This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVT.2021.3134715, IEEE Transactions on Vehicular Technology

Traffic Routing-Based Computation Offloading in Cybertwin-Driven Internet of Vehicles for V2X Applications

Buyun Ma, *Student Member, IEEE,* Zhiyuan Ren, *Member, IEEE,* and Wenchi Cheng, *Senior Member, IEEE*

***Abstract*—With the rapid development of the Internet of Vehicles (IoV), a lot of Vehicle-to-Everything (V2X) applications have sprung up. To tackle the conflict between the resource-hungry V2X applications and the resource-constrained vehicles, most works focus on the computation offloading problem, which is significant to V2X applications by bringing computation tasks from the vehicles to the edge or cloud infrastructure. However, the dynamic network conditions caused by the mobility of vehicles will bring task migration and huge additional costs, resulting in poor latency performance. Motivated by the aforementioned problem, a traffic routing-based computation offloading scheme in cybertwin-driven IoV for V2X applications is proposed, in which cybertwin represents the network hardware devices and the network software functions. Moreover, according to the cybertwin-driven IoV network architecture, the traffic routing-based computation offloading problem is formulated. Finally, to avoid the inconsistency between the data transmission direction and the vehicle's movement direction, the enhanced Heterogeneous Earliest Finish Time (eHEFT) algorithm is designed, which introduces the gradient routing into the traditional HEFT algorithm. Performance evaluation results validate that the proposed joint optimization scheme is indeed capable of reducing latency.**

***Index Terms*—V2X applications, computation offloading, routing, latency.**

I. Introduction

WITH the rapid development of internet of vehicles (IoV), a seamless connectivity of modern transportation system is to be established [1], [2], and a number of Vehicle-to-Everything (V2X) applications (e.g. autonomous driving, image or video-aided real-time navigation, real-time traffic monitoring, etc.) have sprung up. However, a conflict exists between the resource-constrained vehicles and the resource-hungry V2X applications. To relieve this tension, cloud computing has been proposed to allow vehicles to offload their computation-intensive applications to the resource-rich cloud. However, due to the long propagation delay, cloud computing cannot meet the low latency requirements of many V2X applications.

Therefore, edge computing is envisioned to be a promising solution to address the resource-constrained challenges in vehicles. By bringing computational and storage resources to the edge of the network, an edge computing-based IoV system has been studied to improve the latency issues caused by cloud computing [3]. For example, Xiong *et al*. [4] proposed an intelligent task offloading framework in heterogeneous vehicular networks with three V2X communication technologies. Hwang *et al*. [5] proposed an analytical model for communication and computation offloading to roadside units (RSUs) and gNB to minimize average packet delay by finding an optimal offloading probability through a sub-gradient based algorithm.

However, an important property of the IoV network is its dynamic network topology owing to the mobility of vehicles, which brings great challenges for task offloading. To tackle this problem, most existing approaches [6]–[8] consider the service migration, where the RSUs decide whether to migrate the tasks to another near-user RSU on the basis of the vehicle's trajectory. In this mode, all user-related data needs to be migrated, which generates huge additional costs and even causes network congestion. To avoid the costs caused by service migration, Li *et al.* [9] proposed a deep reinforcement learning-based cooperative edge computing framework to provide reliable low-latency computing in vehicular networks. However, [9] only supports tasks for parallel computation without considering the dependencies among subtasks, which makes it difficult to extend [9] to any task models. Therefore, the existing works cannot support the latency-sensitive tasks in dynamic network conditions.

Motivated by the issues discussed above, unlike most works, we propose a traffic routing-based computation offloading strategy in cybertwin-driven IoV for V2X applications, in which the cybertwin [10] can provide fundamental support for digital twin from the network and transport layers while the digital twin serves as the digital representation of a vehicle on the application layer. Furthermore, the meaning of the cybertwin is extended, which can represent both the network devices and the network microservices. The cybertwins that represent physical devices can create digital representations for devices to implement various functionalities, e.g., communication anchor. By combining the cybertwins that represent microservices, we can provide abundant user-required services. Then, based on the cybertwin-driven IoV network architecture, a joint computation and routing problem is formulated to schedule the subtasks with dependencies on the transmission route. The computing while transmitting can be realized and the overhead caused by the service migration can be avoided. Finally, a list-based scheduling algorithm is designed to solve the problem. Specifically, our contributions of this paper can be identified as the following three points:

This work was supported in part by the National Key R&D Program of China (2019YFB1803301).

B. Ma, Z. Ren and W. Cheng are with the State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an 710071, China (e-mail: byma@s-an.org; zyren@xidian.edu.cn; wccheng@xidian.edu.cn).

• To overcome the dynamic topology of the IoV and analyze the transmission latency intuitively, we conceive Weighted time Expanded Graph (WEG) by adding the latency weight parameter for the Time Expanded Graph (TEG);

• To improve the latency performance of V2X applications, we formulate the traffic routing based computation offloading problem based on the WEG;

• To solve the above problem and avoid the inconsistency between the data transmission direction and the vehicle's movement direction, we propose an enhanced Heterogeneous Earliest Finish Time (eHEFT) algorithm by introducing gradient routing into the HEFT algorithm.

The rest of this paper is organized as follows. An overview of existing work is given in Section II. The system model and the eHEFT algorithm are given in Section III. The simulation results and analysis are given in Section IV. Finally, Section V conclude the paper.

II. Related Work

In this section, we introduce the related research from two aspects: (i) computation offloading problem; (ii) routing problem.

A. Computation Offloading Problem for V2X Applications

On one hand, a number of previous works have studied the computation offloading problem applying the cloud computing model to reduce the execution time of end devices. Chang *et al.* [11] proposed a predictive backward shockwave analysis approach based on cloud computing to realize real-time active safe driving. Ashok *et al.* [12] designed a dynamic approach for offloading specific vehicular application components or modules to the cloud server to save the network resources.

On the other hand, edge computing has been proved to be an effective method to support some latency-critical tasks. Hou *et al.* [13] proposed a reliable computation offloading strategy based on the edge-computing-enabled software-defined IoV architecture. Luo *et al.* [14] designed a fully distributed computation offloading algorithm for IoV, which is devised based on a game-theoretic model. Zhou *et al.* [15] proposed a detailed V2X framework for sharing augmented contextual information in real-time and designed a decentralized algorithm for real-time task scheduling to improve the offloading efficiency. Moubayed *et al.* [16] considered the V2X service placement problem while taking into consideration the computational resource availability at the nodes, and proposed a low-complexity greedy-based heuristic algorithm to find the optimal V2X service placement scheme. Wang *et al.* [17] studied the service migration problem in mobile edge computing where the users may move to a new location, and formulated the service migration problem as a Markov decision process (MDP). Moreover, some AI and machine learning algorithms [18]–[20] have been proposed to offload tasks on the edge servers. Ke *et al.* [18] proposed an adaptive computation offloading method based on deep reinforcement learning (ACORL) to obtain the tradeoff between the cost of energy consumption and the cost of data transmission delay. Jia *et al.* [19] proposed a deep reinforcement learning-based queue-aware task offloading algorithm to maximize the throughput of the user vehicles while satisfying the long-term queuing delay constraints in a best-effort way. Zhu *et al.* [20] proposed a deep reinforcement learning algorithm to learn an effective solution to the vehicular task offloading problem. However, both conventional algorithms and AI algorithms only consider computation offloading problems, which will result in service migration and huge additional costs.

In addition, some works studied the combination of edge computing and cloud computing on computation offloading problems to support V2X applications. Liu *et al.* [21] proposed an edge-assisted algorithm that makes use of the resources in both cloud and edge sides of vehicular networks to predict vehicle mobility. Nguyen *et al.* [22] studied the MEC and cloud computing cooperation in a three-tier offloading model of a V2X network where a vehicle can offload computational tasks to cloud computing and MEC.

B. Routing Problem for V2X Applications

With the development of IoV, routing has become a popular research topic. Through establishing wireless communication links [23], [24], the messages can be delivered from the source node to the destination node in the vehicular network. However, due to the mobility of the vehicles and the intermittent links, the conventional routing protocols cannot be directly adopted in IoV. Therefore, some works studied the routing problem in IoV environments. For example, Yao *et al.* [25] exploited the regularity of vehicle moving behaviors to achieve the predictive routing and enhance the transmission performance based on the hidden Markov model for vehicle ad hoc network. Kuhlmorgen *et al.* [26] studied the contention-based multi-hop forwarding jointly with decentralized congestion control based on the Gatekeeper and the rate control algorithm. Luo *et al.* [27] proposed an intersection-based V2X routing protocol that includes a learning routing strategy based on historical traffic flows via Q-learning and monitoring real-time network status.

However, the above works either consider the computation offloading problem or the routing problem, few works consider both of them. On one hand, only studying the computation offloading in the dynamic network will bring service migration [28] and huge additional cost, resulting in poor latency performance; on the other hand, the data needs to be further processed, and only studying the routing problem cannot meet the computation requirement of V2X applications. Therefore, this paper considers both the computation offloading problem and routing problem, conceives a traffic routing-based computation offloading strategy in cybertwin-driven IoV for V2X Applications.

III. System Model and Algorithm

As illustrated in Fig. 1, we consider a cybertwin-driven IoV network architecture [10], [29]. In our model, there are two types of cybertwin: i) cybertwin that represents network devices, ii) cybertwin that represents microservices. The cybertwin that represents network device can serve as the

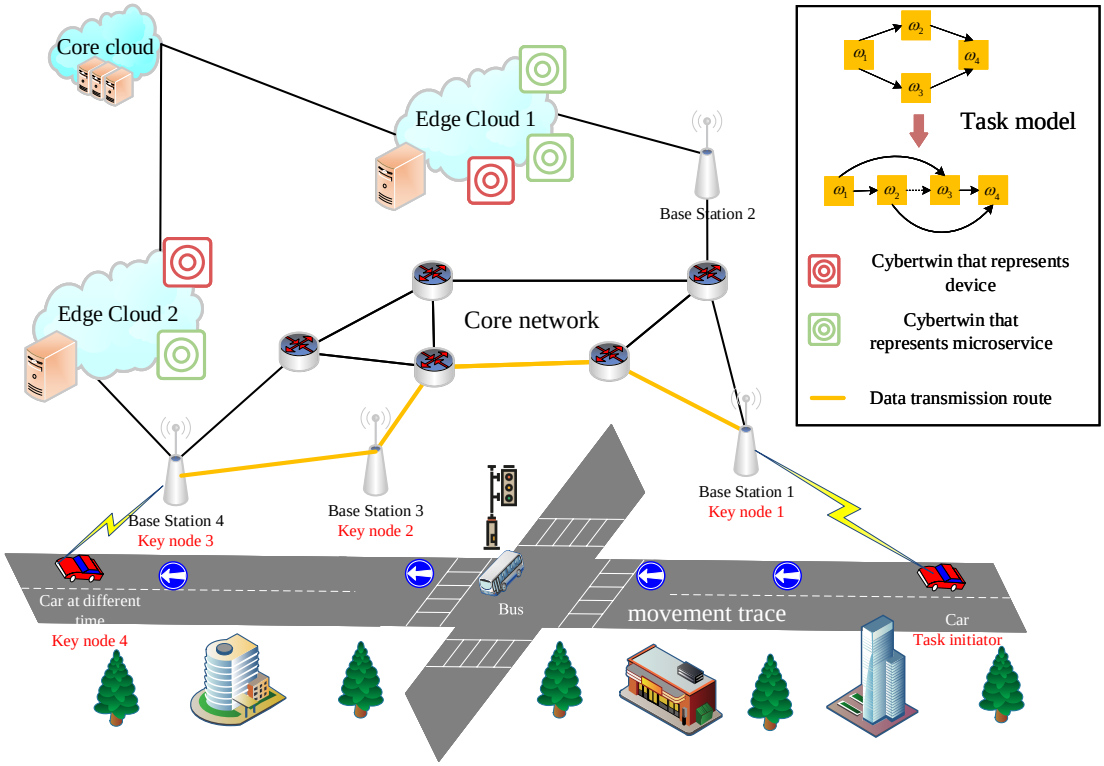

Fig. 1: Cybertwin-driven IoV network architecture.

communication assistant to acquire the required service on behalf of the device, and serve as the data logger to record the behaviors and functions of the devices. The cybertwin that represents network microservices is the virtualization of network software functions. These microservices are combined with each other to provide the required services to users.

Unlike establishing the end-to-end connections between the end device and the core cloud, the car that generates a task first connects to its cybertwin. Then, based on the network information and the vehicle navigation information, the edge cloud makes the task scheduling plan by running the scheduling algorithm, i.e., eHEFT, which transforms a Directed Acyclic Graph (DAG) task into a sequential task and selects key nodes on the transmission route as the computing node. Through collaborating, these nodes can show a strong ability to compute and transmit tasks effectively.

A. IoV Network Model

Since the trajectories of the moving vehicles can be obtained by the navigation information which has been logged by the cybertwin of the vehicles, the time window for the vehicle communication (i.e., the network topology) can be predicted. Moreover, the connections among vehicle to vehicle (V2V), and RSUs are discontinued. In summary, the cybertwin-based IoV network is time-varying and divinable.

With the properties of the cybertwin-based IoV network, we propose WEG by adding the latency weight parameter for the TEG [30], to depict the snapshot of the network for each time interval. For a given time interval $T = (t_0, t_r]$, we can split it into n small time intervals, $\{\tau_1, \cdots, \tau_p, \cdots, \tau_r\}$, where $\tau_p = (t_{p-1}, t_p]$, and the network topology is fixed during τ_p, but may change in $\tau_{p'}(p \neq p', p, p' = 1, 2, \cdots, r)$. Moreover, we define Δt as the length of the time slot, then $\tau_p = \tau_{p'} = \Delta t, (p \neq p', p, p' = 1, 2, \cdots, r)$. Due to the movement of vehicles, only the connections between the vehicles and RSUs will change, while the connections between RSUs is fixed.

We define $V = \{v_1, v_2, \cdots, v_M\}$ be the set of cybertwins that represent vehicles, where $v_i, i \in \{1, 2, \cdots, M\}$ represents the i th vehicle's cybertwin. The vehicles' cybertwins record 2-D coordinates of vehicles at time slot τ_k, which is defined as $(x_i^{\tau_k}, y_i^{\tau_k})$. $N = \{n_1, n_2, \cdots, n_S\}$ be the set of cybertwins that represent RSUs, where $n_i, i \in \{1, 2, \cdots, S\}$ represents the i th RSU's cybertwin. The 2-D coordinates of RSUs is (X_i, Y_i). Then, the distance between v_i to n_j at time slot τ_k can be given by

$$d_{ij}^{\tau_k} = \sqrt{(x_i^{\tau_k} - X_j)^2 + (y_i^{\tau_k} - Y_j)^2}. \quad (1)$$

We denote D_j is the maximum access range of n_j, and different types of RSU have different communication range. Then the conditions for the existence of a link can be given as

$$g_{ij}^{\tau_k} = \begin{cases} 1, & d_{ij}^{\tau_k} \leq D_j; \\ 0, & \text{otherwise.} \end{cases} \quad (2)$$

We define the link transmission rate between v_i and n_j is $C_{i,j}^{\tau_k}$, which can be calculated by the transmission technics [31] and is determined by the types of RSUs (e.g., 5G, 4G, Wifi, Internet, etc.). Then the transmission delay of unit bit can be given by

$$\pi_{i,j}^{\tau_k} = \frac{1}{C_{i,j}^{\tau_k}}. \quad (3)$$

In the time slot τ_k, the weighted adjacency matrix of the cybertwin-based IoV network can be described as

$$\mathbf{G}^{\tau_k} = \begin{bmatrix} 0 & \pi_{1,2}^{\tau_k} & \cdots & \pi_{1,(M-1)}^{\tau_k} & \pi_{1,M}^{\tau_k} \\ \pi_{2,1}^{\tau_k} & 0 & \cdots & \pi_{2,(M-1)}^{\tau_k} & \pi_{2,M}^{\tau_k} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \pi_{(M-1),1}^{\tau_k} & \pi_{(M-1),2}^{\tau_k} & \cdots & 0 & \pi_{(M-1),M}^{\tau_k} \\ \pi_{M,1}^{\tau_k} & \pi_{M,2}^{\tau_k} & \cdots & \pi_{M,(M-1)}^{\tau_k} & 0 \end{bmatrix}. \tag{4}$$

Furthermore, due to the change of the link status, the data may be buffered on the relay node waiting for the link to connect during the transmission process. This phenomenon can be virtually equivalent to the data cache process. The weight of the virtual link $\pi_i^{\tau_k,\tau_{(k+1)}}$ is introduced to express the data cache latency in the same node between two adjacent slots $\tau_k, \tau_{(k+1)}$, and the weight of the different nodes between two adjacent is ∞. Therefore, the weights of all virtual edges between two adjacent slots $\tau_k, \tau_{(k+1)}$ can be represented by the weighted matrix $\mathbf{G}^{\tau_k,\tau_{k+1}}$

$$\mathbf{G}^{\tau_k,\tau_{k+1}} = \begin{bmatrix} \pi_1^{\tau_k,\tau_{(k+1)}} & \cdots & \infty & \cdots & \infty \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \infty & \cdots & \pi_i^{\tau_k,\tau_{(k+1)}} & \vdots & \infty \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \infty & \cdots & \infty & \cdots & \pi_M^{\tau_k,\tau_{(k+1)}} \end{bmatrix}, \tag{5}$$

where $\pi_i^{\tau_k,\tau_{(k+1)}}$ can be given by

$$\pi_i^{\tau_k,\tau_{(k+1)}} = \Delta t - t_i^{\tau_k}, \tag{6}$$

where $t_i^{\tau_k}$ is the consumed time of the i th node in the time slot τ_k.

Therefore, as Fig. 2 shows, the WEG of the cybertwin-based IoV network is a static graph, which can be represented by$\{V, N, \Pi, \mathbf{Graph}\}$.

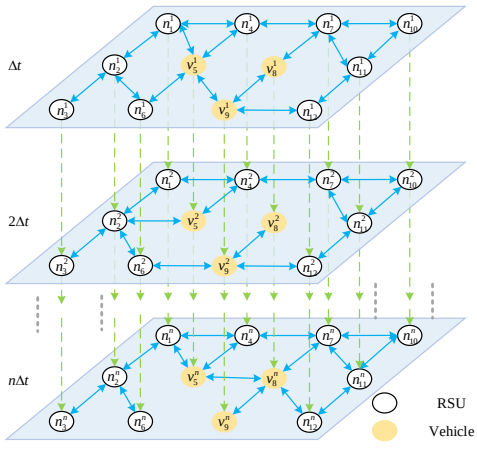

Fig. 2: IoV modeled by WEG.

- $V = \{v_1^{\tau_1}, v_2^{\tau_1}, \cdots, v_M^{\tau_1}; \cdots; v_1^{\tau_r}, v_2^{\tau_r}, \cdots, v_M^{\tau_r}\}$ is the set of cybertwins that represent vehicle nodes. Moreover, the same node in different time slots is regarded as different nodes.
- $N = \{n_1^{\tau_1}, n_2^{\tau_1}, \cdots, n_S^{\tau_1}; \cdots; n_1^{\tau_r}, n_2^{\tau_r}, \cdots, n_S^{\tau_r}\}$ is the set of cybertwins that represent RSUs.
- Π is the edge of the WEG.
- We define $\mathbf{Graph}$ as the matrix of the WEG, which can be given by

$$\mathbf{Graph} = \begin{bmatrix} \mathbf{G}^{\tau_1} & \mathbf{G}^{\tau_1,\tau_2} & \infty & \cdots & \infty & \infty \\ \infty & \mathbf{G}^{\tau_2} & \mathbf{G}^{\tau_2,\tau_3} & \cdots & \infty & \infty \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \infty & \infty & \infty & \cdots & \mathbf{G}^{\tau_{r-1}} & \mathbf{G}^{\tau_{(r-1)},\tau_r} \\ \infty & \infty & \infty & \cdots & \infty & G^{\tau_r} \end{bmatrix}. \tag{7}$$

B. Traffic Routing-Based Computation Offloading

Based on the WEG, traffic routing based computation offloading problem is formulated to offload tasks on the transmission route to avoid huge additional cost brought by service migration. Moreover, we only consider the single user scenario, in fact, for multiple users, it can be decoupled into the independent single-user problem, based on the slicing technology of the network and computation sources [32]. The common assumption can be found in similar works [33], [34].

Without loss of generality, we represent a task as a DAG [35]–[37], $\Phi = (\Omega, \Gamma)$, where Ω=$\{\omega_1, ..., \omega_l\}$ is the set of nodes, Γ is the set of edges. The nodes of the DAG denote the subtasks and the edges denote the execution order among the subtasks. A subtask starts execution only after receiving all necessary input data and the output data is available only after its completion. In $\Phi = (\Omega, \Gamma)$, ω_1 is the single starting subtask, ω_l is the terminal subtask, and other ω_i is the other subtasks. Besides, subtask $\omega_i \in \Omega$ is represented by the tuple $\{D_i, \eta_i, C_i\}$, where D_i represents the input data size, $\eta_i \in (0, 1]$ is the output-input ratio, i.e., the ratio between the output data size and the input data size data, C_i represents the required amount CPU cycles to complete the subtask. Moreover, we define $\varepsilon_i = \frac{C_i}{D_i} cycles/bit$ as the computation complexity of subtask; $\Theta_{\uparrow}(\omega_i) = \{\omega_j \,|(\omega_j, \omega_i) \in \Gamma\}$ as the set of pioneer subtasks of $\omega_i \in \Omega$. Thus, D_i can be given by

$$D_i = \sum_{\omega_j \in \Theta_{\uparrow}(\omega_i)} D_j \eta_j. \tag{8}$$

Then, the task scheduling scheme of task Φ in cybertwin-based IoV can be transformed into DAG to WEG mapping rule.

1) Ω to N Mapping Rules: We define $B : \Omega \to N$ represents the mapping from Ω to N. Then B should meet the conditions shown in Eq.(9). Map the starting subtask ω_1 to the cybertwin that represent task initiator vehicle v_1; since the result receiver and the task initiator is the same cybertwin, map the terminal subtask ω_l to the v_1; map the intermediate subtasks to the cybertwins that represent any other nodes. Moreover, affected by the latency of computing, the task data will be cached to the next time slot. Therefore, for any subtask $\omega_i \in \Omega$, it can be mapped to the χ_i+1 th replica nodes, where

χ_i is the number of slot.

$$B(\omega_i) = \begin{cases} \left\{v_1^1, ..., v_1^{1+\chi_i} \,|1+\chi_i \le r\right\}, \omega_i = \omega_1; \\ \left\{v_1^q, ..., v_1^{q+\chi_i} \,|1 \le q, q+\chi_i \le r\right\}, \omega_i = \omega_l; \\ \left\{n_p^q, ..., n_p^{q+\chi_i} \,|1 \le q, q+\chi_i \le r, 1 \le p \le S\right\}, \text{otherwise.} \end{cases} \tag{9}$$

2) Γ to Π Mapping Rules: We define $Z : \Gamma \rightarrow \Pi$ represents the mapping from Γ to Π. For the directed edge $(\omega_i, \omega_j) \in \Gamma$ of DAG, Z maps the (ω_i, ω_j) to the shortest path $Path_{B(\omega_i)B(\omega_j)}$ in WEG.

$$Z(\omega_i, \omega_j) = Path_{B(\omega_i)B(\omega_j)}. \tag{10}$$

3) Latency Model: Based on the result of node mapping and edge mapping, the computing latency and transmission latency can be given by

$$T_{comp}(\omega_i) = \frac{D_i \eta_i \varepsilon_i}{\rho_{B(\omega_i)}}, \tag{11}$$

$$T_{comm}(\omega_i) = d_{B(\omega_j)B(\omega_i)} D_j \eta_j, \tag{12}$$

where $T_{comp}(\omega_i)$ is the computing delay of subtask ω_i and can be calculated by Eq. (11), $T_{comm}(\omega_i)$ is the communication delay of subtask ω_i and can be calculated by Eq. (12), $\rho_{B(\omega_i)}$ is the computing capacity of $B(\omega_i)$, $d_{B(\omega_j)B(\omega_i)}$ is the transmission delay of $\forall \omega_j \in \Phi_\uparrow(\omega_i)$ transmitting unit bit to ω_i along the shortest path $Path_{B(\omega_i)B(\omega_j)}$.

Therefore, the processing delay of subtask ω_i can be given by

$$T(\omega_i) = T_{comp}(\omega_i) + T_{accu}(\omega_i), \tag{13}$$

where $T_{accu}(\omega_i)$ is the accumulated delay and can be calculated by Eq. (14).

$$T_{accu}(\omega_i) = \max_{\omega_j \in \phi_\uparrow(\omega_i)} [T(\omega_j) + T_{comm}]. \tag{14}$$

Base on Eq. (11)-(14), the processing delay of subtask ω_i can be given by

$$T(\omega_i) = \max_{\omega_j \in \phi_\uparrow(\omega_i)} [T(\omega_j) + d_{B(\omega_j)B(\omega_i)} D_j \eta_j] + \frac{D_i \eta_i \varepsilon_i}{\rho_{B(\omega_i)}}. \tag{15}$$

C. Problem Formulation

Based on the latency model, the total processing delay of task Φ is the completion time of the last subtask ω_l and can be represented as

$$T(\Phi) = T(\omega_l) = \max_{\omega_i \in \phi_\uparrow(\omega_l)} [T(\omega_i) + d_{B(\omega_i)B(\omega_l)} D_i \eta_i] + \frac{D_i \eta_i \varepsilon_i}{\rho_{B(\omega_l)}}. \tag{16}$$

Moreover, for the same task Φ, there are multiple mapping results satisfy the mapping rules, resulting in different processing delays. Therefore, minimizing processing delay is to find the mapping results with the lowest processing delay. To illustrate the assignment of tasks, we define a binary decision matrix $\mathbf{X}$ with l rows $S \times \tau_r$ columns, which can be given by

$$\mathbf{X} = \begin{bmatrix} x(\omega_1, n_1^1) & \cdots & x(\omega_1, n_S^1) & \cdots & x(\omega_1, n_S^{\tau_r}) \\ x(\omega_2, n_1^1) & \cdots & x(\omega_2, n_S^1) & \cdots & x(\omega_2, n_S^{\tau_r}) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x(\omega_l, n_1^1) & \cdots & x(\omega_l, n_S^1) & \cdots & x(\omega_l, n_S^{\tau_r}) \end{bmatrix}, \tag{17}$$

where $x(\omega_i, n_p^q) \in \mathbf{X}$ represents whether ω_i is mapping to the n_p^q. If $x(\omega_i, n_p^q)$=1, ω_i is mapping to the n_p^q. Or ω_i is not mapping to the n_p^q. Besides, based on the Eq. (16), the subtask ω_i is mapped to $\chi_i + 1$ nodes of **Graph**. Therefore, the $x(\omega_i, n_p^q)$ should subject to the following constraints

$$x(\omega_i, n_p^q) \in \{0, 1\}, \ \forall \omega_i \in \Omega, \forall u_p^q \in U, q \in \{\tau_1, \tau_2, \cdots, \tau_r\}, \tag{18}$$

$$\sum_{u_p^q \in U} x(\omega_i, u_p^q) = \chi_i + 1, \forall \omega_i \in \Omega. \tag{19}$$

Therefore, the total processing delay of task Φ can be given by

$$T(\mathbf{X}) = T_{accu}(\omega_l) + \sum_{n_d^w \in N} \frac{D_i \eta_i \varepsilon_i}{\rho_{n_d^w}} x(\omega_l, n_d^w). \tag{20}$$

According to the analysis above, the delay optimization problem can be represented as

$$\begin{aligned} &\mathbf{X} = \arg\min (T(\mathbf{X})) \\ &\text{s.t. : } (18), (19). \end{aligned} \tag{21}$$

D. eHEFT Algorithm

The HEFT algorithm was first proposed by Topcuoglu *et al.* in [38], which verified that HEFT algorithm performs better than other DAG scheduling algorithms (e.g., Dynamic Level Scheduling (DLS) algorithm, Levelized-Min Time (LMT) algorithm, etc.). Thus, due to its low complexity and high performance, the HEFT algorithm has been widely adopted in DAG task scheduling.

As shown in Fig.3, the HEFT algorithm can be divided into two phases: task prioritizing phase and node selection phase. In the task prioritizing phase, the subtasks are prioritized according to a specific metric. Then, in the node selection phase, the node is selected based on the earliest finish time. However, the HEFT algorithm tends to offload tasks in a greedy manner, which will result in the task nodes opposite to the vehicle's movement direction and latency performance degradation. Moreover, the HEFT algorithm is only suitable for static networks. Therefore, in this article, we propose an eHEFT algorithm by introducing gradient routing into the traditional HEFT algorithm to improve latency performance and designing a cross-slot mechanism for dynamic network topology.

Same as HEFT, in the eHEFT, the scheduling process can be divided into two phases: task prioritizing phase and cybertwin selection phase.

In the task prioritizing phase, the subtasks are ordered by their scheduling priorities based on the upward ranking. The upward rank of subtask ω_i can be given by

$$rank_u(\omega_i) = \overline{w_i} + \max_{\omega_j \in succ(\omega_i)} (\overline{c_{i,j}} + rank_u(\omega_j)), \tag{22}$$

where $succ(\omega_i)$ is the set of immediate successors of subtask ω_i, $\overline{c_{i,j}}$ is the average communication cost of edge (i, j), and $\overline{w_i}$ is the average computation cost of subtask ω_i. $\overline{c_{i,j}}$ and $\overline{w_i}$ can be obtain by Eq. (23), Eq. (24) respectively.

$$\overline{w_i} = \sum_{j=1}^{q} w_{i,j}/q, \tag{23}$$

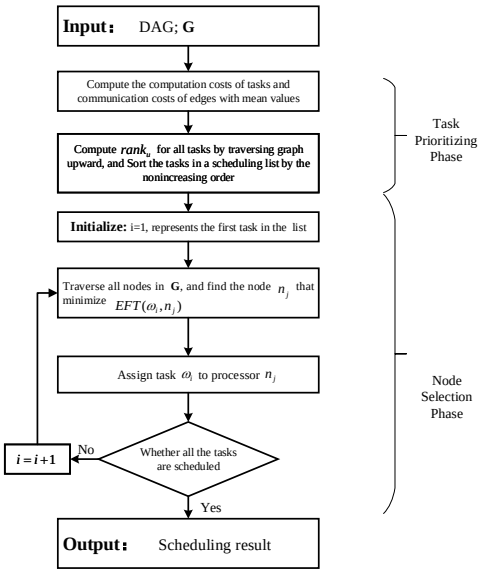

Fig. 3: The flowchart of HEFT algorithm.

$$\overline{c_{i,k}} = \overline{L} + \frac{data_{i,k}}{\overline{B}}, \tag{24}$$

where $w_{i,j}$ is the execution time to complete subtask ω_i on processor n_j, q is the number of the network node, $data_{i,k}$ is the communication data size of subtask ω_i and ω_j, $\overline{B}$ is the average transfer rate among the processors in the domain and $\overline{L}$ is the average communication startup time, here, we set $\overline{L} = 0$.

Since the rank is computed recursively by traversing the task graph upward, starting from the exit task, it is called upward rank. For the exit task , the upward rank value can be represented as

$$rank_u(\omega_{exit}) = \overline{w_{exit}}. \tag{25}$$

In the cybertwin selection phase, the gradient is calculated based on the hops from the cybertwin that represents vehicle to the cybertwins that represent other network nodes, and the subtask is allocated to the edge node on the data transmission route with the help of gradient.

Moreover, we define $H_i^{\tau_k}$ as the gradient of the network node n_i at τ_kth slot, and $H_i^{\tau_k}$ can be obtained by the Dijkstra algorithm. $EFT(\omega_i, n_j)$ is the Earliest Finish Time (EFT) of task ω_i on the node n_j, which can be calculated by Eq. (21).

IV. Simulation Results

To validate the effectiveness of our proposed scheme, extensive experiments are presented. Fig. 4 shows four different DAG task models with different degrees of parallelism. We adopt the distribution of base stations in a region of Yanta District, Xi'an, China, and only consider one vehicle scenario. Based on the distribution of base stations and literature [39], the network topology is shown in Fig. 5, where the vehicle's position in the different time slot is marked and the link is divided into four different types (i.e., 5G, 4G, Internet, Wi-Fi). According to literatures [13], [39], the parameters are summarized in TABLE I, in which ρ_c is the computation capability of cloud center, ρ_0 is the computation

Algorithm 1 eHEFT algorithm

Input: DAG; $\mathbf{G}^{\tau_1}, \mathbf{G}^{\tau_2}, \cdots, \mathbf{G}^{\tau_r}$

1: Compute the computation costs of tasks and communication costs of edges with mean values;
2: Compute $rank_u$ for all tasks by traversing graph upward, starting from the exit task;
3: Sort the tasks in a scheduling list by nonincreasing order of $rank_u$ values;
4: Initialize: $k = 1$;
5: Compute the gradient H of each node in $\mathbf{G}^{\tau_k}$;
6: **while** there are unscheduled tasks in the list **do**
7: Select the first task ω_i in the list for scheduling;
8: **for** each node n_j in the network **do**
9: Compute $EFT(\omega_i, n_j)$ value;
10: **end for**
11: Find the processor n_j that minimizes $EFT(\omega_i, n_j)$. $H_i^{\tau_k}$ of task ω_i;
12: **if** $EFT(\omega_i, n_j) <= \Delta t$ **then**
13: Assign task ω_i to processor n_j;
14: **else**
15: $k = k + 1$;
16: Go back to step 5;
17: **end if**
18: **end while**

Output: Scheduling result;

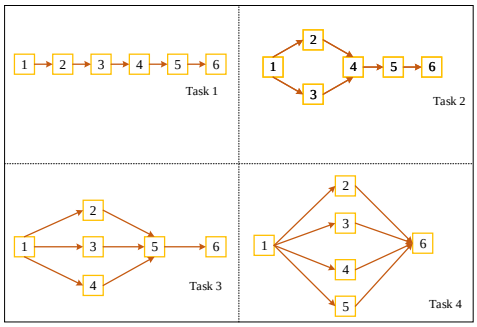

Fig. 4: Different task models.

capability of the vehicle, B_{ci} is the bandwidth between the cloud center and the edge nodes. The simulation is run on the Python-based simulator in this paper. Moreover, the numerical results are based on 3000 Monte Carlo simulations.

Fig. 6 shows the influences of the input data size on the latency performance of different schemes (i.e., cloud computing, local computing, service migration, and proposed scheme) under different task models, which is shown in Fig. 4. And we set the output-input ratio as 1. As we can see, with the increase of the input data size, the latency of each schemes increases because more input data brings more computation latency and communicating latency. Furthermore, the proposed scheme performs better than other schemes and the cloud computing curve is intensely higher than the others. The reason is that the transmission latency of cloud computing architecture is growing linearly with the increase of the input data size while the transmission latency of the

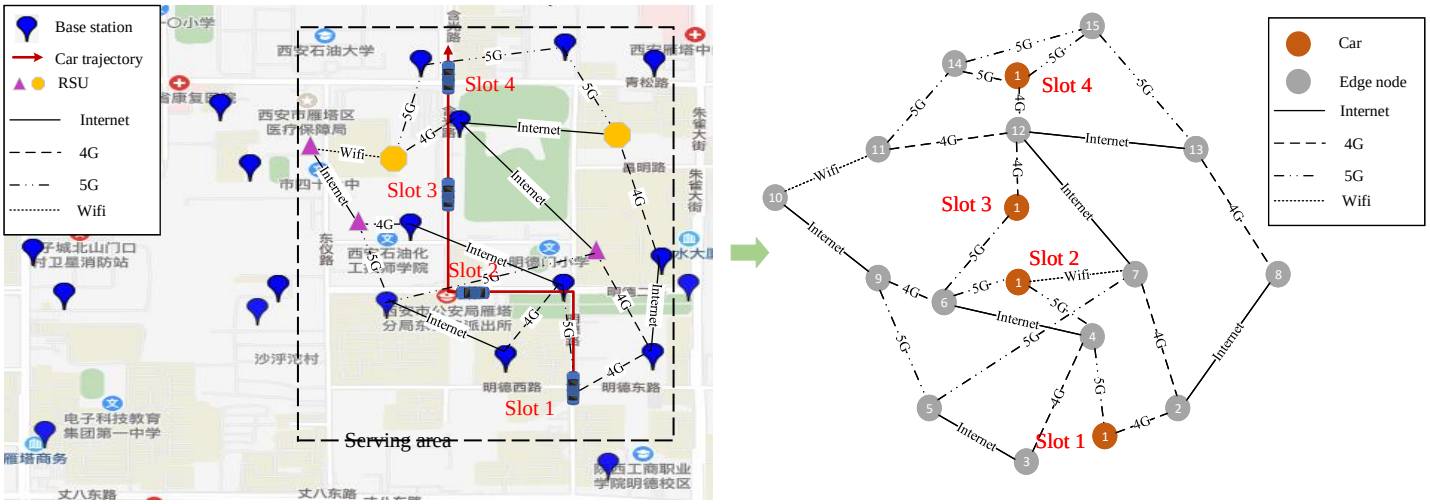

Fig. 5: Network topology.

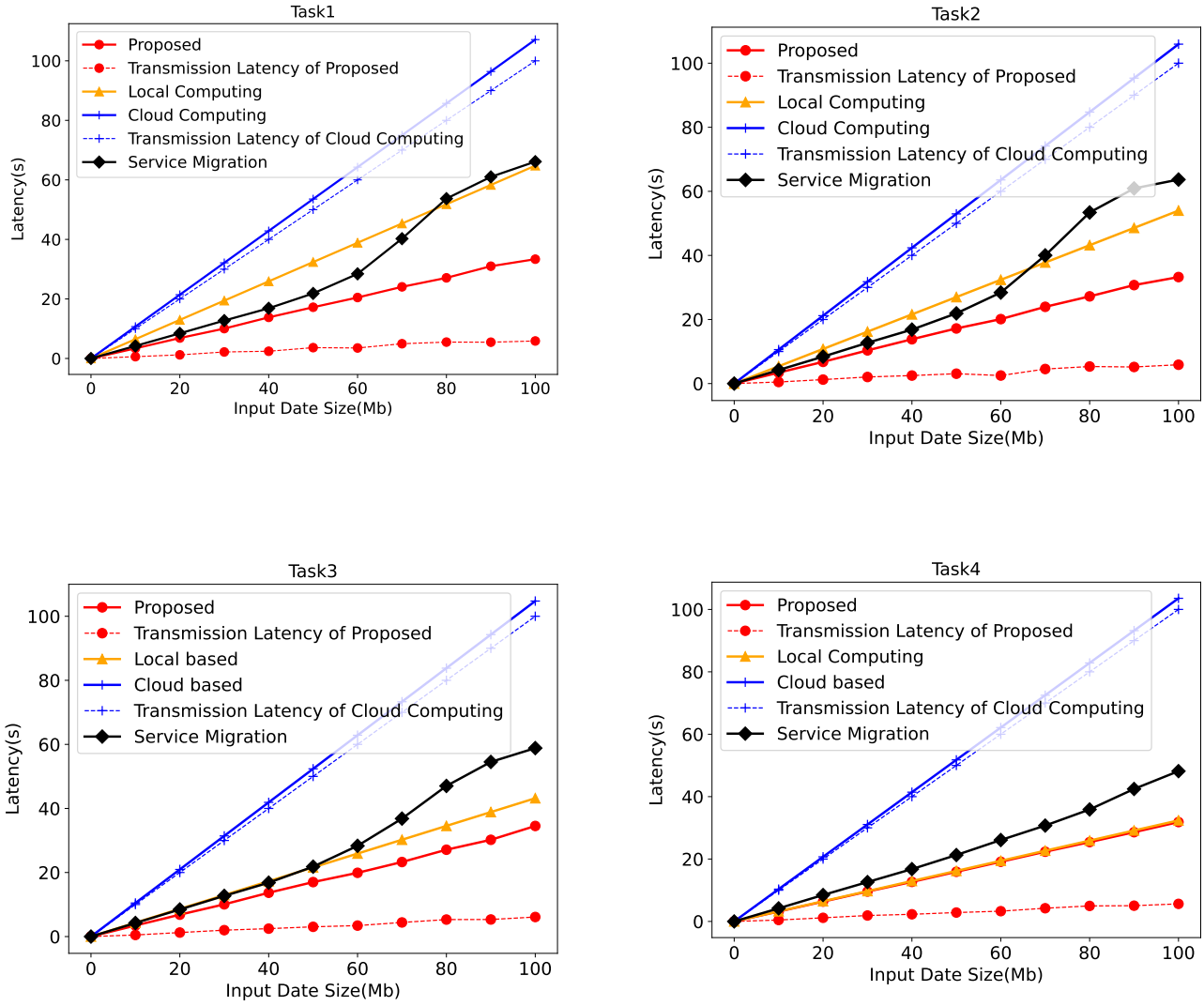

Fig. 6: Latency performance of different schemes versus input data size under different task models.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVT.2021.3134715, IEEE Transactions on Vehicular Technology

TABLE I: System Parameters of IoV

Parameter	Value
Internet	[34.68,100] Mbps
4G	[1.11,17.02] Mbps
5G	[400,1000] Mbps
Wi-Fi	[7.64,29.67] Mbps
B_{ci}	2 Mbps
ρ_c	20 GHz
ρ_0	2200 MHz
ρ_i	[5000,7000] MHz
ε_i	$1900/8$ cycles/bit
Δt	20 s

proposed scheme is much lower than cloud computing. In addition, with the input data size increasing, the latency of service migration is first close to the proposed scheme, then grows rapidly and much higher than the proposed scheme. This is because that the service can be processed within a short time and does not need to be migrated when the input data size is small, and if the input data size is large, the edge node cannot accomplish the task before the vehicle moves into a new area, so the service has to be migrated and brings huge additional cost.

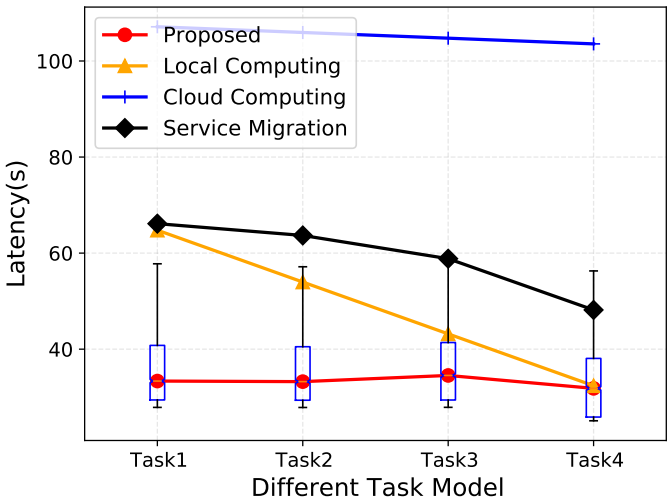

Fig. 7: Latency performance of different schemes versus different task models.

Fig. 7 shows the latency performance of different schemes versus different task models, where the input data size is 100 Mb. It can be seen that the latency cloud computing is always higher than others and the latency of the proposed scheme with eHEFT algorithm keeps steady under different task models, while the service migration and the local computing decreases with the increase of the parallel steps of the tasks. This is because that the eHEFT algorithm transforms any task model into a sequential task. Therefore, the eHEFT algorithm has better adaptability to different task models.

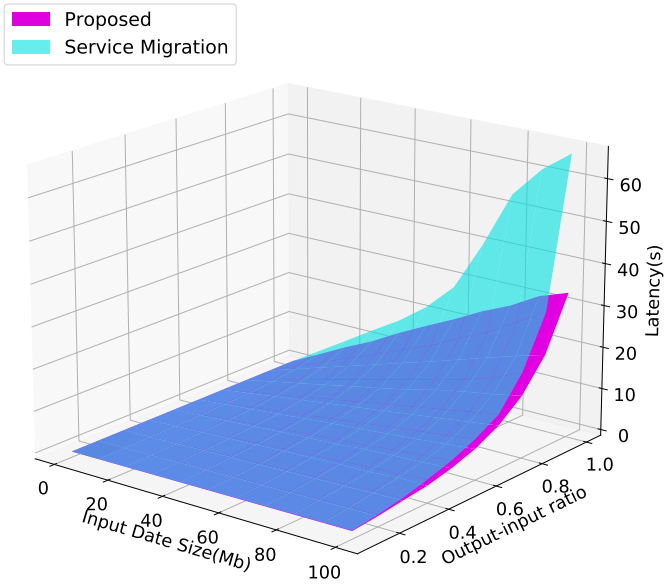

Fig. 8: Latency performance of different output-input ratio.

Fig. 8 shows the latency performance of different output-input ratios, where the task model is the sequential task. As we can see, when the input data size and the output-input ratio are small, the latency of service migration is close to the proposed scheme, this is because that the smaller the input data size and the output-input ratio, the smaller the task processing time. Moreover, when the input data size is fixed, the difference between the service migration and the proposed scheme increases, and the latency of the service migration and the proposed scheme grows exponentially with the increase of the output-input ratio. This is because that the larger the output-input ratio, the smaller the reduction of the data size after each processing. Therefore, the proposed scheme shows better latency performance when processing huge data volume, large output-input ratio services. Moreover, based on the conclusion of Fig. 7, the trends of Fig. 8 is suitable for any task model.

Fig. 9 shows the latency performance of the eHEFT algorithm by comparing it with HEFT algorithm and other typical load balancing algorithms (i.e., Weighted Round Robin (WRR), Greedy Load Balance (Greddy-LB), Pick-KX) versus input data size under different task models. We can observe that the eHEFT algorithm is superior to the other algorithms under different task models. The reason is that the WRR and Greedy-LB algorithms only consider the computing capabilities of the drones but ignore the impact of the capabilities of transmission links, which may result in the mismatch between the optimal solution with the real situation. Although the HEFT algorithm jointly considers computing and communication capabilities, it may offload tasks on the edge nodes that are opposite to the vehicle's movement direction. In a contrast, the eHEFT algorithm jointly considers computing and communication capabilities with gradient routing and achieves relatively high latency performance all along. When the task model is task 1 and input data size is 100 Mb, we can observe that the latency performance of the eHEFT algorithm improved by 16.7 %, 47.6 %, 55.9 %, 59.2 % compared with the HEFT, Greddy-LB, WRR, and Pick-KX algorithm respectively.

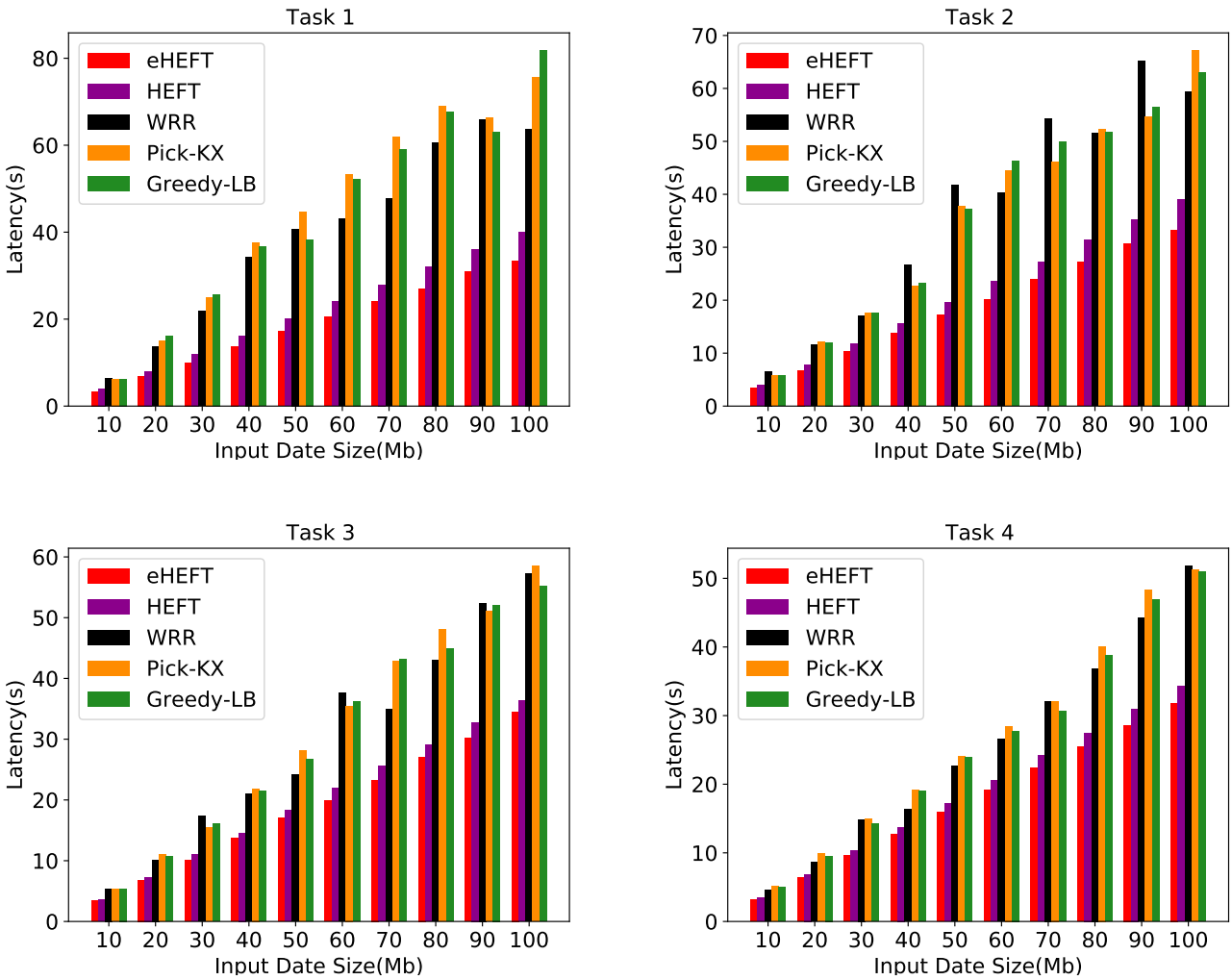

Fig. 9: Latency performance comparison of different algorithms versus input data size under different task models.

V. Conclusion

In this paper, we studied the traffic routing-based computation offloading problem in cybertwin-driven IoV for V2X applications. To support latency-sensitive V2X applications, we extended the conception of the cybertwin, which can be the digital representation of the network hardware devices and the network software functions. Our target is to improve the latency performance of V2X applications. To achieve it, we formulated the optimization problem considering computation offloading and multi-hop routing problems. To solve the optimization problem formulated, we designed eHEFT algorithm, which introduced the gradient routing into the traditional HEFT algorithm to guarantee the consistency between the data transmission direction and the vehicle's movement direction. In future work, we plan to design a decentralized algorithm that only needs to maintain its neighbor's information instead of obtaining global information.

References

[1] C. Yu, B. Lin, P. Guo, W. Zhang, S. Li and R. He, "Deployment and Dimensioning of Fog Computing-Based Internet of Vehicle Infrastructure for Autonomous Driving," in *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 149-160, Feb. 2019.

[2] W. Cheng, Y. Xiao, S. Zhang and J. Wang, "Adaptive Finite Blocklength for Ultra-Low Latency in Wireless Communications," in *IEEE Transactions on Wireless Communications*, doi: 10.1109/TWC.2021.3130269.

[3] X. Qiu, L. Liu, W. Chen, Z. Hong and Z. Zheng, "Online Deep Reinforcement Learning for Computation Offloading in Blockchain-Empowered Mobile Edge Computing," in *IEEE Transactions on Vehicular Technology*, vol. 68, no. 8, pp. 8050-8062, Aug. 2019.

[4] K. Xiong, S. Leng, C. Huang, C. Yuen and Y. L. Guan, "Intelligent Task Offloading for Heterogeneous V2X Communications," in *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 4, pp. 2226-2238, April 2021.

[5] R. -H. Hwang, M. M. Islam, M. A. Tanvir, M. S. Hossain and Y. -D. Lin, "Communication and Computation Offloading for 5G V2X: Modeling and Optimization," *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, 2020, pp. 1-6.

[6] M. R. Anwar, S. Wang, M. F. Akram, S. Raza and S. Mahmood, "5G Enabled MEC: A Distributed Traffic Steering for Seamless Service Migration of Internet of Vehicles," in *IEEE Internet of Things Journal*, doi: 10.1109/JIOT.2021.3084912.

[7] I. Labriji et al., "Mobility Aware and Dynamic Migration of MEC Services for the Internet of Vehicles," in *IEEE Transactions on Network and Service Management*, vol. 18, no. 1, pp. 570-584, March 2021.

[8] J. Xu, X. Ma, A. Zhou, Q. Duan and S. Wang, "Path Selection for Seamless Service Migration in Vehicular Edge Computing," in *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 9040-9049, Sept. 2020.

[9] M. Li, J. Gao, L. Zhao and X. Shen, "Deep Reinforcement Learning for Collaborative Edge Computing in Vehicular Networks," in *IEEE Transactions on Cognitive Communications and Networking*, vol. 6, no. 4, pp. 1122-1135, Dec. 2020.

[10] Q. Yu, J. Ren, Y. Fu, Y. Li and W. Zhang, "Cybertwin: An Origin of Next Generation Network Architecture," in *IEEE Wireless Communications*, vol. 26, no. 6, pp. 111-117, December 2019.
[11] B. Chang and J. Chiou, "Cloud Computing-Based Analyses to Predict Vehicle Driving Shockwave for Active Safe Driving in Intelligent Transportation System," in *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 2, pp. 852-866, Feb. 2020.
[12] Ashok. A, Steenkiste. P, Bai. F, "Vehicular Cloud Computing through Dynamic Computation Offloading," in *Computer Communications*, vol. 120, pp. 125ÍC137, May. 2018.
[13] X. Hou et al., "Reliable Computation Offloading for Edge-Computing-Enabled Software-Defined IoV," in *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7097-7111, Aug. 2020.
[14] Q. Luo, C. Li, T. H. Luan, W. Shi and W. Wu, "Self-Learning based Computation Offloading for Internet of Vehicles: Model and Algorithm," in *IEEE Transactions on Wireless Communications*, doi: 10.1109/TWC.2021.3071248.
[15] P. Zhou et al., "Edge-Facilitated Augmented Vision in Vehicle-to-Everything Networks," in *IEEE Transactions on Vehicular Technology*, vol. 69, no. 10, pp. 12187-12201, Oct. 2020.
[16] A. Moubayed, A. Shami, P. Heidari, A. Larabi and R. Brunner, "Edge-Enabled V2X Service Placement for Intelligent Transportation Systems," in *IEEE Transactions on Mobile Computing*, vol. 20, no. 4, pp. 1380-1392, 1 April 2021.
[17] S. Wang, R. Urgaonkar, M. Zafer, T. He, K. Chan and K. K. Leung, "Dynamic Service Migration in Mobile Edge Computing Based on Markov Decision Process," in *IEEE/ACM Transactions on Networking*, vol. 27, no. 3, pp. 1272-1288, June 2019.
[18] H. Ke, J. Wang, L. Deng, Y. Ge and H. Wang, "Deep Reinforcement Learning-Based Adaptive Computation Offloading for MEC in Heterogeneous Vehicular Networks," in *IEEE Transactions on Vehicular Technology*, vol. 69, no. 7, pp. 7916-7929, July 2020.
[19] Z. Jia, Z. Zhou, X. Wang and S. Mumtaz, "Learning-Based Queuing Delay-Aware Task Offloading in Collaborative Vehicular Networks," *ICC 2021 - IEEE International Conference on Communications*, 2021, pp. 1-6.
[20] X. Zhu, Y. Luo, A. Liu, M. Z. A. Bhuiyan and S. Zhang, "Multiagent Deep Reinforcement Learning for Vehicular Computation Offloading in IoT," in *IEEE Internet of Things Journal*, vol. 8, no. 12, pp. 9763-9773, 15 June15, 2021.
[21] W. Liu and Y. Shoji, "Edge-Assisted Vehicle Mobility Prediction to Support V2X Communications," in *IEEE Transactions on Vehicular Technology*, vol. 68, no. 10, pp. 10227-10238, Oct. 2019.
[22] P. L. Nguyen, R. -H. Hwang, P. M. Khiem, K. Nguyen and Y. -D. Lin, "Modeling and Minimizing Latency in Three-tier V2X Networks," *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, 2020, pp. 1-6.
[23] W. Cheng, W. Zhang, H. Jing, S. Gao and H. Zhang, "Orbital Angular Momentum for Wireless Communications," in *IEEE Wireless Communications*, vol. 26, no. 1, pp. 100-107, February 2019.
[24] R. Lyu, W. Cheng and W. Zhang, "Modeling and Performance Analysis of OAM-NFC Systems," in *IEEE Transactions on Communications*, doi: 10.1109/TCOMM.2021.3110871.
[25] L. Yao, J. Wang, X. Wang, A. Chen and Y. Wang, "V2X Routing in a VANET Based on the Hidden Markov Model," in *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 3, pp. 889-899, March 2018.
[26] S. KÍźhlmorgen, H. Lu, A. Festag, J. Kenney, S. Gemsheim and G. Fettweis, "Evaluation of Congestion-Enabled Forwarding With Mixed Data Traffic in Vehicular Communications," in *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 1, pp. 233-247, Jan. 2020.
[27] L. Luo, L. Sheng, H. Yu and G. Sun, "Intersection-Based V2X Routing via Reinforcement Learning in Vehicular Ad Hoc Networks," in *IEEE Transactions on Intelligent Transportation Systems*, doi: 10.1109/TITS.2021.3053958.
[28] A. Ksentini, T. Taleb and M. Chen, "A Markov Decision Process-based service migration procedure for follow me cloud," *2014 IEEE International Conference on Communications (ICC)*, 2014, pp. 1350-1354.
[29] Q. Yu, J. Ren, H. Zhou and W. Zhang, "A Cybertwin based Network Architecture for 6G," *2020 2nd 6G Wireless Summit (6G SUMMIT)*, 2020, pp. 1-5.
[30] P. Wang, X. Zhang, S. Zhang, H. Li and T. Zhang, "Time-Expanded Graph-Based Resource Allocation Over the Satellite Networks," in *IEEE Wireless Communications Letters*, vol. 8, no. 2, pp. 360-363, April 2019.
[31] W. Cheng, X. Zhang and H. Zhang, "Optimal Power Allocation With Statistical QoS Provisioning for D2D and Cellular Communications Over Underlaying Wireless Networks," in *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 1, pp. 151-162, Jan. 2016.
[32] T. Q. Dinh, J. Tang, Q. D. La and T. Q. S. Quek, "Offloading in Mobile Edge Computing: Task Allocation and Computational Frequency Scaling," in *IEEE Transactions on Communications*, vol. 65, no. 8, pp. 3571-3584, Aug. 2017.
[33] O. Munoz, A. Pascual-Iserte and J. Vidal, "Optimization of Radio and Computational Resources for Energy Efficiency in Latency-Constrained Application Offloading," in *IEEE Transactions on Vehicular Technology*, vol. 64, no. 10, pp. 4738-4755, Oct. 2015.
[34] Y. Wang, M. Sheng, X. Wang, L. Wang and J. Li, "Mobile-Edge Computing: Partial Computation Offloading Using Dynamic Voltage Scaling," in *IEEE Transactions on Communications*, vol. 64, no. 10, pp. 4268-4282, Oct. 2016.
[35] V. Shah, B. K. Dey and D. Manjunath, "Network Flows for Function Computation," in *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 4, pp. 714-730, April 2013.
[36] M. Rost and S. Schmid, "Virtual Network Embedding Approximations: Leveraging Randomized Rounding," in *IEEE/ACM Transactions on Networking*, vol. 27, no. 5, pp. 2071-2084, Oct. 2019.
[37] M. Michael, J. Llorca and A. Tulino, "Approximation Algorithms for the Optimal Distribution of Real-Time Stream-Processing Services," *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, 2019, pp. 1-7.
[38] H. Topcuoglu, S. Hariri and Min-You Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," in *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, no. 3, pp. 260-274, March 2002.
[39] Wang. H, Li. Y, Zhou. A, Guo. Y, Wang. S, "Service migration in mobile edge computing: A deep reinforcement learning approach," in *International Journal of Communication Systems*, doi:10.1002/dac.4413.

Buyun Ma (Student Member, IEEE) received the B.S. degree (Highest Hons.) in communication engineering from the Jilin University, Changchun, China, in 2020. He is currently pursuing the M.S. degree with the State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an, China.

His current research interests include dynamic network computing and distributed computing. He has served as a reviewer of IEEE ICC, IEEE GLOBECOM, and IEEE ACCESS.

Zhiyuan Ren (Member, IEEE) received the M.S. and Ph.D. degrees in communication and information system from Xidian University, Xi'an, China, in 2007 and 2011, respectively.

He is an Associate Professor with the School of Telecommunication Engineering, Xidian University. He has published more than 20 journal publications and conference publications. His major research interests are distributed computing, mobile-edge computing, and network virtualization.

Wenchi Cheng (Senior Member, IEEE) received the B.S. and Ph.D. degrees in telecommunication engineering from Xidian University, Xi'an, China, in 2008 and 2013, respectively.

He is a Professor with Xidian University. He was a Visiting Scholar with the Networking and Information Systems Laboratory, Department of Electrical and Computer Engineering, Texas A&M University, College Station, TX, USA, from 2010 to 2011. He has published more than 80 international journal and conference papers in the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, IEEE Magazines, IEEE INFOCOM, GLOBECOM, and ICC. His current research interests include 5G wireless networks and orbital-angular-momentum-based wireless communications.