Intelligent Edge Computing in Internet of Vehicles: A Joint Computation Offloading and Caching Solution

Zhaolong Ning, Kaiyuan Zhang, Xiaojie Wang, Lei Guo, Xiping Hu, Jun Huang, Bin Hu, Ricky Y. K. Kwok, *Fellow, IEEE*

***Abstract*—Recently, Internet of Vehicles (IoV) has become one of the most active research fields in both academic and industry, which exploits resources of vehicles and Road Side Units (RSUs) to execute various vehicular applications. Due to the increasing number of vehicles and the asymmetrical distribution of traffic flows, it is essential for the network operator to design intelligent offloading strategies to improve network performance and provide high-quality services for users. However, the lack of global information and the time-variety of IoVs make it challenging to perform effective offloading and caching decisions under long-term energy constraints of RSUs. Since Artificial Intelligence (AI) and machine learning can greatly enhance the intelligence and the performance of IoVs, we push AI inspired computing, caching and communication resources to the proximity of smart vehicles, which jointly enable RSU peer offloading, vehicle-to-RSU offloading and content caching in the IoV framework. A Mix Integer Non-Linear Programming (MINLP) problem is formulated to minimize total network delay, consisting of communication delay, computation delay, network congestion delay and content downloading delay of all users. Then, we develop an online multi-decision making scheme (named OMEN) by leveraging Lyapunov optimization method to solve the formulated problem, and prove that OMEN achieves near-optimal performance. Leveraging strong cognition of AI, we put forward an imitation learning enabled branch-and-bound solution in edge intelligent IoVs to speed up the problem solving process with few training samples. Experimental results based on real-world traffic data demonstrate that our proposed method outperforms other methods from various aspects.**

***Index Terms*—Internet of Vehicles, peer offloading, content caching, Lyapunov optimization, imitation learning.**

Z. Ning is with the Chongqing Key Laboratory of Mobile Communications Technology, Chongqing University of Posts and Telecommunications, Chongqing, China; National Mobile Communications Research Laboratory, Southeast University, Nanjing, China; and School of Software, Dalian University of Technology, Dalian, China. Email: z.ning@ieee.org.

K. Zhang is with the School of Software, Dalian University of Technology, Dalian, China. Email: zky123123@live.com.

X. Wang is with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong, China. Email: xiaojie.wang@polyu.edu.hk.

L. Guo (Corresponding author) is with the School of Communication and Information Engineering, Chongqing University of Posts and Telecommunications, Chongqing, China. Email: guolei@cqupt.edu.cn.

J. Huang is with the School of Computer Science and Technology, Chongqing University of Posts and Telecommunications, Chongqing, China. Email: jhuang@cqupt.edu.cn.

X. Hu is with the Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China. Email: xp.hu@siat.ac.cn.

B. Hu is with the School of Information Science and Engineering, Lanzhou University, Lanzhou, China. Email: bh@lzu.edu.cn.

R. Kwok is with the Department of Electrical and Electronic Engineering, University of Hong Kong, Hong Kong, China. Email: Ricky.Kwok@hku.hk.

I. Introduction

The advancements in 5G communication networks and Internet of Vehicles (IoV) enable rapid development of many vehicular applications, such as route planning, video compression and AR navigation, which provide comfortable travel experiences for drivers and passengers. However, individual vehicles own limited computational resources and storage capabilities to deal with applications within their strict delay constraints [1]. Nowadays, edge intelligence, integrating Artificial Intelligence (AI) with edge computing, enables appropriate edge service deployment and flexible resource scheduling in IoVs [2]. Edge intelligence is a promising techinique for IoVs, which can alleviate the bandwidth of backhaul links and provide computing as well as caching services with low-latency. Leveraging the cognitive capability of edge intelligence, the edge intelligent IoV framework can realize efficient processing for mission-critical applications, low latency content delivery for interactive entertainments and smart transmission for QoS-aware information [3].

Road Side Units (RSUs) equipped with edge servers are able to serve vehicles within their coverages, which alleviate computing and backhaul pressures of the cloud server [4]. Computational tasks exceeding the computational capability of edge servers can be offloaded to the cloud server. Hence, a three-layer offloading architecture based on vehicles, RSUs and the cloud server is constructed for IoVs. Although computation offloading has attracted great attention in recent works, it generally ignores the cooperation among edge servers [5]. Due to the asymmetrical distribution of traffic flows in real world, it is essential for the network operator to intelligently dispatch workloads from overloaded RSUs to nearby underused ones, which both enhances network performance and provides high-quality services for users.

Energy consumption of IoV entities is meaningful for stakeholders, which attracts a lot of attention from all parties [6]. Many famous automakers around the world are developing electric vehicles to reduce dependence on non-renewable resources. Government agencies are putting forward new standards to construct green cities in 5G era, which motivates network operators to make use of limited energy effectively. According to the recent analysis [7], the transportation sector accounts for 32% of the overall CO_2 emission. Given the limited resources, mobile network operators need to utilize idle resources in the network to enhance network performance

and improve resource utilization efficiency. Thus, an energy-efficient resource allocation scheme is necessary for both network operators and users.

Recently, AI and machine learning become emerging techniques to address edge service deployment and resource scheduling issues [8], [9]. As an important branch of AI, Deep Reinforcement Learning (DRL) based solutions emerge as promising solutions in many fields. Owning the cognitive capability for dynamic environments, DRL empowered IoV has a superiority in accuracy and efficiency when solving decision making and resource allocation problems [10]. However, most of them need sufficient training data to guarantee their performance [11], [12], and many of them pursue good performance regardless of time complexity. Therefore, imitation learning [13] is proposed as a lightweight framework for real-time processing, which can achieve near-optimal performance with a few training samples.

There are three major challenges for edge computing in IoVs:

- First, the stochastic traffic flow causes uneven workload arrivals in IoVs, which consumes heterogeneous energy resources for RSUs with certain energy constraints. Meanwhile, network operators are hard to obtain future information in IoVs, which motivates an online offloading and caching decision across different time slots.
- Second, vehicular applications are heterogeneous in terms of their different popularity and demands. To satisfy the delay constraints for latency-sensitive tasks, jointly computation offloading and content caching should be considered.
- Finally, existing AI methods usually require adequate training samples to guarantee their learning performance. However, sometimes it is difficult to obtained enough data from users. Thus, it is significant to propose a novel learning framework to obtain the optimal solution with a few training samples.

In this paper, we construct an edge intelligence empowered IoV framework, which utilizes an imitation learning-enabled solution for optimal computation offloading and content caching. We first propose a novel online algorithm, by leveraging Lyapunov optimization, to reduce the total network delay with current information. To deal with the time-varying character of IoVs, we design an imitation learning enabled Branch-and-Bound algorithm (B&B), which combines AI with the traditional multi-objective optimization algorithm to obtain the optimal decision with a few training samples. The contributions are summarized as follows:

- We construct a hierarchical architecture for edge intelligence empowered IoV, which jointly considers vehicle-to-RSU computation offloading, RSU peer offloading, and content caching. Then, a Mixed Integer Non-Linear Programming (MINLP) optimization problem is formulated to minimize the total network delay.
- We propose an online multi-decision making algorithm (named OMEN) by leveraging Lyapunov optimization, which works in an online manner without requiring future system information. Satisfying the long-term energy constraints, we theoretically prove that OMEN approaches the optimal performance within a bounded deviation. Then we mathematically characterize the roles of different RSUs and determine the optimal peer offloading strategy based on Lagrange multipliers method.
- We develop an efficient imitation learning based B&B algorithm, which accelerates the solving process via a few training samples. As the pruning process in B&B can be formulated as a sequential decision problem, we adopt a novel machine learning method, i.e., Data Aggregation, to learn optimal pruning policy, which achieves outstanding learning performance with a few training samples.
- We conduct experiments based on real-world traffic data in Hangzhou, China. Performance results demonstrate that our proposed method significantly reduces the network delay with different traffic flows. Moreover, the imitation learning enabled B&B executing on a few training samples outperforms baselines.

The rest of this paper is organized as follows. We review the related work in Section II, and present the system model in Section III. In Section IV, we transfer the original problem and design corresponding algorithms to solve the optimization problem. Simulation results are provided in Section V, and Section VI concludes this paper.

II. Related Work

Computation offloading and content/service caching in IoVs have attracted much attention, which can promote network performance and reduce energy consumption [14]. Since a variety of computation-intensive and delay-sensitive applications compete for limited resources and energy, a series of optimization solutions have been presented to improve network performance and user experience. The authors in [15] consider time-varieties of IoVs and latency requirements of vehicular applications to propose a multi-timescale framework, which jointly allocates caching and computing resources in IoVs to minimize energy consumption. The authors in [16] formulate a three-layer architecture (i.e., cloud server layer, Mobile Edge Computing (MEC) server layer and mobile user layer) to cache application services on corresponding edge servers, reducing the execution time of computational tasks. In [17], user mobility is integrated with server placement to specify the offloading decision for delay-sensitive computational tasks. However, edge servers in IoVs generally execute computational tasks locally without cooperation, which is inefficient for task processing due to the uneven workloads in different edge servers.

Some researches investigating cooperative mechanisms in IoVs focus on either resource allocation or user association. The authors in [18] consider the cooperation of a cloud server and several edge servers to execute computation-intensive tasks. A vehicular edge multi-access network is introduced in [19], integrating resource-rich vehicles with the cloud server, to construct a cooperative computing architecture. Some existing studies design computation offloading strategies based on users' properties [20], [21], such as social trust, geographic regions and the number of physical neighbors. However,

they focus on either user-to-server or user-to-user offloading architectures, and few studies investigate offloading decisions among servers. Furthermore, the existing researches mainly perform myopic optimization, which can hardly satisfy the long-term constraints imposed on the entire network.

AI algorithms have been widely investigated to address decision making and resource allocation problems in IoVs. The authors in [22] employ a deep reinforcement learning approach to dynamically orchestrate resources, which can improve the task processing capability of IoVs. The authors in [23] integrate naive Bayes with Support Vector Machines (SVM) to analyze vehicular information, which can efficiently detect negative communication conditions. In [24], a multi-armed bandits based framework is proposed to minimize computation offloading delay, which outperforms the upper confidence bound algorithm. However, these researches need a great deal of offline information as training data, which is not suitable for dynamic networks.

Based on the asymmetrical distribution of traffic flows in real world, the computational workload arrivals in IoVs are highly dynamic and heterogeneous. Existing methods can not make full use of network resources and are hard to provide satisfactory services under heavy traffic flows. Therefore, we construct a hierarchical IoV architecture to minimize the total network delay by considering the cooperation among edge servers. Meanwhile, we formulate a multi-decision making problem under long-term energy constraints, and theoretically prove it can achieve near-optimal performance in an online manner. Furthermore, we propose an AI-based approach by employing imitation learning, which can accelerate the problem solving process with a few training samples.

III. System Model

We consider there are N RSUs, indexed by $\mathcal{N} = \{1, 2, ..., N\}$, deployed near a crossroad and connected by a high-speed LAN [25]. These RSUs, integrating with edge servers, have both computing and caching capabilities, so that vehicles can offload their computational tasks to and download their requested contents from the corresponding RSUs. Vehicles communicate RSUs via wireless communication using orthogonal frequency-division multiplexing technique, which allows vehicles communicate with one RSU without interferences. There are M vehicles arriving at the crossroad area, indexed by $\mathcal{M} = \{1, 2, ..., M\}$, which are overlaid by the RSUs nearby. Moreover, we divide the timeline into several discrete time slots $\mathcal{T} = \{0, 1, ..., T-1\}$, in which vehicles and RSUs can update their decisions and resource allocation strategies. Moreover, each edge server has a caching storage container, which can store requested contents to accelerate application processing procedures. In this paper, we consider application requirements from vehicles consist of two parts: 1) Computational tasks, which can be executed at vehicles or offloaded to the edge server of RSUs. 2) Requested contents, which can be downloaded from the cloud server or the caching storage at the edge servers of RSUs [22]. Generally, we assume computational tasks generated from vehicles follow a Poisson process [26], and the rate of Poisson process is denoted by $\pi_i^t \in [0, \pi_{max}]$.

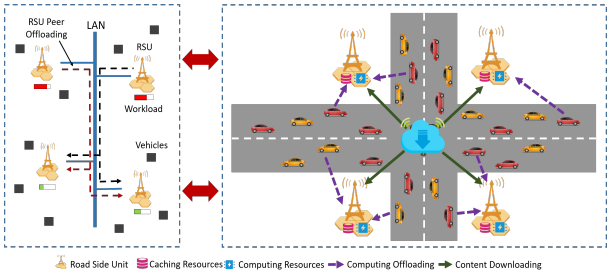


Fig. 1. The framework of edge intelligence empowered IoV.

The framework of the edge intelligence empowered IoV is shown in Fig. 1. Vehicles construct transmission links with nearby RSUs to perform computation offloading and content downloading. If the requested contents are cached, vehicles can download them from RSUs directly. Otherwise, RSUs have to download them from the cloud server. In addition, RSUs are connected by Local-Area Networks (LANs), allowing computational tasks to be transferred among them. According to the observations on the real-world traffic data, it is common that uneven traffic flows come from different directions at the crossroad. Therefore, workload processing among RSUs requires different system costs and resource allocation strategies. The hierarchical IoV is constructed with the cloud server, edge servers and vehicles, which can perform vehicle-to-RSU computation offloading, RSU peer offloading and content caching to improve the network efficiency.

A. Communication Model

The communication rate between vehicle *i* and RSU *m* in time slot *t*, i.e., r_{im}^t, can be calculated by:

$$r_{im}^t = B \log_2(\frac{1 + p_i h_{im}^t}{\sigma^2}), \tag{1}$$

where *B* is the available spectrum bandwidth of the RSU, and p_i denotes the transmission power of vehicle *i*. Variable h_{im}^t represents the channel gain between vehicle *i* and RSU *m*, and σ^2 is the noise power. The expected data size of each computational content is set to s. Obviously, the transmission time depends on the number of input contents (i.e., the task generation rate) from vehicle *i*. The transmission delay of RSU m in time slot *t* is the overall transmission time of its overlaid vehicles, i.e.,

$$T_{m,t}^T = \sum_{i \in M_m} \frac{\pi_i^t s}{r_{im}^t}, \tag{2}$$

where M_m is the set of vehicles associated with RSU m.

B. Computation Model

Computational tasks from vehicles can be computed locally or remotely by the RSUs along roads via computation offloading. The computation delay mainly depends on offloading decisions, including vehicle-to-RSU offloading decisions and RSU peer offloading decisions. If the tasks are computed locally by vehicles, the computation delay can be calculated by:

$$T_{i,t}^L = \frac{\pi_i^t sl}{f}, \tag{3}$$

where l is the expected number of CPU cycles for the calculation of one bit content, and f is the computational capability (CPU cycles per second) of the vehicle.

If the tasks are offloaded to RSUs, the centralized controller first executes RSU peer offloading instead of processing tasks directly. It can alleviate computational pressure, decrease system cost of edge servers with heavy workloads, and improve network efficiency by involving servers with low usage. Let ϕ_i^t denote the total offloading tasks from vehicles to RSU i in time slot t, i.e., $\phi_i^t = \sum_{m \in M_m} \pi_m^t$. We denote $\beta_{ij}^t (j \in \mathcal{N})$ as the tasks offloaded from RSUs i to j in time slot t (notice that β_{ii}^t represents the task processed by RSU i itself). Then, the total task workloads processed by RSU i can be calculated by $\omega_i^t = \sum_{j=1}^{N} \beta_{ji}^t$, consisting of tasks from its overlaid vehicles and offloaded ones from/to other RSUs. Thus, the RSU peer offloading decision are $\beta^t = \{\beta_{ij}^t\}_{i,j \in \mathcal{N}}$.

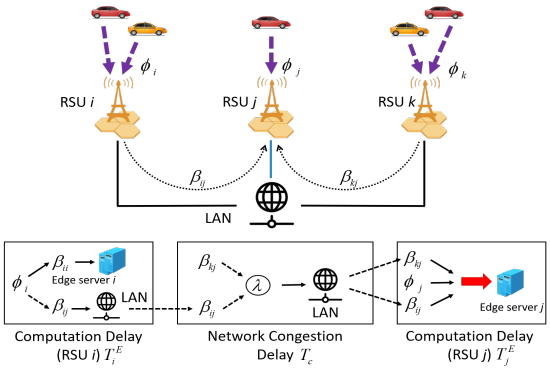


Fig. 2. Illustration of RSU peer offloading system with queuing models.

Proposition 1: *The RSU peer offloading decision β^t is feasible when it satisfies:*

(a) *Conservation:* $\sum_{j=1}^{N} \beta_{ij}^t = \phi_i^t, \forall i \in \mathcal{N}$;

(b) *Stability:* $\omega_i^t \leqslant F_i/sl, \forall i \in \mathcal{N}$.

Condition (a) guarantees the total offloaded tasks by each RSU should be equal to its tasks received from vehicles. Condition (b) indicates that the workloads to be offloaded among RSUs should not exceed the computational capability of each RSU.

Considering the Poisson generation of computational tasks, the peer offloading system can be modeled by an $M/M/1$ queueing model [27]. Hence, the computation delay for RSU m can be calculated by:

$$T_{m,t}^E = \frac{\omega_m^t}{\mu - \omega_m^t}, \tag{4}$$

where $\mu = F/ls$ is the expected service rate, and ω_m^t is the task workloads processed at RSU m with regard to the given peer offloading decision β^t. Herein, F represents the computation capability of the RSU.

Peer offloading among RSUs causes network congestion delay due to the limited bandwidth of the LAN. Let $\lambda_i^t(\beta_t) = \sum_{j \in \mathcal{N}-\{i\}} \beta_{ij}^t = \phi_i^t - \beta_{ii}^t$ denote the tasks offloaded from RSU i, and the total traffic (i.e., the number of offloaded tasks among RSUs) can be calculated by $\lambda^t(\beta_t) = \sum_{i \in \mathcal{N}} \lambda_i^t(\beta_t)$. In order to estimate the congestion delay, we assume that the data size of computational tasks is exponentially distributed as in [16], [22]. According to the queueing theory in [27], the offloading model can be formulated as a first-come first-serve $M/M/1$ system.

Proposition 2: *The congestion delay of the offloading system, i.e., T_t^C, can be calculated by:*

$$T_t^C = \frac{\tau \lambda^t(\beta^t)}{1 - \tau \lambda^t(\beta^t)}, \tau \lambda^t(\beta^t) < 1, \tag{5}$$

where τ is the expected delay for transferring one computational task in the LAN without congestion, and $\lambda^t(\beta_t) = \sum_{i \in \mathcal{N}} \lambda_i^t(\beta_t)$ is the total traffic (i.e., the number of offloaded tasks among RSUs). Herein, $\lambda_i^t(\beta_t) = \sum_{j \in \mathcal{N}-\{i\}} \beta_{ij}^t = \phi_i^t - \beta_{ii}^t$.

Proof: See Appendix A. ■

An example of RSU peer offloading system with queueing models is shown in Fig. 2, in which RSUs i and k offload workloads β_{ij} and β_{kj} to RSU j through LAN, respectively. We define the vehicle-to-RSU offloading decision for vehicle i as $x_i^t \in \{0, 1\}$, where $x_i^t = 0$ represents that vehicle i computes tasks locally, and $x_i^t = 1$ means tasks are computed remotely at RSUs by traffic offloading. Hence, the total computation delay in time-slot t is:

$$T_t^{com} = \sum_{i=1}^{M} \{(1 - x_i^t) T_{i,t}^L + x_i^t (T_t^C + \sum_{m=1}^{N} (T_{m,t}^T + T_{m,t}^E))\}. \tag{6}$$

C. Caching Model

When computational tasks from vehicles are offloaded, the centralized controller first checks whether its requested content has been stored in the server. If so, the edge server does not need to download it from the cloud server. Similar to [22], [28], we assume that the popularity of requested contents from the Internet is determined by a Zipf-like distribution, thus the popularity of i-th ($i = 1, 2, ..., N_f$, where N_f is the total types of contents in the Internet) contents requested by vehicle j can be expressed as:

$$\zeta_i^j = \frac{1}{\rho i^{\epsilon}}. \tag{7}$$

where $\rho = \sum_{i=1}^{N_f} 1/i^{\epsilon}$ with Zipf slope ϵ ($0 < \epsilon < 1$). If the task is cached, the system can reduce the transmission delay between vehicle and the MEC server. However, due to the limited caching storage of each edge server, it is impossible to store all the requested contents. We define $y_i^t \in \{0, 1\}$ as the caching decision of vehicle i, where $y_i^t = 0$ represents the edge server caches the requested contents, and otherwise $y_i^t = 1$. Hence, the caching delay in time-slot t can be expressed by:

$$T_t^{ca} = \sum_{i=1}^{M} x_i^t y_i^t \frac{s}{\zeta_i \overline{R}}, \tag{8}$$

where $\overline{R}$ is the average transmission rate between the Internet and MEC servers.

D. System Cost Model

Hybrid computation offloading and intelligent caching empower satisfactory quality of services, however, it is worth noticing that computation offloading would cause additional system costs for the network operator [29], such as computation cost, communication cost and caching cost. The computation cost for RSU m is used to execute corresponding computational tasks in time slot t, depending on the amount of tasks, i.e., ω_m^t. The communication cost is used to access the virtual network, which maintains stable links between vehicles and servers. The caching cost for RSU m is consumed to store caching contents in edge servers, which depends on the size of caching contents, i.e., $\sum_{i\in\mathcal{M}_i}(1-y_i^t)s$. Hence, the system cost for RSU m in time slot t can be expressed by:

$$E_m^t = \sum_{i\in M_m}[\gamma\cdot r_{im}^t + \epsilon\cdot(1-y_i^t)s] + \delta\cdot\omega_m^t, \gamma,\epsilon,\delta>0, \quad (9)$$

where M_m is the set of vehicles associated with RSU m, and γ is the unit cost for accessing the virtual network. Symbol ϵ is the unit cost for content caching, and δ is the unit cost for task execution.

E. Problem Formulation

Given the latency and resource constraints of vehicular applications as well as the cost budgets of each edge server, we jointly consider hybrid computation offloading (i.e., vehicle-to-RSU offloading and RSU peer offloading), and intelligent caching to minimize the total network delay. In time slot t, the network delay consists of computation delay and caching delay, which can be expressed by:

$$T_t = T_t^{ca} + T_t^{com}. \quad (10)$$

Define $\mathbf{X} = \{x_i^t\}_{i\in\mathcal{M},t\in\mathcal{T}}$, $\mathbf{Y} = \{y_i^t\}_{i\in\mathcal{M},t\in\mathcal{T}}$, and $\mathbf{B} = \{\beta_t\}_{t\in\mathcal{T}}$ as the vectors of vehicle-to-RSU offloading decision, content caching decision, and RSU peer offloading decision, respectively. The optimization problem can be formulated as:

$$\mathbf{P0}: \min_{\mathbf{X},\mathbf{Y},\mathbf{B}} \frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\{T_t\}, \quad (11a)$$

$$\text{s.t. } \frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\{E_m^t\} \leqslant \overline{E}_m, \forall m\in\mathcal{N}, \quad (11b)$$

$$E_m^t \leqslant E_{max}, \forall m\in\mathcal{N}, \forall t\in\mathcal{T}, \quad (11c)$$

$$T_t \leqslant T_{max}, \forall t\in\mathcal{T}, \quad (11d)$$

$$\sum_{i\in\mathcal{M}_i}(1-y_i^t)s \leqslant C, \forall t\in\mathcal{T}, \quad (11e)$$

$$x_i^t\in\{0,1\}, \forall i\in\mathcal{M}, \forall t\in\mathcal{T}, \quad (11f)$$

$$y_i^t\in\{0,1\}, \forall i\in\mathcal{M}, \forall t\in\mathcal{T}, \quad (11g)$$

where constraint (11b) is the long-term cost budget for each RSU. Constraints (11c) and (11d) ensure the system cost and delay in each time slot. Constraint (11e) guarantees that all caching contents cannot exceed the storage capability of each edge server (i.e., C). Constraints (11f) and (11g) are binary variables, representing offloading decision and caching decisions of each vehicle, respectively.

There are two challenges in solving the formulated problem. The first one is that binary variables $\mathbf{X}$ and $\mathbf{Y}$ make **P0** be an MINLP problem, which is proved to be NP-hard in [30]. Furthermore, the offloading decisions are coupled with long-term energy constrains across different time slots, i.e., consuming more energy at the current time slot will leave less available energy for future usage, which calls for an online optimization method without complete information (since tasks arrive in all time slots).

IV. Problem Transformation and Solution

In this section, we reformulate and transform the optimization problem **P0**. Then, we introduce an online method to make traffic control schemes by leveraging the Lyapunov technique and imitation learning enabled branch-and-bound method.

A. Problem Transformation

To address the long-term cost constraints of RSUs without global information, we leverage the Lyapunov optimization method to decouple the long-term cost budget, which strikes a balance between the system delay and energy cost. First, we construct a cost queue for RSU m as the historical measurement of the deficit system cost, which can be expressed by:

$$q_m(t+1) = \max\{q_m(t) + E_m^t - \overline{E}_m, 0\}, \quad (12)$$

where $q_m(t)$ is the queue length in time slot t, and the initial queue backlog $q_m(0)=0$. The cost condition of RSU m is evaluated by $q_m(t)$ for executing peer offloading decision. If the value of $q_m(t)$ gets larger, the system cost will exceed the long-term cost budget, i.e., $\overline{E}_m$. In order to guarantee constraint (11b), $q_m(t)$ of each RSU $m\in\mathcal{N}$ should be stable, i.e., $\lim_{T\to\infty}\mathbb{E}\{q_m(t)\}/T=0$. For all RSUs in the IoV, the set of cost queues is expressed by $\Theta(t)=\{q_m(t)\}_{m\in\mathcal{N}}$.

We define a quadratic Lyapunov function as $L(\Theta(t)) \triangleq \frac{1}{2}\sum_{m=1}^{N}q_m^2(t)$, which represents a scalar metric of the virtual queue length. Variable $L(\Theta(t))$ with small values indicates queue backlogs are small, and queues are stable. A *one-step conditional Lyapunov drift* $\Delta(\Theta(t))$ is defined to consistently push the quadratic Lyapunov function towards a low value, which keeps the queue stable. The drift $\Delta(\Theta(t))$ denotes the change of the cost queue in the Lyapunov function over one time slot. Based on the cost queue, the original problem can be decomposed into a series of real-time optimization problems. Therefore, we define a *Lyapunov-drift-plus-penalty* function, which integrates network delay with cost queue stability by $\Delta(\Theta(t)) + V\sum_{m=1}^{N}\mathbb{E}\{T_t|\Theta(t)\}$. Herein, V is a positive parameter, controlling the trade-off between network delay and system cost.

Lemma 1. *For all feasible values of $\Theta(t)$ determined by peer offloading decision β^t, there is a supremum bound for the drift-plus-penalty function, i.e.,*

$$\begin{aligned}
&\Delta(\Theta(t)) + V\sum_{m=1}^{N}\mathbb{E}\{T_t|\Theta(t)\}\\
\overset{\Delta}{=}&\mathbb{E}\{L(\Theta(t+1)) - L(\Theta(t))|\Theta(t)\} + V\sum_{m=1}^{N}\mathbb{E}\{T_t|\Theta(t)\}\\
\leq& H + \sum_{m=1}^{N} q_m(t)\mathbb{E}\{E_m^t - \overline{E}_m|\Theta(t)\} + V\sum_{m=1}^{N}\mathbb{E}\{T_t|\Theta(t)\}
\end{aligned} \tag{13}$$

where $H = \frac{1}{2}\sum_{m=1}^{N}(E_{max}^2 + \overline{E}_m^2)$ *is a constant value.*

As **Lemma 1** shows, the objective function of **P0** has a supremum bound in each time slot t. Then, we can determine the offloading and caching decisions by solving the optimization problem **P1**, which intends to minimize the supremum bound of problem **P0**:

$$\begin{aligned}
&\mathbf{P1}: \min_{(\mathbf{X},\mathbf{Y},\mathbf{B})^t}(V\cdot T_t(x^t,y^t,\beta^t) + \sum_{m=1}^{N} q_m(t)\cdot E_m^t),\\
&\text{s.t. } (11c)-(11g),
\end{aligned} \tag{14}$$

where the additional term $\sum_{m=1}^{N} q_m(t)\cdot E_m^t$ is supplemented to satisfy the cost constraint in (11b). Generally, system cost minimization is critical for the optimization problem **P1** when $q_m(t)$ is larger. We propose the OMEN algorithm to make optimal offloading and caching decisions, which is shown in Algorithm 1.

Algorithm 1 Online Multi-Decision Making Algorithm

Input: Trade-off parameter V, energy deficit queue backlog $q_m = 0, m \in \mathcal{N}$;

Output: Vehicle-to-RSU offloading decision **X**, content caching decision **Y**, RSU peer offloading decision **B**;

1: **for** $t = 0, 1, ..., T-1$ **do**

2: **Solve** the optimization problem **P1** to get optimal $(x^t, y^t, \beta^t)^*$:
$\min_{x^t,y^t,\beta^t}(V\cdot T_t(x^t,y^t,\beta^t) + \sum_{m=1}^{N} q_m(t)\cdot E_m^t)$

3: Update the cost queue for all RSUs:
$q_m(t+1) = \max\{q_m(t) + E_m^t - \overline{E}_m, 0\}$

4: **end for**

5: **return** $(x^t, y^t, \beta^t)^*_{t=1,2,...,T-1}$

Although we have converted the original problem into a real-time optimization problem, it is also a combinatorial optimization problem, which is NP-hard [31], [32]. The B&B method is leveraged to the obtain optimal offloading and caching policy in each time slot. In one time slot, the centralized controller first decides vehicle-to-RSU policies, i.e., vehicle-to-RSU offloading decisions, and content caching decisions. When computational tasks are offloaded to RSUs, the controller makes optimal RSU peer offloading polices to minimize the network delay. We decompose the objective function as shown in Eq. (15), i.e., (i) a vehicle-to-RSU dependent part, controlling task offloading and content caching, and (ii) an RSU-to-RSU dependent part. Then, we propose the Iterative Multi-RSU Balancing Algorithm (IAMB) to find solution β^* to make optimal RSU peer offloading decisions.

B. Iterative Multi-RSU Balancing Algorithm

In time slot t, for simplicity, we rewrite $\omega_m^t(\beta_t)$ and $\lambda^t(\beta_t)$ as ω_m and λ, respectively. Then, the RSU-to-RSU part of problem **P1** is formulated as:

$$\mathbf{P1}-\beta: \min_{\omega_m,\lambda}\sum_{m=1}^{N}(\frac{V\omega_m}{\mu-\omega_m} + \frac{\tau\lambda}{1-\tau\lambda} + \delta q_m\omega_m), \tag{16a}$$

$$\text{s.t. } E_m(\omega_m,\lambda) \leqslant E_{max}, \forall m \in \mathcal{N}, \tag{16b}$$

$$T_m(\omega_m,\lambda) \leqslant T_{max}, \forall m \in \mathcal{N}, \tag{16c}$$

where ω_m and λ are two independent variables and they are determined by the RSU peer offloading decision β. The relationship between these two variables has been illustrated in Section III-B. Note that an RSU cannot concurrently offload and receive workloads, because it will cause additional network congestion delay.

During RSU peer offloading, the workload flow equation holds in every time slot, i.e., $\sum_{m=1}^{N} I_m = \sum_{m=1}^{N} O_m$, where $I_m \geq 0$ and $O_m \geq 0$ are the inbound and outbound workloads of RSU m, respectively. According to the definition in Section III-B, the variables in $\mathbf{P1}-\beta$ can be written as $\omega_m = \phi_m + I_m - O_m$ and $\lambda = \sum_{m=1}^{N} I_m$, respectively. By substituting I_m and O_m into (16a), we can obtain the optimization problem **P2**, i.e.,

$$\begin{aligned}
\mathbf{P2}: \min_{\mathbf{X},\mathbf{Y}}\sum_{m=1}^{N}[&\frac{V(\phi_m + I_m - O_m)}{\mu - (\phi_m + I_m - O_m)} + \delta q_m(\phi_m + I_m - O_m)\\
&+ \frac{\tau\sum_{m=1}^{N} I_m}{1-\tau\sum_{m=1}^{N} I_m}],
\end{aligned} \tag{17a}$$

s.t.

$$\phi_m + I_m - O_m > 0, \forall m \in \mathcal{N}, \tag{17b}$$

$$-\sum_{m=1}^{N} I_m + \sum_{m=1}^{N} O_m = 0, \tag{17c}$$

Based on **P2** and the conditions of I_m and O_m, we clarify RSUs into three types, i.e., Source RSU ($\mathcal{R}$), Neutral RSU ($\mathcal{U}$) and Sink RSU ($\mathcal{S}$). If $I_m = 0$ and $O_m > 0$, RSU *m* is a Source RSU, meaning that it partially offloads its received workloads to other RSUs and processes the rest of workloads locally. If $I_m = 0$ and $O_m = 0$, RSU *m* is a Neutral RSU, denoting that it processes the *source-offloading* workloads locally without receiving any workloads. Here, the *source-offloading* workloads of RSU *m* is the total offloading tasks from vehicles to RSU *m*. If $I_m > 0$ and $O_m = 0$, RSU *m* is a Sink RSU, representing that it processes the workloads received from other RSUs without offloading any workloads to others. Because the objective function of **P2** is convex and all the constraints are linear, we can leverage Lagrangian Multipliers method to obtain the optimal solution.

Theorem 1. *According to different task processing patterns of RSUs, the optimal solution* ω_m^*, λ^* *to* ***P1***-β *are:*

(a) If $m \in \mathcal{U}$, *then* $\omega_m^* = \phi_m$.

(b) If $m \in \mathcal{R}$, *then* $\omega_m^* = [d_m^{-1}(\frac{1}{V}(\vartheta + Vg(\lambda^*) - \delta q_m))]^+$.

(c) If $m \in \mathcal{S}$, *then* $\omega_m^* = d_m^{-1}(\frac{1}{V}(\vartheta - \delta q_m))$. *Here,* $d_m(\omega_m) \overset{\Delta}{=} \frac{\partial}{\partial\omega_m}(T_m^E) = \mu/(\mu-\omega_m)^2$ *and* $g(\lambda) \overset{\Delta}{=} \frac{\partial}{\partial\lambda}(T_c) =$

$$V \cdot T_t(x^t, y^t, \beta^t) + \sum_{m=1}^{N} q_m(t) \cdot E_m^t = \sum_{i=1}^{M} V\{(1 - x_i^t) \cdot \frac{\pi_i^t sl}{f} + x_i^t \cdot \sum_{m=1}^{N} [\underbrace{\frac{V\omega_m^t(\beta^t)}{\mu - \omega_m^t(\beta^t)} + \frac{\tau\lambda^t(\beta^t)}{1 - \tau\lambda^t(\beta^t)} + \delta q_m(t)\omega_m^t(\beta^t)}_{\text{RSU-to-RSU dependent}} + \gamma q_m(t) r_{im}^t + \epsilon q_m(t)(1 - y_i^t)s + \frac{\pi_i^t s}{r_{im}^t} + y_i^t \frac{s}{\zeta_i \overline{R}}]\}. \tag{15}$$

$\tau/(1-\tau\lambda)^2$. *Variables* λ^* *and* ϑ *are the solutions of the workload flow equation:*

$$\underbrace{\sum_{m \in \mathcal{N}} I_m = \sum_{m \in \mathcal{S}} (d_m^{-1}(\frac{1}{V}(\vartheta - \delta q_m)) - \phi_m)}_{\text{inbound workloads}} = \underbrace{\sum_{m \in \mathcal{N}} O_m = \sum_{m \in \mathcal{R}} (\phi_m - [d_m^{-1}(\frac{1}{V}(\vartheta + Vg(\lambda^*) - \delta q_m))]^+)}_{\text{outbound workloads}}. \tag{18}$$

Proof: See Appendix B. ∎

We develop the IAMB algorithm, which employs a binary search method to find optimal solutions ω_m^* and λ^*. The detailed iterative procedures are described in Algorithm 2. In each iteration, we first determine the set of sink RSUs ($\mathcal{S}(\vartheta)$) according to parameter ϑ, and then calculate inbound workloads λ_s according to Eq. (18). Given the total workloads in the LAN as $\lambda = \lambda_S$, we calculate $\mathcal{R}(\vartheta)$, $\mathcal{U}(\vartheta)$ and λ_R sequentially. After that, we make a judgment on whether λ_S equals to λ_R to determine optimal ϑ or go into the next iterative step. After finding optimal ω_m^* and λ^*, we can obtain the minimal value of **P1-**β by the optimal RSU peer offloading strategy β^*.

Algorithm 2 Iterative Multi-RSU Balancing Algorithm

Input: Offloading tasks to RSU $\phi_m, m \in \mathcal{N}$;
Expected communication delay τ;
Output: ω_m^* and λ^*

$\omega_m \leftarrow \phi_m, m \in \mathcal{N}$
Calculate $\varepsilon_m \triangleq Vd_m(\phi_m) + \delta q_m$ for each RSU
$\varepsilon_{max} \leftarrow \max \varepsilon_m$; $\varepsilon_{min} \leftarrow \min \varepsilon_m$
if $\varepsilon_{min} + Vg(0) \geq \varepsilon_{max}$ **then**
  No RSU performs peer offloading
end if
$a \leftarrow \varepsilon_{min}$; $b \leftarrow \varepsilon_{max}$
$\vartheta \leftarrow \frac{1}{2}(a+b)$
$\lambda_R = \sum_{m \in \mathcal{N}} I_m$, $\lambda_S = \sum_{m \in \mathcal{N}} O_m$
repeat
  $\lambda_S(\vartheta) \leftarrow 0, \lambda_R(\vartheta) \leftarrow 0$
  $\vartheta \leftarrow \frac{1}{2}(a+b)$
  Calculate $\mathcal{S}(\vartheta)$, $\lambda_S(\vartheta)$, $\mathcal{R}(\vartheta)$, $\mathcal{U}(\vartheta)$, $\lambda_R(\vartheta)$
  if $\lambda_R(\vartheta) > \lambda_S(\vartheta)$ **then**
    $b \leftarrow a$
  else
    $a \leftarrow \vartheta$
  end if
until $\lambda_R(\vartheta) - \lambda_S(\vartheta) \geq \widetilde{\nu}$
return ω_m^* and λ^* according to **Theorem** 1

C. Imitation Learning based branch-and-bound method

After determining the optimal value of **P1-**β, the problem consists of vehicle-to-RSU offloading decisions and caching decisions, which are integer variables. Thus, we can adopt the B&B method for the following constitution of problem **P3**:

$$\begin{aligned} \textbf{P3}: \min_{\mathbf{X},\mathbf{Y}} \frac{1}{T} \sum_{t=0}^{T-1} \sum_{i=1}^{M} \{x_i^t \cdot [P(B^*) + \gamma q_m(t) r_{im}^t + \epsilon q_m(t)s \\ + \frac{\pi_i^t s}{r_{im}^t} + y_i^t (\frac{s}{\zeta_i \overline{R}} - \epsilon q_m(t)s)] + (1 - x_i^t) \cdot \frac{\pi_i^t sl}{f}\}, \\ \text{s.t. } (11\text{d}), (11\text{e}), (11\text{f}), (11\text{g}). \end{aligned} \tag{19}$$

where $P(B^*)$ is the minimal value with the corresponding optimal RSU peer offloading decision. Since the original B&B has exponential time complexity (i.e., $\mathrm{O}(2^{MT})$), it is impractical to solve our problem. As the most time consuming part is searching all feasible solutions to guarantee the optimality of the current solution, it is significant to design a good pruning policy to reduce the time complexity.

In the learning based B&B method, the set of vehicle-to-RSU offloading decisions x and caching decisions y is utilized to construct the binary searching tree to obtain the optimal solution of problem **P3**. Then, we determine the optimal set of (x, y) by a tree searching process from the root to a proper leaf node, and decide whether a node is fathomed sequentially using the pruning policy. Imitation learning is employed to solve the sequential problems [13]. We denote the nodes leading to the optimal solution as optimal nodes and others are non-optimal ones. State $s \in \mathscr{S}$ reflects the situations of the visited nodes in the tree, action $a \in \mathscr{A}$ decides whether to preserve the node as the optimal one, and the policy $\pi \in \Pi$ constructs a mapping relation between the state and the action, i.e., $\pi(s) = a$. Imitation learning aims to find optimal action $a^* \in \mathscr{A}$ with different states, i.e., the best pruning policy π^* to find the optimal solution in our problem. The action space is two-dimensional (i.e., $\{prune, preserve\}$), so that we can solve the problem via binary classification to decide whether to prune or preserve the node.

1) Feature designing: We investigate problem-independent and problem-dependent features to formulate the features closely related to state $s \in \mathscr{S}$.

Problem-Independent Features: They mainly reflect the situation of the searching tree constructed by the B&B method, including:

Node feature: State s is closely related to the location of current node s, including the depth of node s, the plunge depth of node s, and the relaxed optimal value b_U^s by substituting the current decisions into Eq. (19).

Branching feature: It explores the source of the current node, which is the branching variable leading to current node s. We

take the branching variable obtained by its father node as the branching feature of node s.

Tree feature: It captures the features obtained from the searching tree to describe the features of state s, containing current optimal objective value b^* and the obtained number of solutions.

Problem-Dependent Features: They mainly reflect the characters of IoVs, including:

Transmission feature: When the offloading decision is not determined at node s in the searching process, the transmission rate can affect decision making. Therefore, the transmission rate between the vehicle and the edge server is captured as the transmission feature.

Storage feature: It concentrates on caching decisions at node s, and can be formulated by a function combining caching size C with task attributes (e.g., task arrival rate π_i and size per content s). Because caching size of RSUs and computational task attributes vary across different problems, the normalized storage feature can be expressed as $g(C, \pi_i, s) = NC/\sum_{i=1}^{M} \pi_i s$.

2) Binary classifier learning: After obtaining proper features of the formulated problem, SVM [33] is utilized to train the classifier, where the input is the designed features of each node and the output is the binary label of $\{prune, preserve\}$, corresponding to the non-optimal node (equals to 0) and the optimal node (equals to 1), respectively. Two aspects of the B&B searching process deserve to be noticed. The first one is the node with a small depth occupies an important position in the searching process. In addition, pruning optimal nodes is more serious than preserving non-optimal nodes.

To deal with the problems mentioned above, we first place a weight parameter ω_1 to train examples from nodes with different depths. It is set as $\omega_1 = Pe^{\frac{-Rd}{H}}$, where d is the depth of the node, and H is the maximum depth of the searching tree. While P and R are utilized to adjust the training process. A larger value of P emphasizes the priority for the node with a smaller depth, while a larger value of R differentiates nodes at different depths more clearly. Furthermore, we set weight parameter ω_2 for optimal and non-optimal nodes, depending on their quantity in the training data. Then, the weight placed for each training sample is calculated by $\omega = \omega_1 \times \omega_2$.

Algorithm 3 Learning Based Branch-and-bound Algorithm

Input: State space $\mathscr{S}$, action space $\mathscr{A}$ and policy space Π.
Output: Best policy π^+ on validation dataset

Initialize policy $\pi^1 = \pi^*$
Initialize training data $\mathscr{D} = \varnothing$
for $m = 1, 2, ..., M$ **do**
  for s in $\mathscr{S}$ **do**
    $\mathscr{D}^s \leftarrow \mathrm{COLLECT(s, \pi^m, \pi^*)}$
    $\mathscr{D} \leftarrow \mathscr{D} \cup \mathscr{D}^s$
  end for
  $\pi^{m+1} \leftarrow$ train binary classifier (SVM) using $\mathscr{D}$
end for
return Best policy π^+ on validation dataset

3) Imitation Learning Process: As the number of nodes increases exponentially, training binary classifier consumes much time and space. Therefore, we employ an imitation learning method, dataset aggregation (DAgger) [34] to reduce learning costs on the premise of guaranteeing the learning performance. The DAgger is an iterative training algorithm, which collects data with current policy in each iteration and then trains a new policy with the aggregation of those collected data. Detailed training procedures of our learning based branch-and-bound algorithm are shown in Algorithm 3.

Algorithm 4 COLLECT(s, π^m, π^*)

Initialize nodelist $\mathscr{N}_s = n([])$
Initialize training data $\mathscr{D} = \varnothing$
while $\mathscr{N}_s \neq \varnothing$ **do**
  Pop a node $n_s(\beta)$ from $\mathscr{N}_s$
  $\mathscr{D} \leftarrow f(n_s(\beta)), \pi^*(n_s(\beta))$
  if $\pi^m(n_s(\beta)) = branch$ **then**
    Branch on node $n_s(\beta)$
    Add children nodes of $n_s(\beta)$ into $\mathscr{N}_s$
    Solve **P1-**β according to Algorithm 2 corresponding to node $n_s(\beta)$
  end if
end while
return $\mathscr{D}$

We first set the expert policy of imitation learning π^* as an initial policy and the training dataset be an empty set. In each iteration, we search each problem in state space $\mathscr{S}$ with π^m and collect data into training dataset $\mathscr{D}$. Then, we train binary classifier using $\mathscr{D}$ in order to learn a new policy π^{m+1}. We repeat this process M times and choose the best-performed policy π^+. The data collection process is shown in Algorithm 4, which collects data with different labels, i.e., *prune* or *preserve*.

D. Performance Analysis

Theorem 2. *Given the optimal solution $(x^t, y^t, \beta^t)^*$ obtained by OMEN, the long-term network delay should satisfy:*

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\{T_t(x^t, y^t, \beta^t)^*\} < T^{opt} + \frac{H}{V}, \tag{20}$$

and the long-term energy deficit should satisfy:

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \sum_{m=1}^{M} \mathbb{E}\{E_m^{c,t}(\beta^{*,t}) - \overline{E}_m\} \leq \frac{1}{\eta}(H + V(T^{max} - T^{opt})), \tag{21}$$

*where $T^{opt} = \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\{T_t(x^t, y^t, \beta^t)^{opt}\}$ represents the optimal solution of **P0**, and $T^{max} = NT_{max}$ represents the largest network delay. Variable $\eta > 0$ is a constant tuning value with stable strategies.*

Proof: See Appendix C. ∎

Compared with the optimal solution of **P0**, **Theorem 2** demonstrates that the OMEN algorithm is able to achieve a rigorous performance bound. Moreover, the OMEN algorithm asymptotically achieves the optimal performance of

offline problem **P0** by setting $V \to \infty$. Equations (20) and (21) demonstrate that there is a $[\mathrm{O}(1/V), \mathrm{O}(V)]$ trade-off between delay and energy consumption, indicating that the time-average energy deficit grows linearly with parameter V. Meanwhile, a large energy deficit is required to stabilize the network and guarantee the converged network delay.

Following the common assumptions and analysis for imitation learning in [34], the computational complexity of our proposed method depends on the number of samples we have, and the expected number of convex problems solved. The computational complexity of convex problems depends on the IMAB method, which employs a binary search method to find the optimal solutions. Thus, the computational complexity of this part is $\mathrm{O}(\log N)$, where N is the number of RSUs. The number of training samples for the classifier is the sum of nodes explored in the branch-and-bound tree of each problem instance. Note that the number of problem instances is $|T|$ and the computational complexity of exploring nodes in the branch-and-bound tree is $\mathrm{O}(M^2)$, where M is the number of vehicles. Therefore, the total computational complexity can be expressed as $\mathrm{O}(M^2|T|\log N)$. This is much more efficient than the standard branch-and-bound, whose time complexity is $\mathrm{O}(2^{M|T|}\log N)$.

V. Performance Evaluation

One IoV system is constructed at a crossroad covered with a 200×200 m^2 area, where 4 RSUs are distributed at the roadside of each entrance. The arrival rate of Poisson process is $\pi_i^t \in [0, 4]$ content per time slot for vehicle i. Meanwhile, the expected data size of each computational content is $s = 20$Mb and the required CPU resource is $l = 1$ cycle per bit. Expected transmission delay is $\tau = 0.2$s in the LAN with 100Mps, and transmission power of vehicles is set to $p_i = 100$mW. The channel gain h_{im} is modeled as $L[dB] = 128.1 + 37.5 \log 10(d[m])$, and the noise power is -174dBm/Hz when tasks are offloaded to the edge server. The energy consumption for computation is $9 \times 10^{-5} W \cdot h$, and $\overline{E}_m = 1W \cdot h/GHz$ is the energy constraint for utilizing computing resources. The total types of requested content are $N_f = 50$ with size 20Mb for each content, and the caching capability of each server is 100Mb. We select 50 crossroads in the city center of Hangzhou, China, and analyze the traffic flows during the evening peaks (from 5 p.m. to 7 p.m.). The number of vehicles is M $\in [50, 450]$, and the network operator can provide different computing resources $F \in [4, 36]$ GHz. The illustration of the selected crossroads in the real-world map is shown in Fig. 3.

We compare the presented OMEN algorithm in comparison with other three baselines:

(a) No RSU Peer offloading (NoRP) [7]: No RSU performs peer offloading and the received tasks from vehicles are processed by their associated RSUs. This method can fulfill tasks without considering long-term energy constraints;

(b) Average Energy Constraint (AEC) [35]: It aims to strictly satisfy the long-term energy constraint by posing an average constraint on RSU in each time slot, i.e., $E_m^t \leqslant \overline{E}_m/T$;

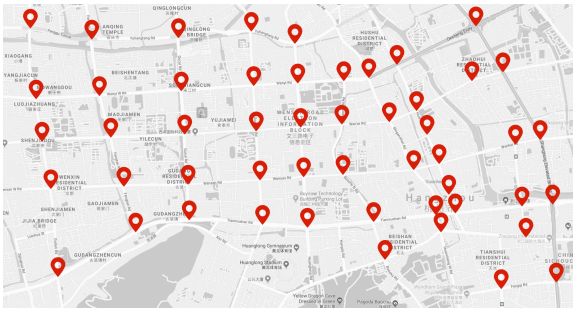
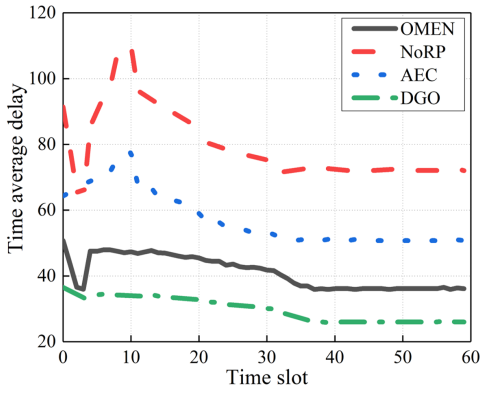
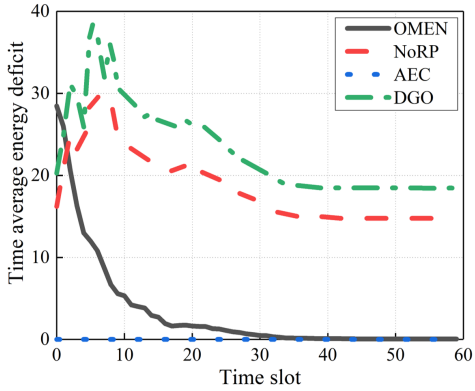


Fig. 3. Illustration of data selection in Hangzhou, China.

(c) Delay-Greedy Optimization (DGO): It aims to minimize the delay of each task greedily. However, it can hardly satisfy the long-term energy constraints.



(a) Network delay

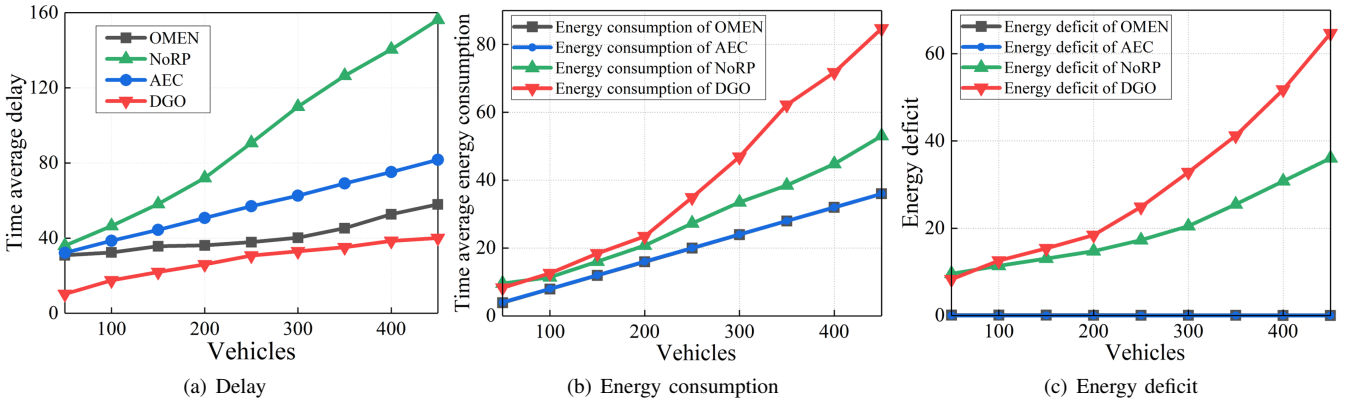


(b) Network energy deficit

Fig. 4. Performance comparison with different number of time slots.

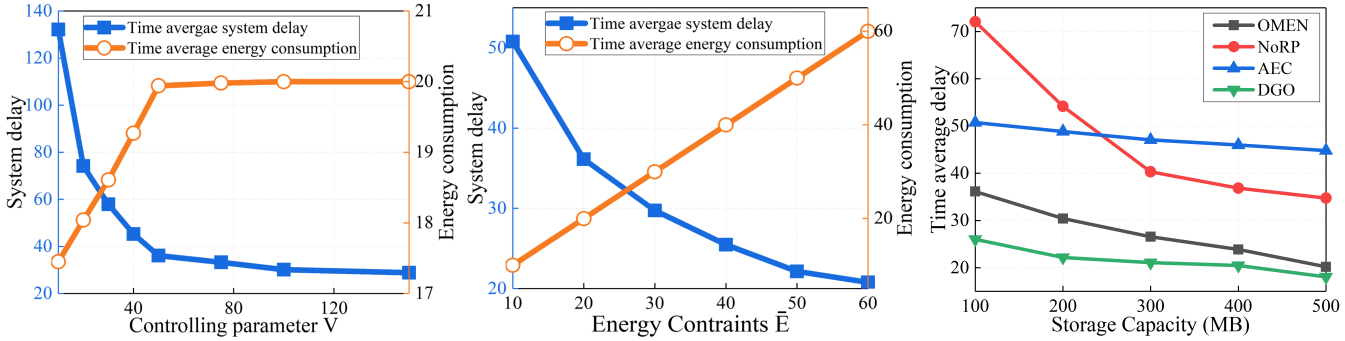
A. System Performance

Fig. 4 shows the performance of different methods from two aspects, i.e., network delay and network energy deficit. The scheme with a large energy deficit means that it consumes much energy on task computing. Moreover, energy deficit converges to 0 if the scheme tightly follows the long-term energy constraints. We can observe that the IoV system without RSU peer offloading has the largest delay and the second largest energy consumption. The IoV system with AEC scheme has a rather large delay. However, it has the lowest energy consumption, which strictly satisfies the long-term energy constraints in all time slots. By contrast, the



(a) Delay (b) Energy consumption (c) Energy deficit

Fig. 5. Performance comparison with different number of vehicles.



(a) Impact of V (b) Impact of $\overline{E}$ (c) Impact of C

Fig. 6. Impacts of three key parameters, i.e., controlling parameter V, energy constraint $\overline{E}$, and storage capability C.

DGO method consumes much energy to achieve the lowest network delay. However, its consumed energy outdistances the energy constraints. Compared with those three baselines, the presented OMEN sacrifices the network delay slightly and reduces the energy consumption significantly.

Fig. 5 illustrates the performance of our method with different number of vehicles. In Fig. 5(a), we can see that the network delay of all these schemes increases when the network serves more vehicles. Meanwhile, the energy consumption increases since more computing resources are deployed for vehicles. Specifically, the network delay of OMEN and DGO increases slowly when the number of serving vehicles grows, meaning that they are suitable for dense traffic flows. However, DGO achieves the lowest network delay with much consumption, which can hardly satisfy the predetermined energy constraints. As observed in Fig. 5(b) and 5(c), the energy consumption of DGO scheme increases more sharply when the network serves more vehicles, which can hardly meet the long-term energy constraints. Compared with other baselines, our proposed scheme can work well in dense vehicular networks, and strictly satisfy the energy constraints.

B. Impact of Parameters

Impacts of three key parameters are evaluated in Fig. 6. As mentioned in Section IV-D, there is a $[\mathrm{O}(1/V), \mathrm{O}(V)]$ trade-off between network delay and energy consumption. A higher value of V achieves lower delay performance, but it calls for more energy consumption. Fig. 6(a) shows that $V = 50$ is appropriate for the network operator, because OMEN has little improvement on network delay and energy consumption, its performance almost remains unchanged by increasing V. Fig. 6(b) illustrates network delay and energy consumption with different energy constraints. We can observe that OMEN achieves lower network delay and consumes larger energy with the increase of energy constraints. A higher energy constraint implies the RSU is able to process more tasks. Meanwhile, we can observe that the energy consumption of our method is strictly lower than the predetermined energy consumption constraint. Fig. 6(c) illustrates network delay under different storage capabilities of RSUs. As the energy is mainly consumed for computational tasks, the caching storage has little influence on the network delay. We can observe that the network delay declines with the increase of the storage capacity, since more contents can be cached at RSUs. In addition, the network delay of OMEN becomes close to that of DGO as the storage capacity increases due to less impact on energy constraints.

C. Time complexity

We compare the performance of our proposed method with Deep Q Network (DQN) [22] and the optimal Branch-and-Bound method (B&B) [36] for their optimality (i.e., system delay) and running speed. Notice that we change the size of experience relay to represent different training samples for

DQN. In addition, we use 200 testing samples to test the performance, and the results are presented in Table I.

TABLE I
PERFORMANCE OF OMEN WITH DIFFERENT TRAINING SAMPLES

Training samples		50	100	150	200
Delay	OMEN	92.61%	93.61%	94.61%	97.07%
	DQN	75.32%	85.06%	91.33%	96.06%
	B&B	100%			
Speed	OMEN	9.15x	4.84x	4.63x	4.32x
	DQN	2.64x	2.29x	2.17x	2.06x
	B&B	1.00x			

Using the B&B as the baseline, we can see that OMEN can achieve near-optimal system delay and reduce computational complexity significantly. It can be observed that OMEN speeds up the B&B algorithm by 4.32 times, while only sacrificing 2.93% of the performance with 200 training samples. However, DQN speeds up the B&B algorithm by only 2.06 times and the system performance drops 3.94%. For training models with less examples (i.e., 50 training samples), OMEN can speed up about 9.15 times with 7.39% performance loss. However, DQN only achieves 75.32% system delay, and sometimes does not converge to the optimal value.

VI. CONCLUSION

In this paper, we construct a novel hierarchical framework for edge intelligence empowered IoV with computation offloading and content caching, and formulate an MINLP optimization problem to minimize the total network delay under long-term energy constraints for RSUs. Then, we propose an efficient online algorithm (i.e., OMEN) to orchestrate edge computing and caching resources by leveraging the Lyapunov optimization method, and prove its near-optimal performance compared with the oracle method. By formulating the pruning process in the B&B as a sequential decision problem, we proposed an imitation learning based method to obtain the near-optimal solution to the formulated MINLP problem. We can learn the pruning action by the SVM method and reduce the learning cost by DAgger, which achieves outstanding learning performance with a few training samples. Performance evaluations based on real-world traffic data in Hangzhou, China, demonstrate that our method for computation offloading and intelligent caching is practical, and achieves high-efficiency performance in large-scale vehicular networks.

VII. ACKNOWLEDGEMENTS

This work is partially supported by National Key R&D Program of Chinaunder Grant No. 2018YFE0206800, National Natural Science Foundation of China under Grant Nos. 61971084, 61771120 and 61671092, Fundamental Research Funds for the Central Universities under Grant No. DUT19JC18, National Natural Science Foundation of Chongqing under Grant No. cstc2019jcyj-msxmX0208, open research fund of National Mobile Communications Research Laboratory, Southeast University under Grant No. 2020D05, and Shenzhen Science and Technology planning project under Grant No. JCYJ20170818111012390.

APPENDIX

A. Proof of Proposition 2

In the formulated queueing system, the service rate is $1/\tau$ and the arriving workload is $\lambda^t(\beta^t)$. According to the state-transition diagram, we can obtain the following equilibrium equation:

$$\begin{cases} \lambda P_0 = \dfrac{P_1}{\tau}, \\ \lambda P_{n-1} + \dfrac{P_{n+1}}{\tau} = (\lambda^t(\beta^t) + \dfrac{1}{\tau})P_n, n \geq 1, \end{cases} \tag{22}$$

Then we can obtain the state probability as:

$$\begin{cases} P_0 = 1 - \lambda^t(\beta^t)\tau, \\ P_n = (1 - \lambda^t(\beta^t)\tau)(\lambda^t(\beta^t)\tau)^n, n \geq 1, \end{cases} \tag{23}$$

According to the definition of expectation, the congestion delay can be calculated by:

$$\begin{aligned} T_t^C &= \sum_{n=0}^{\infty} nP_n = \sum_{n=0}^{\infty}(1 - \lambda^t(\beta^t)\tau)(\lambda^t(\beta^t)\tau)^n \\ &= \frac{\tau\lambda^t(\beta^t)}{1 - \tau\lambda^t(\beta^t)}, \tau\lambda^t(\beta^t) < 1. \end{aligned} \tag{24}$$

B. Proof of Theorem 1

Proof: To simplify the formulation, we use $F(\mathbf{x}, \mathbf{y})$ to replace the objective function of problem **P2**. Then, we formulate the Lagrangian function as:

$$\begin{aligned} L = F(\mathbf{x}, \mathbf{y}) + \vartheta(-\sum_{m=1}^{N} I_m + \sum_{m=1}^{N} O_m) \\ + \sum_{m=1}^{N} a_m(\phi_m + I_m - O_m) + b_m \sum_{m=1}^{N} I_m + c_m \sum_{m=1}^{N} O_m, \end{aligned} \tag{25}$$

where ϑ, a_m, b_m and c_m are Lagrange multipliers. The first-order Kuhn-Tucker condition of the optimal solution is:

$$\frac{\partial L}{\partial I_m} = Vd_m(\phi_m + I_m - O_m) + \delta q_m - \vartheta + a_m + b_m = 0, \tag{26a}$$

$$\begin{aligned} \frac{\partial L}{\partial O_m} = &- Vd_m(\phi_m + I_m - O_m) - \delta q_m \\ &+ Vg(\lambda) + \vartheta - a_m + c_m = 0, \end{aligned} \tag{26b}$$

$$\frac{\partial L}{\partial \vartheta} = -\sum_{m=1}^{N} I_m + \sum_{m=1}^{N} O_m = 0, \tag{26c}$$

$$\begin{aligned} \phi_m + I_m - O_m \geqslant 0, a_m(\phi_m + I_m - O_m) = 0, \\ a_m \leqslant 0, \forall m \in \mathcal{N}, \end{aligned} \tag{26d}$$

$$I_m \geqslant 0, b_m O_m = 0, b_m \leqslant 0, \forall m \in \mathcal{N}, \tag{26e}$$

$$O_m \geqslant 0, c_m O_m = 0, c_m \leqslant 0, \forall m \in \mathcal{N}, \tag{26f}$$

By summing up (26a) and (26b), we have $-Vg(\lambda) = b_m + c_m, \forall m \in \mathcal{N}$. Since $V > 0$ and $g(\lambda) > 0$, we can determine that either $b_m < 0$ or $c_m < 0$. According to (26e) and (26f), there are three conditions for I_m and O_m:

(a) $I_m = 0$ and $O_m = 0$: By substituting these variables into $\omega_m^* = \phi_m + I_m - O_m$, we can obtain:

$$\omega_m^* = \phi_m. \tag{27}$$

In order to search optimal ω_m^* for ***Neutral RSU***, we first obtain $a_m = 0$ by following (26d). Then, we substitute these variables into (26a) and (26b) to obtain the following search inequation:

$$\frac{\vartheta - \delta q_m}{V} \leq d_m(\omega_m^*) \leq \frac{\vartheta + g(\lambda) - \delta q_m}{V}. \tag{28}$$

(b) $I_m = 0$ and $O_m > 0$: To satisfy (26f), we have $c_m = 0$ and $0 \leqslant \omega_m^* < \phi_m$. Then we substitute these variables into (26a) and (26b), and can obtain:

$$\omega_m^* = [d_m^{-1}(\frac{1}{V}(\vartheta + Vg(\lambda^*) - \delta q_m))]^+. \tag{29}$$

In order to search optimal ω_m^* for ***Source RSU***, we leverage the following search inequation:

$$d_m(\omega_m^*) \geq \frac{\vartheta + g(\lambda^*) - \delta q_m}{V}. \tag{30}$$

(c) $I_m > 0$ and $O_m = 0$: To satisfy (26d) and (26e), we have $\omega_m^* > \phi_m$, $a_m = 0$ and $b_m = 0$. Then we substitute these variables into (26a), and can obtain:

$$\omega_m^* = d_m^{-1}(\frac{1}{V}(\vartheta - \delta q_m)) \tag{31}$$

In order to search optimal ω_m^* for ***Sink RSU***, we leverage the following search inequation:

$$d_m(\omega_m^*) < \frac{\vartheta - \delta q_m}{V}. \tag{32}$$

By considering (17c), (29) and (31), we can derive the following workload flow equation:

$$\begin{aligned} &\underbrace{\sum_{m\in\mathcal{N}} I_m = \sum_{m\in\mathcal{S}} (d_m^{-1}(\frac{1}{V}(\vartheta - \delta q_m)) - \phi_m)}_{\text{inbound workloads}} \\ &= \underbrace{\sum_{m\in\mathcal{N}} O_m = \sum_{m\in\mathcal{R}} (\phi_m - [d_m^{-1}(\frac{1}{V}(\vartheta + Vg(\lambda^*) - \delta q_m))]^+)}_{\text{outbound workloads}}. \end{aligned} \tag{33}$$

Therefore, **Theorem 1** can be proved. ∎

C. Proof of Theorem 2

Proof: We introduce a supplemental Lemma to prove the performance bound of the OMEN.

Lemma 2. *There is a stationary and randomized policy* Π *for* ***P1*** *with an arbitrary* $\upsilon > 0$*, which decides* $(x^t, y^t, \beta^t)^\Pi$ *independently from the current queue backlog. Then, we have the following inequalities:*

$$\sum_{m=1}^{N} \mathbb{E}\{T_t(x^t, y^t, \beta^t)^\Pi\} \leq T^{opt} + \upsilon, \tag{34a}$$

$$\mathbb{E}\{E_m^{c,t}(\beta^{\Pi,t}) - \overline{E}_m\} \leq \upsilon, \tag{34b}$$

$$\begin{aligned} &\Delta(\Theta(t)) + V\sum_{m=1}^{N} \mathbb{E}\{T_t(x^t, y^t, \beta^t)^*|\Theta(t)\} \\ &\leq H + \upsilon\Theta(t) + V(T^{opt} + \upsilon). \end{aligned} \tag{34c}$$

Proof: **Lemma 2** can be proved by Theorem 2 in [16]. ∎

By summing up (34c) in all time slots and setting $\upsilon \to 0$, we have:

$$\begin{aligned} &\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\{T_t(x^t, y^t, \beta^t)^*\} \\ &\leq \frac{H}{V} + T^{opt} - \frac{1}{TV}\mathbb{E}\{L(\Theta(t)) - L(\Theta(0))\}. \end{aligned} \tag{35}$$

where $L(\Theta(t)) \geq 0$ and $L(\Theta(0)) = 0$, yielding the upper-bound for long-term network delay, i.e., $H/V + T^{opt}$.

We rewrite υ, $(T^{opt} + \upsilon)$ as $-\eta$ and $\Gamma(\eta)$ ($\Gamma(\eta) < T^{opt}$), respectively, and plug (34a), (34b) into (13):

$$\begin{aligned} &\Delta(\Theta(t)) + V\sum_{m=1}^{N} \mathbb{E}\{T_t(x^t, y^t, \beta^t)^*|\Theta(t)\} \\ &\leq H + V\Gamma(\eta) - \eta\Theta(t) < H + VT^{opt} - \eta\Theta(t). \end{aligned} \tag{36}$$

By summing up (36) in all time slots, we can derive the bound of long-term energy deficit as:

$$\begin{aligned} &\frac{1}{T}\sum_{t=0}^{T-1}\sum_{m=1}^{M} \mathbb{E}\{E_m^{c,t}(\beta^t) - \overline{E}_m\} \\ &\leq \frac{1}{T}\sum_{t=0}^{T-1}\sum_{m=1}^{M} \mathbb{E}\{q_i(t)\} < \frac{1}{\eta}(H + V(T^{max} - T^{opt})). \end{aligned} \tag{37}$$

Therefore, **Theorem 2** can be proved. ∎

REFERENCES

[1] J. Zhao, S. Ni, L. Yang, Z. Zhang, Y. Gong, and X. You, "Multiband cooperation for 5g hetnets: A promising network paradigm," *IEEE Vehicular Technology Magazine*, vol. 14, no. 4, pp. 85–93, 2019.

[2] K. Zhang, Y. Mao, S. Leng, Y. He, and Y. Zhang, "Mobile-edge computing for vehicular networks: A promising network paradigm with predictive off-loading," *IEEE Vehicular Technology Magazine*, vol. 12, no. 2, pp. 36–44, 2017.

[3] K. Zhang, Y. Zhu, S. Maharjan, and Y. Zhang, "Edge intelligence and blockchain empowered 5g beyond for the industrial internet of things," *IEEE Network*, vol. 33, no. 5, pp. 12–19, 2019.

[4] J. Zhao, Q. Li, Y. Gong, and K. Zhang, "Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 8, pp. 7944–7956, 2019.

[5] K. Zhang, Y. Mao, S. Leng, Y. He, S. Maharjan, S. Gjessing, Y. Zhang, and D. H. Tsang, "Optimal charging schemes for electric vehicles in smart grid: A contract theoretic approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 9, pp. 3046–3058, 2018.

[6] K. Zhang, S. Leng, X. Peng, L. Pan, S. Maharjan, and Y. Zhang, "Artificial intelligence inspired transmission scheduling in cognitive vehicular communications and networks," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1987–1997, 2019.

[7] S. Guo, B. Xiao, Y. Yang, and Y. Yang, "Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing," in *IEEE INFOCOM 2016*, 2016, pp. 1–9.

[8] K. Zhang, Y. Zhu, S. Leng, Y. He, S. Maharjan, and Y. Zhang, "Deep learning empowered task offloading for mobile edge computing in urban informatics," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 7635–7647, 2019.

[9] J. Yu, J. Liu, R. Zhang, L. Chen, W. Gong, and S. Zhang, "Multi-seed group labeling in rfid systems," *to be appeared in IEEE Transactions on Mobile Computing*, DOI: 10.1109/TMC.2019.2934445, 2019.

[10] Z. Zheng, Y. Yang, J. Liu, H.-N. Dai, and Y. Zhang, "Deep and embedded learning approach for traffic flow prediction in urban informatics," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 10, pp. 3927–3939, 2019.

[11] Z. Ning, K. Zhang, X. Wang, M. S. Obaidat, L. Guo, X. Hu, B. Hu, Y. Guo, B. Sadoun, and R. Y. Kwok, "Joint computing and caching in 5g-envisioned internet of vehicles: A deep reinforcement learning-based traffic control system," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[12] Z. Ning, P. Dong, X. Wang, J. J. Rodrigues, and F. Xia, "Deep reinforcement learning for vehicular edge computing: An intelligent offloading system," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 6, pp. 1–24, 2019.

[13] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne, "Imitation learning: A survey of learning methods," *ACM Computing Surveys*, vol. 50, no. 2, pp. 21:1–21:35, 2017.

[14] Z. Zhou, H. Yu, C. Xu, Y. Zhang, S. Mumtaz, and J. Rodriguez, "Dependable content distribution in d2d-based cooperative vehicular networks: A big data-integrated coalition game approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 3, pp. 953–964, 2018.

[15] L. T. Tan, R. Q. Hu, and L. Hanzo, "Twin-timescale artificial intelligence aided mobility-aware edge caching and computing in vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 4, pp. 3086–3099, 2019.

[16] J. Xu, L. Chen, and P. Zhou, "Joint service caching and task offloading for mobile edge computing in dense networks," in *IEEE INFOCOM*, 2018, pp. 207–215.

[17] T. Ouyang, Z. Zhou, and X. Chen, "Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 10, pp. 2333–2345, 2018.

[18] Z. Ning, P. Dong, X. Kong, and F. Xia, "A cooperative partial computation offloading scheme for mobile edge computing enabled internet of things," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4804–4814, 2019.

[19] G. Qiao, S. Leng, K. Zhang, and Y. He, "Collaborative task offloading in vehicular edge multi-access networks," *IEEE Communications Magazine*, vol. 56, no. 8, pp. 48–54, 2018.

[20] Y. Li, X. Wang, X. Gan, H. Jin, L. Fu, and X. Wang, "Learning-aided computation offloading for trusted collaborative mobile edge computing," *IEEE Transactions on Mobile Computing*, pp. 1–1, 2019.

[21] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2795–2808, October 2016.

[22] Y. He, F. R. Yu, N. Zhao, V. C. Leung, and H. Yin, "Software-defined networks with mobile edge computing and caching for smart cities: A big data deep reinforcement learning approach," *IEEE Communications Magazine*, vol. 55, no. 12, pp. 31–37, 2017.

[23] N. Cheng, F. Lyu, J. Chen, W. Xu, H. Zhou, S. Zhang, and X. S. Shen, "Big data driven vehicular networks," *IEEE Network*, vol. 32, no. 6, pp. 160–167, 2018.

[24] K. Wang, Y. Tan, Z. Shao, S. Ci, and Y. Yang, "Learning-based task offloading for delay-sensitive applications in dynamic fog networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 11, pp. 11 399–11 403, Nov 2019.

[25] K. Lin, J. Luo, L. Hu, M. S. Hossain, and A. Ghoneim, "Localization based on social big data analysis in the vehicular networks," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 4, pp. 1932–1940, 2017.

[26] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017.

[27] C. Newell, *Applications of queueing theory*. Springer Science & Business Media, 2013, vol. 4.

[28] W. Jiang, G. Feng, and S. Qin, "Optimal cooperative content caching and delivery policy for heterogeneous cellular networks," *IEEE Transactions on Mobile Computing*, vol. 16, no. 5, pp. 1382–1393, 2017.

[29] S. Ni, J. Zhao, H. H. Yang, and Y. Gong, "Enhancing downlink transmission in mimo hetnet with wireless backhaul," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 7, pp. 6817–6832, 2019.

[30] M. R. Garey and D. S. Johnson, *Computers and intractability*. wh freeman New York, 2002, vol. 29.

[31] L. Blumrosen and S. Dobzinski, "Welfare maximization in congestion games," *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 6, pp. 1224–1236, August 2007.

[32] M. Chen, B. Liang, and M. Dong, "Joint offloading and resource allocation for computation and communication in mobile cloud with computing access point," in *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, May 2017, pp. 1–9.

[33] C.-C. Chang and C.-J. Lin, "Libsvm: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, p. 27, 2011.

[34] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011, pp. 627–635.

[35] Q. Liao and D. Aziz, "Modeling of mobility-aware rrc state transition for energy-constrained signaling reduction," in *IEEE GLOBECOM*, 2016, pp. 1–7.

[36] Z. Li, C. Wang, and R. Xu, "Computation offloading to save energy on handheld devices: A partition scheme," in *Proceedings of the 2001 International Conference on Compilers, Architecture, and Synthesis for Embedded Systems*. ACM, 2001, pp. 238–246.

Zhaolong Ning (M'14-SM'18) received the M.S. and PhD degrees from Northeastern University, Shenyang, China. He was a Research Fellow at Kyushu University, Japan. He is an associate Professor Dalian University of Technology, China, and an Adjunct Professor with Chongqing University of Posts and Telecommunications, China. He has published over 100 scientific papers in international journals and conferences. His research interests include Internet of vehicles, mobile edge computing, and artificial intelligence.

Kaiyuan Zhang received B.S. degree from Dalian University of Technology, Dalian, China, in 2017. He is currently working toward the M.S. degree in the same university. His research interests include mobile edge computing, artificial intelligence and network caching.

Xiaojie Wang received the M.S. degree from Northeastern University, China, in 2011. From 2011 to 2015, she was a software engineer in NeuSoft Corporation, China. She received the PhD degree from Dalian University of Technology, Dalian, China, in 2019. Currently, she is a postdoctor in the Hong Kong Polytechnic University. Her research interests are wireless networks, mobile edge computing and machine learning.

Lei Guo received the Ph.D. degree from the University of Electronic Science and Technology of China, Chengdu, China, in 2006. He is currently a Full Professor with Chongqing University of Posts and Telecommunications, Chongqing, China. He has authored or coauthored more than 200 technical papers in international journals and conferences. He is an editor for several international journals. His current research interests include communication networks, optical communications, and wireless communications.

Xiping Hu is a professor with Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, China. He was the co-founder and CTO of Bravolol Limited in Hong Kong, a leading language learning mobile application company with over 100 million users, and listed as top 2 language education platform globally. He has around 90 papers published and presented in prestigious conferences and journals, such as IEEE TETC/TVT/TII/IoT journal, ACM TOMM, IEEE COMST, IEEE Communications Magazine, IEEE Network, HICSS, ACM MobiCom, and ACM WWW etc. His research areas consist of distributed intelligent systems, crowdsensing, social networks, and cloud computing. He holds a PhD in Electrical and Computer Engineering from The University of British Colombia, Vancouver, Canada.

Jun Huang received the Ph.D. (Hons.) degree in communication and information system from the Institute of Network Technology, Beijing University of Posts and Telecommunications, Beijing, China, in 2012. He is currently a Full Professor of Computer Science with the Chongqing University of Posts and Telecommunications, Chongqing, China. He has authored more than 100 publications including papers in prestigious journal and conferences. His current research interests include network optimization and control, machine-to-machine communications, and the Internet of Things.

Bin Hu (M'10-SM'15) is currently a professor in Lanzhou University and a guest professor in ETH Zurich, Switzerland. He is an IET Fellow, co-chairs of IEEE SMC TC on Cognitive Computing, and Member at Large of ACM China, Vice President of International Society for Social Neuroscience (China committee) etc. He has published more than 100 papers in peer reviewed journals, conferences, and book chapters including Science, Journal of Alzheimer's Disease, PLoS Computational Biology, IEEE Trans., IEEE Intelligent Systems, AAAI, etc. He has served as Chairs/Co-Chairs in many IEEE international conferences/workshops, and associate editors in peer reviewed journals on Cognitive Science and Pervasive Computing, such as IEEE Trans. Affective Computing, Brain Informatics, IET Communications, etc.

Ricky Y. K. Kwok (F'14) received a B.Sc. degree in Computer Engineering from the University of Hong Kong in 1991, and the M.Phil. and Ph.D. degrees, both in Computer Science, from the Hong Kong University of Science and Technology (HKUST) in 1994 and 1997, respectively. His research focus has been on designing efficient communication protocols and robust resources management algorithms toward enabling large scale distributed mobile computing. In these research areas, he has authored one textbook, co-authored another two textbooks, and published more than 200 technical papers in various leading journals, research books, and refereed international conference proceedings. He is a Fellow of the HKIE, the IEEE, and the IET. From March 2006 to December 2011, Ricky served on the Editorial Board of the Journal of Parallel and Distributed Computing as a Subject Area Editor in Peer-to-Peer Computing. He also served as an Associate Editor for the IEEE Transactions on Parallel and Distributed Systems from January 2013 to December 2016.