sample a batch from the train loader.
6: Feed the graph data into the model.
7: Apply attention mechanisms to aggregate information from neighbouring nodes and update node embeddings.
8: Concatenate the node features and edge features to calculate the output offloading decisions for each task vehicle.
9: Compare the predicted offloading decisions with the labels generated by GA.
10: Calculate the loss for the entire batch using cross-entropy loss.
11: Perform backpropagation to compute the gradients of the loss with respect to the model parameters.
12: Update the weights of the GNN layers using the computed gradients and the Adam optimisation algorithm.
13: **end for**
14: Save the parameters of the trained GNN and deploy it at the base station to make offloading decisions.

5 | Performance Evaluation

In this section, we conduct extensive experiments to verify that our proposed algorithm can make real-time decisions and adapt to different network topology sizes. Firstly, the environment parameters and algorithm parameters used in the experiment are introduced. Subsequently, the performance of G-TORA is compared with several baseline algorithms across different environments. These comparisons illustrate the generalisation ability and the real-time decision-making of G-TORA.

5.1 | Experiment Settings

In this section, we conducted a comprehensive evaluation of the G-TORA through numerical simulations.

5.1.1 | Network Parameters Configuration

The vehicular network parameters are depicted in Table 1. We set the task size to {10, 20, 50, 100} MB, and the computation requirements are set to $[1, 2] \times 10^9$ cycles. The computing capabilities of service vehicles are set to [3,5] GHz. Furthermore, we constrained the range of vehicles, with the communication distance set between 20 and 500 m. The bandwidth of a V2V channel is set to 10 MHz, the transmission power of each vehicle is assumed to be identical, and the channel noise power is 100 dBm/Hz. Before transforming the network parameters into a graph and feeding them into the GNN, we applied normalisation to the input data. Node features (e.g., computational capabilities and task demands) were normalised using min–max scaling to the [0,1] range to prevent feature dominance while preserving their relative differences. Edge weights were dynamically adjusted based on real-time signal-to-noise ratios, ensuring accurate reflection of communication quality.

TABLE 1 | Simulation settings.

Parameter	Value
Data size of tasks	{10, 20, 50, 100} MB
Computation requirements of tasks	$[1, 2] \times 10^9$ cycles
Computational capacity of vehicles	[3, 5] GHz
Transmit power	100 mW
Channel noise power	100 dBm/Hz
Bandwidth	10 MHz
Distance between vehicles	[20, 500] m

5.1.2 | Simulation Platform Configuration

In this article, we set the number of convolution layers of GNN to $L = 2$ and the dimensions of node embeddings to $d = 256$. For the MLP, the hidden dimensions are set to $d = 128$. The optimisation epochs are set to epochs $= 3000$. The network is updated using the Adam optimiser, with an initial learning rate $lr = 1 \times 10^{-3}$. We adopt a multi-step decay learning rate, with decay milestones set to 100, 200, 500, 1000, and 2000, and a decay factor $\gamma = 0.8$.

We utilised the Python 3.8 development environment and the PyTorch deep learning framework as the simulation software platform for our experiments. The hardware includes a machine with 12th Gen Intel(R) Core(TM) i7-12700F CPU, and one Nvidia Geforce RTX 3060 Ti.

5.2 | Baseline Algorithms

To evaluate the performance of the proposed algorithm, we compare G-TORA with the following two benchmarks.

- Genetic algorithm (GA): The genetic algorithm is a heuristic algorithm based on natural selection and genetic inheritance, and its specific process is described in Section 4.2.
- Hill climbing (CH): Hill climbing is a local search algorithm that iteratively improves the solution by exploring neighbouring solutions. It is faster than GA but often gets stuck in local optima.
- Multi-layer perceptron: MLP is based on a simple DNN network structure with fixed input and output dimensions. To compare with other methods, when the number of vehicles in the system exceeds the input dimension of MLP, the extra task vehicles will perform computing locally; when the number is less than the input dimension of MLP, we will fill the remaining dimensions with zeros [31].
- Online multi-armed bandit (OMAB): OMAB uses the ϵ-greedy approach to balance exploration and exploitation, adjusting strategies in real-time to optimise decisions under uncertainty.

For the sake of brevity, we will use GNN to represent the proposed method in subsequent figures.

5.3 | Impact of Varying Task Vehicles

We conduct an experiment to observe the impact of the varying number of task vehicles on different solutions. The number of service vehicles is fixed at 4, while the network topology is modified by varying the number of task vehicles from 5 to 10. We compare the delay performance of the algorithms, focusing specifically on three aspects: task processing delay, decision inference time, and total delay. Task processing delay refers to the time taken for both communication and computation, encompassing the time required to transmit tasks and perform the necessary computations. Decision inference time is the time needed for the algorithm to make a decision regarding task offloading or resource allocation. Finally, total delay is the sum of the task processing delay and the decision inference time, representing the overall time from task arrival to decision execution and task completion.

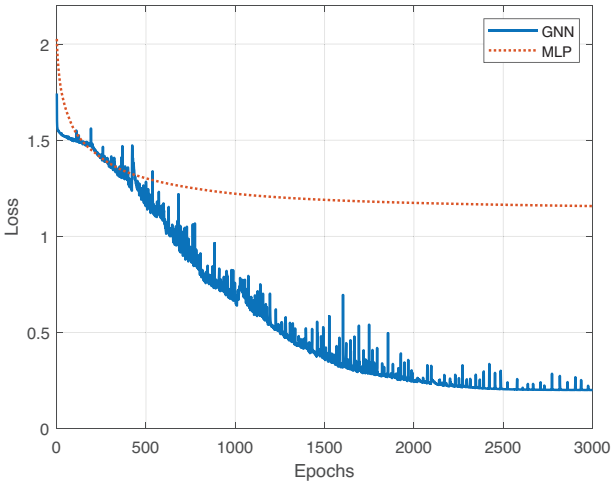

FIGURE 4 | Loss curve for GNN and MLP.

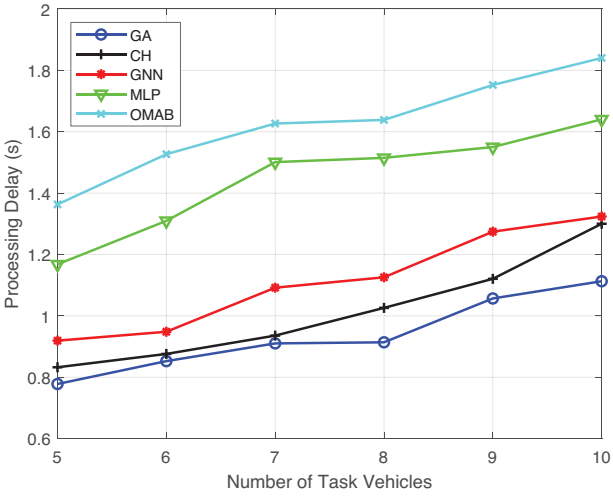

FIGURE 5 | Processing delay with the increasing number of task vehicles.

Figure 4 depicts the loss curves for the GNN and MLP. It is evident from the graph that the loss of GNN loss steadily decreases with increasing training iterations, eventually converging to a value close to 0.2. On the other hand, MLP, due to its constraint of accepting fixed-dimensional information, exhibits higher loss values, indicating its limitations in capturing complex patterns and representations.

Figure 5 presents the average task processing delay for the GNN and the other benchmarks. As the number of task vehicles increases, all methods experience higher delays due to intensified competition for limited resources. GA consistently achieves the lowest delays, showing its effectiveness in handling complex optimisation. CH has higher delays than GA but performs better than other methods, though it tends to get stuck in local optima, limiting its performance. GNN performs well, particularly in larger networks, thanks to its ability to model dynamic topologies, but still lags behind GA due to its reliance on supervised labels.

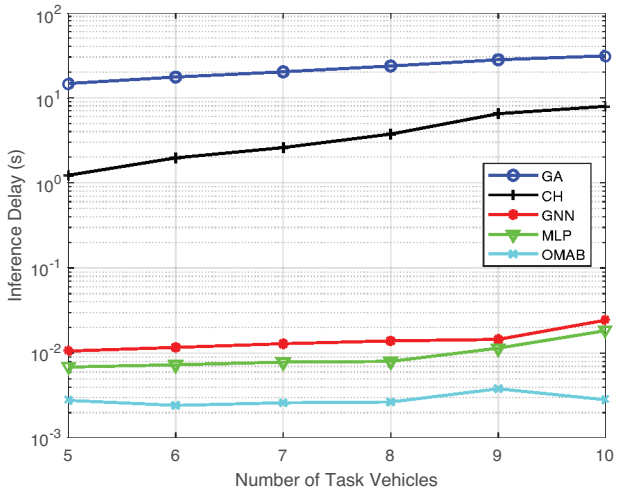

FIGURE 6 | Inference delay with the increasing number of task vehicles.

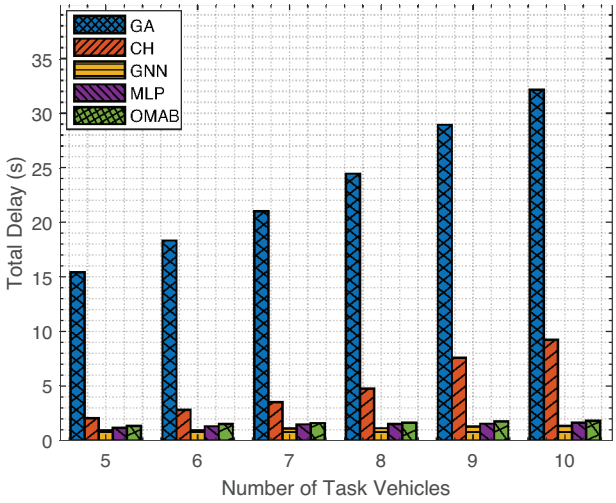

FIGURE 7 | Total delay with the increasing number of task vehicles.

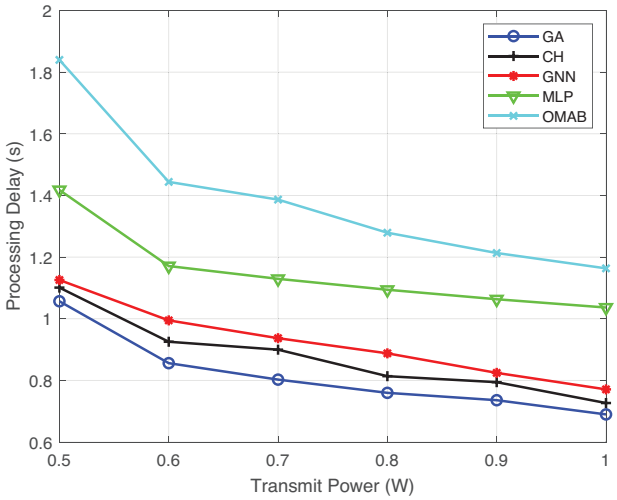

FIGURE 8 | Processing delay with the increasing transmit power.

Specifically, when the number of task vehicles reaches 10, GNN's delay is around 17% higher than GA's and about 1% higher than CH's. However, GNN significantly outperforms MLP by 26% and OMAB by 41%. MLP shows significantly higher delays because it cannot capture network topology, making it less effective. OMAB has the highest delays, as its ϵ-greedy strategy struggles with the network's dynamic nature. It is worth noting that as the number of task vehicles increases, both GNN and MLP show a slightly widening gap compared to GA. This is attributed to the growing decision space, which introduces a degree of precision loss. Despite this, GNN remains far superior to MLP in maintaining delay performance under increasing task loads.

Figure 6 presents the decision inference delay across different algorithms. GA incurs the longest decision time due to recalculating at each slot, which increases sharply as task vehicles grow. The decision time rises from 14.664 s with five vehicles to 31.016 s with 10 vehicles, showing poor scalability. CCH exhibits a more gradual increase in time (from 1.226 to 7.929 s), offering greater efficiency than GA, though it still has a relatively long inference time. In contrast, GNN achieves consistently low decision times, staying below 0.03 s even as the number of vehicles increases, indicating its capability for real-time applications. MLP, while slightly slower than GNN, remains competitive, especially for larger networks where it stays under 0.02 s. Finally, OMAB offers the fastest inference, remaining under 0.003 s across all vehicle numbers, though it compromises solution accuracy.

Figure 7 presents the total response delay with increasing task vehicles. It can be seen that the total time delay of all the algorithms increases as the number of task vehicles increases. Both GA and CH exhibit relatively low processing delays; however, their inference delays become predominant as the scale increases, leading to a significant rise in total latency. In contrast, the GNN, despite not having the lowest processing or inference delays, achieves the smallest overall latency. This indicates a well-balanced approach that effectively manages both processing and inference delay, thereby optimising performance as the problem size scales. Additionally, both MLP and OMAB demonstrate lower inference delays. However, the optimisation accuracy of these methods is compromised, resulting in higher total delays.

5.4 | Impact of Varying Transmit Power

We conducted the second set of experiments to investigate the impact of varying transmit power on the proposed method.

Figure 8 shows the task processing delay for the algorithms. The increase in transmit power leads to a decrease in processing delays across all algorithms. GA consistently demonstrates the lowest processing delay across all power levels, with delays decreasing from 1.056 to 0.690 s as transmit power increases. CH follows closely, with a notable reduction from 1.101 to 0.727 s. The GNN algorithm exhibits a similar trend, with delays decreasing from 1.126 to 0.771 s, indicating its competitive performance in this context. In contrast, both MLP and OMAB show significantly higher processing delays than the aforementioned algorithms, with MLP starting at 1.418 and dropping to 1.036 s, while OMAB begins at 1.840 s and reduces to 1.163 s. This indicates that while

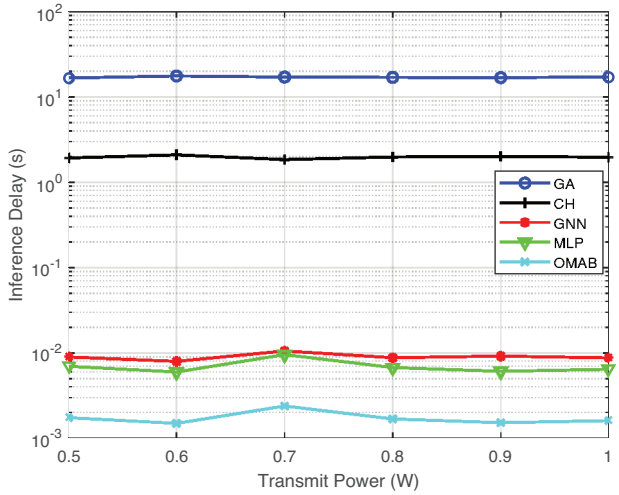

FIGURE 9 | Inference delay with the increasing transmit power.

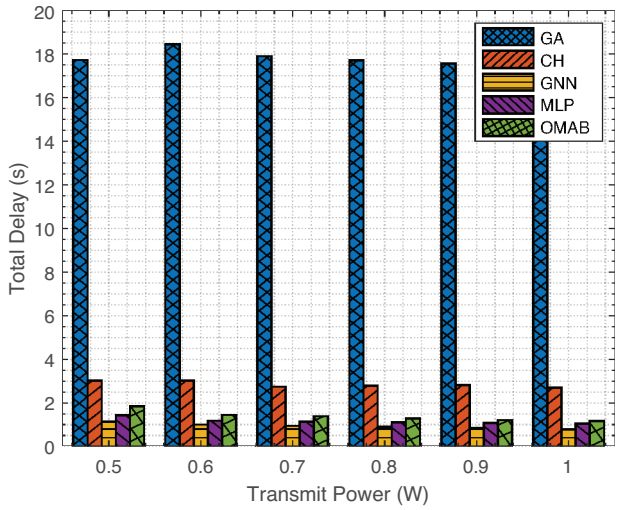

FIGURE 10 | Total delay with the increasing transmit power.

MLP and OMAB benefit from increased transmit power, they do not achieve the same level of efficiency as GA, CH or GNN.

Figure 9 shows the task inference delay with the varying transmit power. The inference delays for all algorithms remain relatively unchanged with increasing transmit power. This stability can be attributed to the fact that the overall problem size remains constant throughout the simulation, indicating that improvements in transmit power do not affect inference performance.

Figure 10 shows the total task delay as transmit power varies. As transmit power increases, GA and CH exhibit only slight fluctuations in total delay, with minimal change overall. In contrast, GNN, MLP and OMAB demonstrate a noticeable reduction in total delay. This difference arises because the total delay for GA and CH is primarily dominated by inference delays, while for GNN, MLP and OMAB, it is dominated by processing delays. As transmit power increases, the communication delay decreases, leading to a reduction in processing delay, which explains the observed decline in total delay for GNN, MLP and OMAB. GNN's total delay consistently remains the lowest, dropping from 1.135 s

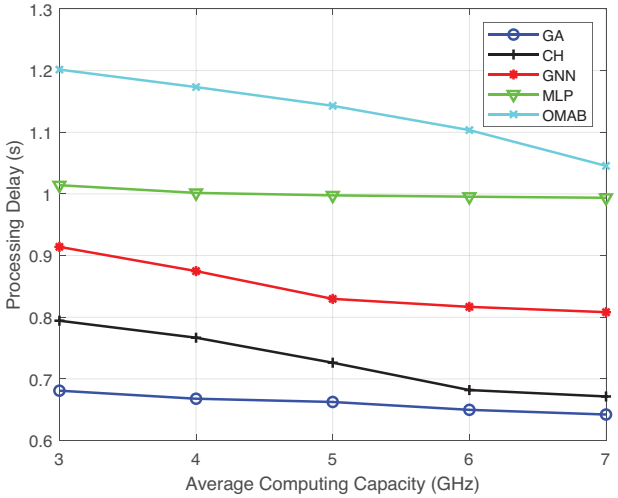

FIGURE 11 | Processing delay with the increasing computing capacity.

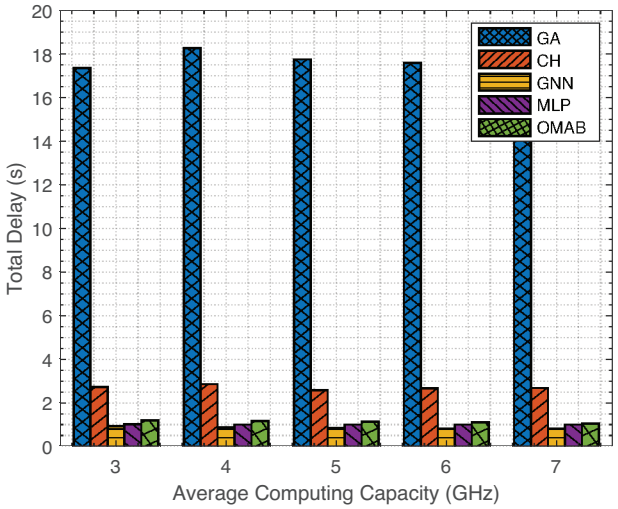

FIGURE 12 | Total delay with the increasing computing capacity.

at lower transmit power levels to 0.780 s as transmit power increases from 0.5 to 1 W.

5.5 | Impact of Varying Computing Capacity

To further validate the effectiveness of the proposed method, we change in the average computing capacity of servers from 3 to 7 GHz to inspect the variation of delay of different approaches with the increasing computing resources. The simulation results are illustrated in Figure 11. As computing capacity increases, all algorithms show a clear reduction in processing delay, underscoring the positive impact of enhanced computational power on task execution efficiency. Figure 12 illustrates the variation in total task delay as the computing capability of the server increases. For GA and CH, the total delay shows only slight fluctuations and remains relatively stable. On the other hand, GNN, MLP and OMAB experience a clear reduction in total delay. This difference arises because GA and CH are predominantly affected by inference delays, while GNN, MLP and OMAB are

influenced more by processing delays. As computing capability improves, the reduction in computation time leads to a decrease in processing delay, which results in a noticeable decrease in total delay for GNN, MLP and OMAB. GNN's total delay consistently remains the lowest, decreasing from 0.923 s at lower computing capabilities to 0.817 s as the server's computing capacity increases from 3 to 7 GHz. This demonstrates that our proposed method achieves an effective balance between optimisation accuracy and real-time inference, resulting in a significant reduction in overall latency.

6 | Concluding Remarks

Task offloading and resource allocation are critical for vehicular networks, as they greatly enhance computational efficiency and reduce latency, thereby enabling more responsive and reliable in-vehicle services and applications. However, existing offloading approaches often rely on static neural network architectures, which are limited in their ability to handle dynamic and variable-dimensional environmental inputs. This restricts their effectiveness in real-world vehicular scenarios where both the number of connected vehicles and their communication patterns fluctuate over time. To address these challenges, we proposed G-TORA, a GNN-based framework capable of learning adaptive task offloading strategies across varying network sizes and dynamic topologies. In our approach, vehicle tasks and service vehicles are represented as graph nodes, while communication links form the edges. GNNs are then used to infer real-time offloading decisions by extracting and learning from the structural and relational features of the vehicular network. Simulation results confirm that G-TORA performs well under diverse and evolving network conditions, achieving high scalability and improved task processing efficiency. Despite its promising performance, G-TORA has several limitations. First, the training process depends on a large volume of network topology data and supervision from labels generated by optimisation algorithms such as GA, which may be difficult to collect and maintain in practice. Second, the framework assumes static communication states within each time slot, which does not fully reflect the high mobility and fast-changing connectivity of real vehicular networks. In future work, we plan to address these limitations by exploring more robust and label-efficient learning paradigms such as semi-supervised or unsupervised GNN training, incorporating real-time modelling of network dynamics, and extending the framework to support QoS-aware offloading with considerations for task deadlines, reliability, sub-task dependencies, and multi-hop transmission. These enhancements will further improve the adaptability and practicality of GNN-based solutions in dynamic vehicular environments.

Author Contributions

Menghan Shao: conduct simulations and paper writing. **Rongqing Zhang:** supervise the idea and revise the paper. **Liuqing Yang:** lead the project.

Conflicts of Interest

The authors declare no conflicts of interest.

Data Availability Statement

Research data are not shared.

References

1. E. S. Ali, M. K. Hasan, R. Hassan, et al., "Machine Learning Technologies for Secure Vehicular Communication in Internet of Vehicles: Recent Advances and Applications," *Security and Communication Networks* 2021 (2021): 1–23.

2. A. Waheed, M. A. Shah, S. M. Mohsin, et al., "A Comprehensive Review of Computing Paradigms, Enabling Computation Offloading and Task Execution in Vehicular Networks," *IEEE Access* 10 (2022): 3580–3600.

3. A. B. De Souza, P. A. L. Rego, T. Carneiro, J. D. C. Rodrigues, P. P. R. Filho, and J. N. De Souza, "Computation Offloading for Vehicular Environments: A Survey," *IEEE Access* 8 (2020): 198214–198243.

4. C. Feng, P. Han, X. Zhang, B. Yang, Y. Liu, and L. Guo, "Computation Offloading in Mobile Edge Computing Networks: A Survey," *Journal of Network and Computer Applications* 202 (2022): 202–216.

5. M. Talebkhah, A. Sali, V. Khodamoradi, T. Khodadadi, and M. Gordan, "Task Offloading for Edge-IoV Networks in the Industry 4.0 Era and Beyond: A High-Level View," *Engineering Science and Technology* 54 (2024): 1–40.

6. K. Luo, Y. Wang, Y. Liu, and K. Zhu, "Collaborative Integration of Vehicle and Roadside Infrastructure Sensor for Temporal Dependency-Aware Task Offloading in the Internet of Vehicles," *International Journal of Intelligent Systems* 2025 (2025): 8064–8086.

7. R. Rauch, Z. Becvar, P. Mach, J. Gazda, "Cooperative Multi-Agent Deep Reinforcement Learning for Dynamic Task Execution and Resource Allocation in Vehicular Edge Computing," *IEEE Transactions on Vehicular Technology* 74, no. 4 (2025): 5741–5756.

8. Y. He, J. Xu, B. Zheng, J. Hu, and Y. Xie, "Timing-Oriented Task Offloading Algorithms for Internet-Of-Vehicles," *Journal of Circuits Systems and Computers* 31, no. 08 (2022).

9. A. Boukerche and V. Soto, "Computation Offloading and Retrieval for Vehicular Edge Computing: Algorithms, Models, and Classification," *ACM Computing Surveys* 53, no. 4 (2021): 1–35.

10. J. Liu, M. Ahmed, M. A. Mirza, et al., "RL/DRL Meets Vehicular Task Offloading Using Edge and Vehicular Cloudlet: A Survey," *IEEE Internet of Things Journal* 9, no. 11 (2022): 8315–8338.

11. T. Li, K. Zhu, N. C. Luong, et al., "Applications of Multi-Agent Reinforcement Learning in Future Internet: A Comprehensive Survey," *IEEE Communications Surveys Tutorials* 24, no. 2 (2022): 1240–1279.

12. S. Midya, A. Roy, K. Majumder, and S. Phadikar, "Multi-Objective Optimization Technique for Resource Allocation and Task Scheduling in Vehicular Cloud Architecture: A Hybrid Adaptive Nature Inspired Approach," *Journal of Network and Computer Applications* 103 (2018): 58–84.

13. A. R. Hameed, S. ul Islam, I. Ahmad, and K. Munir, "Energy- and Performance-Aware Load-Balancing in Vehicular Fog Computing," *Sustainable Computing-Informatics & Systems* 30 (2021): 1–16.

14. A. Bozorgchenani, S. Maghsudi, D. Tarchi, and E. Hossain, "Computation Offloading in Heterogeneous Vehicular Edge Networks: On-Line and Off-Policy Bandit Solutions," *IEEE Transactions on Mobile Computing* 21, no. 12 (2022): 4233–4248.

15. A. Barbosa, P. A. L. Rego, V. Chamola, T. Carneiro, P. H. G. Rocha, and J. N. de Souza, "A Bee Colony-Based Algorithm for Task Offloading in Vehicular Edge Computing," *IEEE Systems Journal* 17, no. 3 (2023): 4165–4176.

16. A. Mekrache, A. Bradai, E. Moulay, and S. Dawaliby, "Deep Reinforcement Learning Techniques for Vehicular Networks: Recent Advances and Future Trends Towards 6G," *Vehicular Communications* 33 (2022): 1–23.

17. Y. Li and S. Xu, "Collaborative Optimization of Edge-Cloud Computation Offloading in Internet of Vehicles," in *Proceedings of the 2021*

International Conference on Computer Communications and Networks (ICCCN) (IEEE, 2021), 1–6.

18. A. Asheralieva and D. Niyato, "Hierarchical Game-Theoretic and Reinforcement Learning Framework for Computational Offloading in UAV-Enabled Mobile Edge Computing Networks With Multiple Service Providers," *IEEE Internet of Things Journal* 6, no. 5 (2019): 8753–8769.

19. H. Liang, X. Zhang, J. Zhang, Q. Li, S. Zhou, and L. Zhao, "A Novel Adaptive Resource Allocation Model Based on SMDP and Reinforcement Learning Algorithm in Vehicular Cloud System," *IEEE Transactions on Vehicular Technology* 68, no. 10 (2019): 10018–10029.

20. Z. Yan and H. Tabassum, "Reinforcement Learning for Joint V2I Network Selection and Autonomous Driving Policies," in *2022 IEEE Global Communications Conference (GLOBECOM 2022)* (IEEE, 2022), 1241–1246.

21. H. Peng and X. Shen, "Deep Reinforcement Learning Based Resource Management for Multi-Access Edge Computing in Vehicular Networks," *IEEE Transactions on Network Science and Engineering* 7, no. 4 (2020): 2416–2428.

22. Y. Wang, W. Fang, Y. Ding, and N. Xiong, "Computation Offloading Optimization for UAV-Assisted Mobile Edge Computing: A Deep Deterministic Policy Gradient Approach," *Wireless Networks* 27, no. 4 (2021): 2991–3006.

23. I. Budhiraja, N. Kumar, H. Sharma, M. Elhoseny, Y. Lakys and J. J. P. C. Rodrigues, "Latency-Energy Tradeoff in Connected Autonomous Vehicles: A Deep Reinforcement Learning Scheme," *IEEE Transactions on Intelligent Transportation Systems* 24, no. 11 (2022): 1–13.

24. J. Wu, J. Wang, Q. Chen, et al., "Resource Allocation for Delay-Sensitive Vehicle-to-Multi-Edges (V2Es) Communications in Vehicular Networks: A Multi-Agent Deep Reinforcement Learning Approach," *IEEE Transactions on Network Science and Engineering* 8, no. 2 (2021): 1873–1886.

25. P. Tehrani, F. Restuccia, and M. Levorato, "Federated Deep Reinforcement Learning for the Distributed Control of NextG Wireless Networks," in *2021 IEEE International Symposium on Dynamic Spectrum Access Networks* (IEEE, 2021), 248–253.

26. X. Ma, J. Zhao, Q. Li, et al., "Reinforcement Learning Based Task Offloading and Take-Back in Vehicle Platoon Networks," in *2019 IEEE International Conference on Communications Workshops (ICC Workshops)* (IEEE, 2019), 1–6.

27. Y. Hou, Z. Wei, R. Zhang, X. Cheng, and L. Yang, "Hierarchical Task Offloading for Vehicular Fog Computing Based on Multi-Agent Deep Reinforcement Learning," *IEEE Transactions on Wireless Communications* 23, no. 4 (2024): 3074–3085.

28. J. Wang, C. Jiang, K. Zhang, T. Q. S. Quek, Y. Ren, and L. Hanzo, "Vehicular Sensing Networks in a Smart City: Principles, Technologies and Applications," *IEEE Wireless Communications* 25, no. 1 (2018): 122–132.

29. X. Chen, L. Jiao, W. Li, et al., "Efficient Multi-User Computation Offloading for Mobile-Edge Cloud Computing," *IEEE/ACM Transactions on Networking* 24, no. 5 (2015): 2795–2808.

30. J. Shi, J. Du, J. Wang, J. Wang, and J. Yuan, "Priority-Aware Task Offloading in Vehicular Fog Computing Based on Deep Reinforcement Learning," *IEEE Transactions on Vehicular Technology* 69, no. 12 (2020): 16067–16081.

31. X. Wang, N. Cheng, L. Fu, et al., "Scalable Resource Management for Dynamic Mec: An Unsupervised Link-Output Graph Neural Network Approach," in *2023 IEEE 34th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)* (IEEE, 2023), 1–6.