# CoPace: Edge Computation Offloading and Caching for Self-Driving With Deep Reinforcement Learning

Hao Tian, *Graduate Student Member, IEEE*, Xiaolong Xu, *Member, IEEE*, Lianyong Qi, *Member, IEEE*, Xuyun Zhang, Wanchun Dou, *Member, IEEE*, Shui Yu, *Senior Member, IEEE*, and Qiang Ni, *Senior Member, IEEE*

***Abstract*—Currently, self-driving, emerging as a key automatic application, has brought a huge potential for the provision of in-vehicle services (e.g., automatic path planning) to mitigate urban traffic congestion and enhance travel safety. To provide high-quality vehicular services with stringent delay constraints, edge computing (EC) enables resource-hungry self-driving vehicles (SDVs) to offload computation-intensive tasks to the edge servers (ESs). In addition, caching highly reusable contents decreases the redundant transmission time and improves the quality of services (QoS) of SDVs, which is envisioned as a supplement to the computation offloading. However, the high mobility and time-varying requests of SDVs make it challenging to provide reliable offloading decisions while guaranteeing the resource utilization of content caching. To this end, in this paper we propose a collaborative computation offloading and content caching method, named CoPace, by leveraging deep reinforcement learning (DRL) in EC for self-driving system. Specifically, we first introduce OSTP to predict the future time-varying content popularity, taking into account the temporal-spatial attributes of requests. Moreover, a DRL-based algorithm is developed to jointly optimize the offloading and caching decisions, as well as the resource allocation (i.e., computing and communication resources) strategies. Extensive experiments with real-world datasets in Shanghai, China, are conducted to evaluate the performance, which demonstrates that CoPace is both effective and well-performed.**

***Index Terms*—Self-driving, deep reinforcement learning, edge computing, computation offloading, content caching.**

## I. Introduction

SELF-driving is emerging as a key automatic application in intelligent transportation system (ITS) to alleviate the urban traffic congestion, enhance the fuel efficiency, and improve the travel safety [1]. Self-driving vehicles (SDVs), equipped with a large amount of on-board sensors (e.g., camera and radar), are capable of perceiving and analyzing the sophisticated information on roads, which supports numerous driving assistance services such as automatic path planning and collision avoidance [2]. Besides, since the SDVs have independent decision-making and control functionalities without manual intervention, passengers gain more time on the infotainment services, e.g., augmented reality (AR). These services have prompted the considerable real-time requirements for processing capabilities [3]. However, due to the limited computation and storage capacities imposed on the SDVs, executing the computation-intensive services is heavily constrained.

Edge computing (EC) enables the timely completion of latency-sensitive computation tasks, which is provided as a promising alternative to cloud computing by sinking the processing and storage resources to the network edges in the proximity of SDVs [4]. With EC, the sensing infrastructures, e.g., base stations (BSs) and roadside units (RSUs), are endowed with the computing and storage capacities. Therefore, offloading the computation tasks of the SDVs to the nearby edge servers (ESs) attached to the BSs and RSUs for implementation is feasible to satisfy the requirements of ultra-low execution delay and mitigates the backhaul pressure of the cloud data centers [5].

Generally, since there is a high probability that the required data of computation tasks from diverse SDVs have the similarities, the transmission of such duplicated data to the ESs results in a huge waste of computing and communication resources [6]. To minimize the redundant data transmission and guarantee resource utilization, content caching is introduced

Manuscript received June 21, 2021; revised September 27, 2021; accepted October 13, 2021. Date of publication October 19, 2021; date of current version December 17, 2021. This work was supported in part by the Natural Science Foundation of Jiangsu Province of China under Grant BK20211284, in part by the Financial and Science Technology Plan Project of Xinjiang Production and Construction Corps under Grant 2020DB005, and in part by the National Natural Science Foundation of China under Grant 61872219. This research was also supported by the Priority Academic Program Development of Jiangsu Higher Education Institutions (PAPD) fund. The work of Dr. Xuyun Zhang is the recipient of an ARC DECRA (project No. DE210101458) funded by the Australian Government. The review of this article was coordinated by Dr. Joongheon Kim. *(Corresponding author: Xiaolong Xu.)*

Hao Tian is with the School of Computer and Software, Nanjing University of Information Science and Technology, Nanjing 210044, China (e-mail: withhaotian@gmail.com).

Xiaolong Xu is with the School of Computer and Software, Nanjing University of Information Science and Technology, Nanjing 210044, China, and with the Jiangsu Collaborative Innovation Center of Atmospheric Environment and Equipment Technology (CICAEET), Nanjing University of Information Science and Technology and Engineering, Nanjing 210044, China, and also with the State Key Laboratory Novel Software Technology, Nanjing University, China (e-mail: xlxu@ieee.org).

Lianyong Qi is with the School of Information Science and Engineering, Qufu Normal University, Rizhao 276826, China (e-mail: lianyongqi@gmail.com).

Xuyun Zhang is with the Department of Computing, Macquarie University, Sydney, NSW 2109, Australia (e-mail: xuyun.zhang@mq.edu.au).

Wanchun Dou is with the State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China (e-mail: douwc@nju.edu.cn).

Shui Yu is with the School of Computer Science, University of Technology Sydney, Ultimo, NSW 2007, Australia (e-mail: Shui.Yu@uts.edu.au).

Qiang Ni is with the School of Computing and Communications, Lancaster University, Lancaster LA1 4WA, U.K. (e-mail: q.ni@lancaster.ac.uk).

Digital Object Identifier 10.1109/TVT.2021.3121096

by mobile network operators (MNOs) that enables the contents to be prefetched [7]. Caching reusable contents of computation tasks on the ESs is feasible for SDVs to cut down the repetitive computation offloading and execution, which reduces the network delay and improves the quality of service (QoS) of SDVs. Consequently, the content caching is envisioned as an effective complement to the computation offloading [8].

However, considering the constrained storage capacity, it is unattainable for ESs to cache all the contents. For the non-cached contents, the SDVs with computation requests have to offload them to the ESs for completion. Thus, how to effectively cache popular contents on the ESs is crucial to improve the QoS of SDVs. In most of the existing works, the content popularity is usually supposed to follow the Zipf distribution [9], [10]. Nevertheless, owing to the dynamic requests of SDVs, the content popularity is time-varying, such that the Zipf distribution is not always practical [11]. On the other hand, due to the high mobility of SDVs, the computation offloading and content caching strategies with imbalanced resource allocation cause the overload or underutilized in some ESs. Accordingly, the reliable offloading and caching decisions is imperative to enhance the resource efficiency of ESs, and thereby provide high-quality services for SDVs.

Technically, deep reinforcement learning (DRL) is empowered with the powerful decision-making capability to tackle the sophisticated offloading and caching problems in the vehicular networks, which plays a critical role in self-driving systems [12]. Multiple DRL approaches, e.g., Q-learning and deep Q-network (DQN), are utilized to finding the optimal offloading and caching solutions in vehicular networks [13]–[15]. However, to the best of our knowledge, few works have focused on the joint optimization of computation offloading, content caching, and resource allocation, taking into account the prediction of time-varying and spatial-temporal content popularity. Therefore, we introduce CoPace for EC-envisioned self-driving to jointly orchestrate the offloading, caching, and resource allocation decision-making of SDVs. Specifically, a spatial-temporal based content popularity prediction algorithm is designed to acquire the future popularity of content requests. Then, we propose an online DRL-based offloading, caching, and resource allocation algorithm to minimize the system latency. The main contributions of this paper are as follows.

- A collaborative computation offloading and content caching framework for EC-enabled self-driving is developed. Moreover, the computation offloading and content caching issue is formulated as a mixed integer non-linear programming (MINLP) optimization problem to minimize the latency.
- An offline spatial-temporal based content popularity prediction algorithm, named OSTP, is proposed to predict the future time-varying and mobility-aware content popularity of SDV requests.
- A DRL-based algorithm is designed to collaboratively optimize the offloading, caching decisions, and resource allocation policies. Especially, the MDP problem is solved by the deep deterministic policy gradient (DDPG) approach.
- Comparative experiments are conducted based on the real-world datasets of base station telecom accessing histories and taxi trajectories in Shanghai, China, to demonstrate the performance and effectiveness of the CoPace.

The rest of this paper is organized as follows. Section II reviews the related work. Section III introduces the system model and formulates the optimization problem. Section IV proposes the CoPace method. In Section V, the experimental evaluation is presented. The conclusion is drawn in Section VI.

## II. Related Work

Currently, EC mitigates the computation limitation of mobile devices by sinking cloud resources to the network edges, which has attracted considerable attention. Besides, with the advance of 5 G technology and the soaring demands of computation-intensive and delay-sensitive vehicular services, the EC-empowered Internet of Vehicles (IoV) has been drawing widespread attention. Hou *et al.* [16] combined the software-defined networking (SDN) technique and EC-aided IoV, and considered the computation offloading and task allocation problems. A heuristic algorithm was presented to address the optimization problem. Considering the high mobility of vehicles, providing an energy-efficient IoV environment for vehicular applications still faces challenges. Pei *et al.* [17] designed a power dispatch framework, which minimized the whole network delay by taking into account the queuing model, incomplete channel status, and the mobility of vehicles. In addition, the joint optimization for computing, caching, and resource scheduling is also concerned. Zhao *et al.* [18] focused on the edge computation and caching management that aims to cooperatively optimize caching, request scheduling, and resource allocation.

Proactive caching contributes to minimizing the content fetching time and improving the QoS in vehicular networks. Specifically, considering the increasing heterogeneous requirements for vehicular applications, multiple works have been done on edge caching in IoV. Taking into account the mobility of vehicles and dynamic content popularity, Gao *et al.* [19] put forward a probabilistic caching approach to optimize the content placement. This dynamic content caching method outperforms some static caching strategies (e.g., LRU). To achieve the long-term reduction of system cost and delay and the Improvement of hit ratio, Qiao *et al.* [15] collaboratively learn the optimal content delivery and placement method by applying the DRL-based approach, considering the content popularity. Different from the conventional content popularity distribution, they proposed a global content popularity function that can adapt to the EC-enabled IoV. Yu *et al.* [20] presented a proactive edge caching approach, leveraging the federated learning to protect the private data information in distinct vehicles.

Due to the stochastic features of resource allocation and decision-making in IoV, DRL is a nature-inspired approach for reducing the solution complexity and enabling intelligent perceiving and decision-making capabilities. To improve resource utilization, Xiong *et al.* [21] developed an enhanced algorithm based on DRL technology to achieve the minimization of the execution delay and the increasing amount of requested

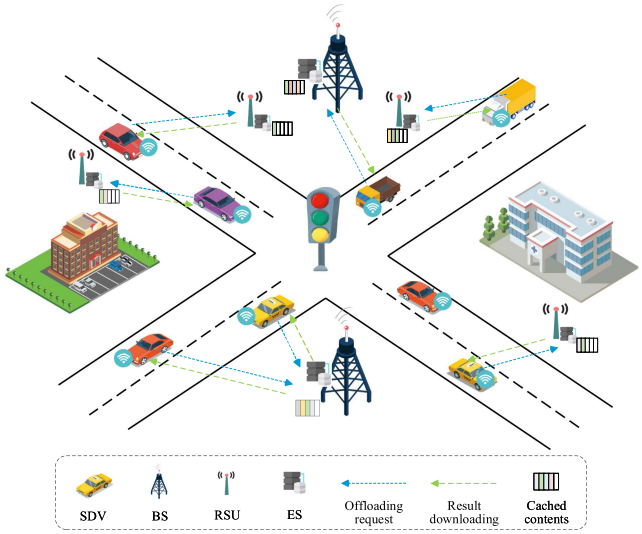



Fig. 1. The architecture of collaborative computation offloading and content caching in EC-empowered self-driving system.

resources. Besides, the resource utilization and energy efficiency in IoV also need to be considered. Luo *et al.* [22] designed an integrated framework with several key elements in vehicular EC and resorted to the improved DQN scheme to address the resource scheduling problem for better utilization performance. Ke *et al.* [23] introduced the computation offloading scheme, which leverages the DRL technology to cooperatively optimize the energy cost and transmission delay. Due to the multiple data sharing among the increasing number of vehicles, data security and privacy are also needed to be concerned. To guarantee data privacy in knowledge sharing, Chai *et al.* [24] designed a method to jointly utilize the blockchain and federated learning technologies.

Different from these works, the DRL-based collaborative method to optimize the offloading and caching policies, as well as the resource allocation strategies, are considered in this paper, in view of the spatial-temporal content request features. Furthermore, considering the high dimension states and the hybrid continuous and discrete actions, the existing DRL approaches Q-learning and DQN are difficult to handle. As a result, we resort to a DDPG-based algorithm to solve this issue.

## III. System Model and Problem Formulation

### A. System Overview

Fig. 1 illustrates the EC-empowered self-driving system, which mainly includes three kinds of entities, i.e., BS, RSU, and SDV. RSUs and BSs, equipped with ESs, are endowed with computing and caching functionalities and can provision computing and caching services to the proximity of SDVs. Let $\mathcal{M} = \{1, 2, \ldots, M\}$ represent the index set of BSs, $\mathcal{N} = \{1, 2, \ldots, N\}$ represent the index set of RSUs, $\mathcal{V} = \{1, 2, \ldots, V\}$ represent the index set of SDVs, respectively. Without loss of generality, the constructed system works in a time slot mode, and the timeline is divided into discrete time frame $\tau \in \mathcal{T} = \{1, 2, \ldots, T\}$. The BSs or RSUs can deal with the computation requirements from SDVs, then SDVs upload the task to the corresponding BSs or RSUs. Let $\mathcal{K} = \{1, 2, \ldots, K\}$ represent the index set of required contents of computation task. Moreover, computation-intensive services from SDVs are required to be accomplished with the least possible delay.

TABLE I
Main Notations

| Notation | Description |
|---|---|
| $\mathcal{M}$ | The set of BSs, $\mathcal{M} = \{1, 2, \ldots, M\}$ |
| $\mathcal{N}$ | The set of RSUs, $\mathcal{N} = \{1, 2, \ldots, N\}$ |
| $\mathcal{V}$ | The set of SDVs, $\mathcal{V} = \{1, 2, \ldots, V\}$ |
| $\mathcal{K}$ | The set of contents, $\mathcal{K} = \{1, 2, \ldots, K\}$ |
| $L_i(\tau)$ | The mobility of SDV $i$ at time slot $\tau$ |
| $c_k$ | The input size of content $k$ |
| $\delta_k$ | The required CPU cycles for processing content $k$ |
| $r_k$ | The size of processed result of content $k$ |
| $F_j$ | The total computation capability at ES $j$ |
| $C_j$ | The total caching storage at ES $j$ |
| $B_j$ | The total bandwidth capacity at ES $j$ |
| $SNR_{i,r}(\tau)$ | The signal noise ratio between SDV $i$ and RSU/BS $r$ |
| $R_{i,r}(\tau)$ | The data transmission rate between SDV $i$ and RSU/BS $r$ |
| $b_{i,r}(\tau)$ | The allocated bandwidth on RSU/BS $r$ for SDV $i$ |
| $f_{i,r}(\tau)$ | The allocated computation resource on RSU/BS $r$ for SDV $i$ |

As presented in Fig. 1, moving SDVs in the crossroad can access the neighboring RSUs via the wireless channel for computation task offloading. Considering the delay constraints of the vehicular services and the limited computation capabilities of the SDVs, the contents are uploaded to and executed on the target RSUs or BSs to satisfy the computation demands. In addition, if the required contents are cached on the RSUs or BSs, the processed results are fetched directly from the ESs to SDVs without uploading. Otherwise, SDVs have to offload the non-cached contents to the target RSU for execution. Furthermore, due to the high movement of SDVs and dynamic vehicular network topology, how to choose the optimal ES to implement the computational tasks and cater to delay-constraint requirements is challenging. Intuitively, the MNOs deploy the cooperative task offloading with a caching mechanism, which can not only reduce the access latency and task execution time but also provide SDVs high-quality vehicular services. Key notations are illustrated in Table I.

### B. SDV Mobility Model

Generally, SDVs require high-quality vehicular services to meet the application demand from users. However, the high movement of SDVs poses a great challenge to construct the reliable computation offloading and content caching for EC-envisioned self-driving system [3]. To this end, it is essential to consider the dynamic mobility of SDVs. For SDV $i$, let $lx_i$ and $ly_i$ denote the location in 2D map. In the time interval from $\tau$ to $\tau'$, the location $L_i(\tau') =< L_i^x(\tau'), L_i^y(\tau') >$ of SDV $i$ at time

slot $\tau'$ is given by

$$L_i^x(\tau') = lx_i(\tau) + d_i \cos\theta_i, \tag{1}$$

and

$$L_i^y(\tau') = ly_i(\tau) + d_i \sin\theta_i, \tag{2}$$

where $d_i$ is the distance of SDV $i$ between time slot $\tau$ and $\tau'$ and $\theta_i$ is the moving angle of SDV $i$.

### C. Communication Model

In the constructed system, SDVs are connected to the RSUs or BS via a wireless channel and can offload the computation tasks and download required results with no channel interference. Without the loss of generality, assumed that the topology of the vehicular network is invariable during one time slot. For the available accessing RSU $r$, SDV $i$ uploads the required content and obtain the desired output data over the wireless communication. Let $x_{i,r}(\tau) = \{0, 1\}$ be defined as the binary variable between the SDV $i$ and RSU $r$ at time slot $\tau$, which indicates whether SDV $i$ accesses to its target ES or not. If $x_{i,r}(\tau) = 1$, SDV $i$ is connecting to RSU $r$ for task execution, otherwise $x_{i,r}(\tau) = 0$. Meanwhile, at one time slot, each SDV can only be served by one RSU.

Besides, the signal noise ratio (SNR) between SDV $i$ and RSU $r$ is given by [6]

$$SNR_{i,r}(\tau) = \frac{p_{i,r}(\tau) G_{i,r}(\tau)}{\xi_{i,r}(\tau) d_{i,r}(\tau) \sigma_{i,r}(\tau)^2}, \tag{3}$$

where $p_{i,r}(\tau)$ is the transmission power; $G_{i,r}(\tau)$ is the channel gain, $\xi_{i,r}(\tau)$ is the path loss, $\sigma_{i,r}(\tau)^2$ is the power of the white Gaussian noise, and $d_{i,r}(\tau)$ is the distance between SDV $i$ and RSU $r$. Note that the mobility of SDVs is time-varying, the distance between SDV $i$ and RSU $r$ is calculated as

$$d_{i,r}(\tau) = \sqrt{[lx_i(\tau) - lx_r(\tau)]^2 + [ly_i(\tau) - ly_r(\tau)]^2}. \tag{4}$$

Considering the parallel processing capability of each ES, it can handle requirements from distinct SDVs simultaneously. The allocated bandwidth resource between SDV $i$ and RSU $r$ is denoted by $b_{i,r}(\tau)$. Therefore, the instantaneous data transmission rate of the wireless connection between the SDV and RSU is expressed as

$$R_{i,r}(\tau) = x_{i,r}(\tau) b_{i,r}(\tau) \log_2(1 + SNR_{i,r}(\tau)). \tag{5}$$

### D. Computation Model

In our model, considering the large enough coverage range of the BSs, the offloaded tasks are performed both on RSU $r$ and the BSs collaboratively. $\mu_i(\tau) \in \{0, 1\}$ is defined as a processing decision variable, which denotes whether or not the RSU has to process the computation task requested from SDV $i$ at time slot $\tau$. If $\mu_i(\tau) = 0$, the BSs process the computation task from SDV $i$. Otherwise, the RSUs are selected to perform. The offloading decision variable $\mu_i(\tau)$ is given by

$$\mu_i(\tau) = \begin{cases} 1, & \text{if task from SDV } i \text{ is processed at RSUs,} \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

The required CPU cycles for processing content $k$ is defined as $\delta_k$. Note that multiple SDVs can share the same RSU within the coverage range. Due to the computation resource constraint of each RSU, it is hard to allocate the full computational capabilities to the SDVs which request task offloading. Therefore, $f_{i,r}(\tau)$ denotes the instantaneous resource allocation at RSU $r$ for the requested SDV $i$. The completion time for the computation task at RSU $r$ and the required task at the BS are both given by

$$\omega_{i,r,k}^{com}(\tau) = \mu_i(\tau) \frac{\delta_k}{f_{i,r}(\tau)}, \tag{7}$$

and

$$\omega_{i,l,k}^{com}(\tau) = (1 - \mu_i(\tau)) \frac{\delta_k}{f_{i,l}(\tau)}, \tag{8}$$

where $f_{i,l}(\tau)$ is the allocated computational resource at the BS $l$ for SDV $i$.

Therefore, the total execution time for SDV $i$ is given by

$$\omega_{i,k}^{com}(\tau) = \mu_i(\tau) \sum_{r=1}^{N} \frac{\delta_k}{f_{i,r}(\tau)} + (1 - \mu_i(\tau)) \sum_{l=1}^{M} \frac{\delta_k}{f_{i,l}(\tau)}. \tag{9}$$

### E. Caching Model

For the offloaded task generated from SDV $i$, the ES on RSU $r$ or BS $l$ has the caching capability to satisfy the required task, which improves the QoS and reduces the computation delay. Note that each ES has a limited caching storage constraint, it is impossible to cache all the requested data. Let $y_k(\tau)$ be the caching decision variable, which denotes whether the required content $k$ is cached on the RSU or BS, where $y_k(\tau)$ is expressed as

$$y_k(\tau) = \begin{cases} 1, & \text{if requested content } k \text{ is cached,} \\ 0, & \text{otherwise.} \end{cases} \tag{10}$$

If the required data have cached on the ES, only the computed result transmits to the corresponding SDV $i$ with no content uploading. Otherwise, SDV $i$ needs to upload the task $k$ to the ES on the RSU or the BS for execution. The feedback time for RSU $r$ and the BS $l$ that sending back the computed results to SDV $i$ is given by

$$\omega_{i,r,k}^{fee}(\tau) = \mu_i(\tau) \frac{r_k}{R_{i,r}(\tau)}, \tag{11}$$

and

$$\omega_{i,l,k}^{fee}(\tau) = (1 - \mu_i(\tau)) \frac{r_k}{R_{i,l}(\tau)}, \tag{12}$$

where $r_k$ denotes the size of processed result of task $k$. Similar to $R_{i,r}(\tau)$, $R_{i,l}(\tau)$ is the instantaneous allocated bandwidth for the BS $l$.

Moreover, the total feedback time for SDV $i$ is given by

$$\omega_{i,k}^{fee}(\tau) = \mu_i(\tau) \sum_{r=1}^{N} \frac{r_k}{R_{i,r}(\tau)} + (1 - \mu_i(\tau)) \sum_{l=1}^{M} \frac{r_k}{R_{i,l}(\tau)}. \tag{13}$$

Let $c_k$ be the size of required content $k$, then the offloading latency for the non-cached data delivers to the RSU $r$ and the BS $l$ by SDV $i$ at time slot $\tau$ is given by

$$\omega_{i,r,k}^{off}(\tau) = (1 - y_k(\tau))\mu_i(\tau)\frac{c_k}{R_{i,r}(\tau)}, \tag{14}$$

and

$$\omega_{i,l,k}^{off}(\tau) = (1 - y_k(\tau))(1 - \mu_i(\tau))\frac{c_k}{R_{i,l}(\tau)}. \tag{15}$$

The total offloading latency for SDV $i$ is expressed as

$$\omega_{i,k}^{off}(\tau) = (1 - y_k(\tau)) \times \left(\mu_i(\tau)\sum_{r=1}^{N}\frac{c_k}{R_{i,r}(\tau)} + (1 - \mu_i(\tau))\sum_{l=1}^{M}\frac{c_k}{R_{i,l}(\tau)}\right). \tag{16}$$

As a result, the total transmission latency contains the data uploading and downloading time, which is expressed as

$$\omega_{i,k}^{tra}(\tau) = \omega_{i,k}^{fee}(\tau) + \omega_{i,k}^{off}(\tau). \tag{17}$$

### F. Problem Formulation

Considering the latency of vehicular services and the network efficiency, the collaborative computation offloading and content caching between the BSs, RSUs, and SDVs aims to minimize the total system latency. For the offloaded task, the target ES on RSU and BS executes in a cooperative manner. If the requested data are cached on the ESs, only the execution time and feedback time are taken into consideration. Otherwise, the additional offloading latency cost is incurred. Therefore, the system latency $\Theta(\tau)$ consists of the execution time and transmission time, where $\Theta(\tau)$ is given by

$$\Theta(\tau) = \sum_{i=1}^{V}\sum_{k=1}^{K}\left[(1 - y_k(\tau))\omega_{i,k}^{com}(\tau) + \omega_{i,k}^{tra}(\tau)\right]. \tag{18}$$

The joint optimization problem, considering offloading and caching decision, as well as the resource allocation, aims to minimizes the system latency is given by

$$\min_{x,\mu,f,b,y}\frac{1}{T}\sum_{\tau=1}^{T}\Theta(\tau). \tag{19}$$

s.t.

$$\sum_{i=1}^{V} x_{i,z}(\tau)\, b_{i,z}(\tau) \le B_z, \forall z \in \mathcal{N} \text{ or } \mathcal{M}, \tag{20a}$$

$$\sum_{i=1}^{V} \mu_i(\tau)\, f_{i,z}(\tau) \le F_z, \forall z \in \mathcal{N} \text{ or } \mathcal{M}, \tag{20b}$$

$$\sum_{k=1}^{K} y_k(\tau)\, c_k(\tau) \le C_z, \forall z \in \mathcal{N} \text{ or } \mathcal{M}, \tag{20c}$$

$$\mu_i(\tau) \in \{0, 1\}, \forall i \in \mathcal{V}, \tag{20d}$$

$$x_{i,z}(\tau) \in \{0, 1\}, \forall i \in \mathcal{V}, z \in \mathcal{N} \text{ or } \mathcal{M}, \tag{20e}$$

$$y_k(\tau) \in \{0, 1\}, \forall k \in \mathcal{K}. \tag{20f}$$

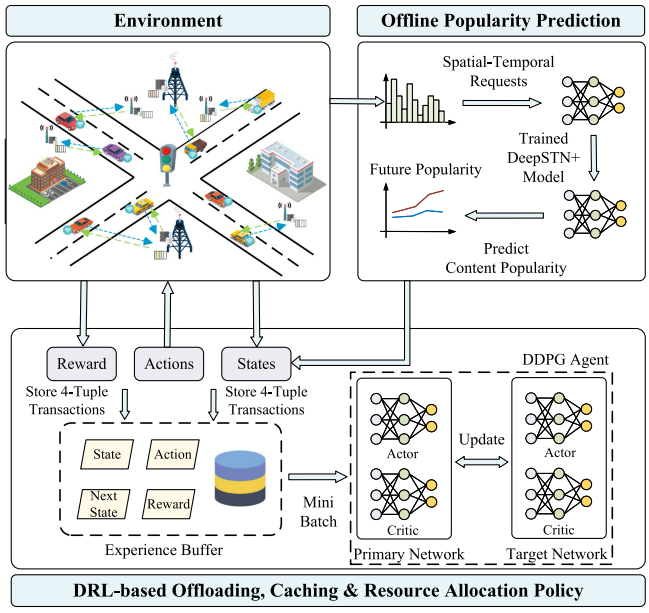



Fig. 2. An overview of CoPace.

where the constraint (20a), (20b), and (20c) ensure the allocated bandwidth, computation resource, and occupied storage to SDVs have to be less than or equal to the communication capability $B_z$, processing capability $F_z$, and storage capability $C_z$ of ES $z$. The constraint (20e) is the decision variable that determines whether computation task is performed at the RSU or at the BS. The constraint (20e) and (20f) are the binary variable, which represent the offloading and caching decisions.

## IV. The Design of CoPace

In this section, we introduce the design of CoPace, which aims to minimize the long-term latency by jointly optimizing the offloading and caching decisions, as well as the resource allocation strategies. First, an overview of CoPace is presented. Afterwards, we propose an offline content popularity prediction algorithm and an online DRL-based algorithm. Furthermore, the complexity analysis of proposed algorithms is drawn.

### A. CoPace Overview

The CoPace method is illustrated in Fig. 2, which is divided into three parts. First, the constructed EC-empowered self-driving system is served as an environment that observes the key information including computation requests, the status of both RSUs and BSs, and the mobility of SDVs. Next, the offline popularity prediction leverages the OSTP algorithm to predict the future content popularity level based on the historical request records by a deep learning model. Furthermore, the DRL-based offloading, caching and resource allocation policy is proposed to utilize the DDPG agent to optimize the system delay. Specifically, the original optimization problem is first modeled as a Markov decision process (MDP). After that, the DRL-based approach interacts with the EC-enabled self-driving environment to address the formulated MDP, aiming to maximize the reward value to achieve the optimal collaborative offloading, caching,

and resource dispatching decisions. The details of CoPace are further elaborated as follows.

### B. Spatial-Temporal Content Popularity Prediction

To minimize the system latency, the ESs can cache the popular contents when multiple SDVs requests the same computation tasks. The popularity of requested content can reflect the preference by requirement. Namely, high popular content is most probably to be requested. Many existing works supposed that the content popularity usually follows the Zipf distribution [9], [10]. However, in the vehicular networks, given the mobility of SDVs and the dynamic resource requirements, the popularity is time-varying such that the assumption may not be always realistic. Therefore, it is challenging to acquire accurate popularity in practical self-driving scenarios to cache target contents to further enhance the QoS of SDVs. In the light of this, we propose OSTP, a spatial-temporal based content popularity prediction algorithm. Considering the movements of SDVs, the content request distributions from SDVs are also affected by the spatial and temporal dependence [25]. Therefore, motivated by DeepSTN+ [26], a deep learning model to capture the spatial-temporal features, the OSTP leverages the DeepSTN+ based model to train and predict the future time-varying and mobility-aware content request distributions. Then, based on the request distribution prediction, OSTP derives the future content popularity. The details of OSTP are described as follows.

First, considering the dynamic content requests, the problem of requested content distribution prediction can be modeled as a time series. To better capture the request distributions in different regions, an area is divided into several grids where each grid has the same size. Let $\mathcal{G} = \{1, 2, \ldots, G\}$ be the index set of regions. Meanwhile, $\phi_{k,g}(\tau)$ is defined as the counts of requested content $k$ generated from SDVs in region $g$ at time slot $\tau$, which is given by

$$\phi_{k,g}(\tau) = \begin{cases} 1, & \text{if content } k \text{ is requested in region } g, \\ 0, & \text{otherwise.} \end{cases} \tag{21}$$

To extract the temporal dependence in different ranges of time, the timeline is grouped into three parts, i.e., recent time, near history, and distant history. These three temporal features are fused into one distribution that denotes the historical record during the $i^{th}$ time interval. The historical content request records from $\mathbf{X}_1$ to $\mathbf{X}_{\tau-1}$ are combined with the prior time-weighted PoI information, feeding to the DeepSTN+ model for training. After the multi-scale outputs, the long-range spatial and temporal attributes are extracted. Then, the convolution operation is utilized to obtain the future predicted content request distributions. Specifically, based on the future request distributions, the content popularity $p_k(\tau)$ of content $k$ at time interval $\tau$ can be computed and updated, where $p_k(\tau)$ is calculated as

$$p_k(\tau) = \frac{\sum_{g=1}^{G} \phi_{k,g}(\tau)}{\sum_{i=1}^{K} \sum_{g=1}^{G} \phi_{i,g}(\tau)}, \forall k \in \mathcal{K}. \tag{22}$$

**Algorithm 1:** OSTP for CoPace.

**Input:** $\mathcal{K}$, $\mathcal{G}$, $\phi_{k,g}(\forall k \in \mathcal{K}, \forall g \in \mathcal{G})$
**Output:** $P(\tau)$
/* Training phase */
1 **while** *Training not finished* **do**
2 Generate the time-weighted request tensor $\mathbf{X}$ by historical record $\phi_{k,g}(\forall k \in \mathcal{K}, \forall g \in \mathcal{G})$ ;
3 Feed to the model with three different granularities of time inputs (closeness, period, and trend) ;
4 Minimize the loss function with Adam optimizer ;
5 **end**
/* Prediction phase */
6 **for** *each testing sample* **do**
7 Predict the future request distribution matrix with the trained model on sampled batch;
8 **end**
9 Calculate the content popularity $p_k(\tau)$ of $k$ at time slot $\tau$ by Eq. (22) ;
10 Derive the content popularity vector $P(\tau)$ at time slot $\tau$ by Eq. (23) ;
11 Return $P(\tau)$

Therefore, the popularity of all contents at time interval $\tau$ is defined as

$$P(\tau) = [p_1(\tau), p_2(\tau), \ldots, p_K(\tau)]. \tag{23}$$

Algorithm 1 shows the process of OSTP for CoPace. In the training phase, the leveraged DeepSTN+ model is trained with the historical records of content requests from SDVs. Specifically, the time-weighted training inputs of request tensor $\mathbf{X}$, which is transformed into three kinds of time granularity, i.e., closeness, period, and trend (denote the difference during the length of time), to better capture temporal features of data (lines 2 to 3). In addition, during the training phase, the objective is to minimize the loss function with Adam optimizer until overfitting occurs (lines 4). Based on the trained model, the future request distributions are derived in the prediction phase. Eventually, we compute the content popularity $p_k(\tau)$ of $k$ and the content popularity vector $P(\tau)$ by Eq. (22) and (23) (lines 6 to 10). According to the predicted content popularity, we then propose an online computation offloading, caching, and resource allocation algorithm to optimize the system latency to improve the QoS of SDVs. The formulated optimization problem is modeled as an MDP and the prediction content popularity is converted as the component of state definition, which is explained in detail in the following.

### C. Implementation of DRL-Based Collaborative Algorithm

Intuitively, the Eq. (19) is a MINLP optimization problem, which is proved to be NP-hard [27]. Due to the stochastic and time-varying requests and high movements of SDVs, conventional techniques, e.g., convex optimization and game theory, hardly handle this problem effectively. Currently, DRL is emerging as a promising approach that enables to reduce the solution complexity by nature-inspired exploration of actions. Inspired by the DRL technology which is endowed with

powerful decision-making capabilities, it is attempted to utilize the DRL approach to get the optimal decision-making to deal with the formulated optimization problem in a time-varying and dynamic environment.

To implement the DRL-based method, the proposed problem is modeled as a MDP. Three main elements (i.e., states, actions, and reward) are taken into consideration, which are presented as follows in detail.

*a) States:* The system state space reflects the observation from the constructed system environment, consisting of the status of SDVs and available resources of ESs. The state $s(\tau)$ at time slot $\tau$ is given by

$$\begin{aligned} s(\tau) = \{&R_1(\tau), R_2(\tau), \ldots, R_V(\tau), C_1(\tau), C_2(\tau), \ldots, \\ &C_V(\tau), P_1(\tau), P_2(\tau), \ldots, P_K(\tau), F_1(\tau), F_2(\tau), \\ &\cdots, F_{N+M}(\tau), B_1(\tau), B_2(\tau), \ldots, B_{N+M}(\tau), \\ &L_1(\tau), L_2(\tau), \ldots, L_V(\tau)\}, \end{aligned} \tag{24}$$

where $R_i(\tau)$ represents the requested content from SDV $i$, $C_i(\tau)$ represents the input size of requested content, $P_i(\tau)$ represents the predicted content popularity level based on Algorithm 1 described in Section IV-B, $F_i(\tau)$ represents available computation resources of ESs, $B_i(\tau)$ represent the available bandwidth resources of ESs, and $L_i(\tau)$ represents the location of SDV $i$ including the latitude and longitude. Considering the mobility of SDVs and different vehicular services, the computation requirement from SDVs and the location of SDVs are time-varying.

*b) Actions:* According to the observed environment states, the agent decides which ES should be selected by SDVs and how many computation and communication resources should be allocated to SDVs. The action $a(\tau)$ is given by

$$\begin{aligned} a(\tau) = \{&o_1(\tau), o_2(\tau), \ldots, o_V(\tau), c_1(\tau), c_2(\tau), \\ &\cdots, c_{N+M}(\tau), f_1(\tau), f_2(\tau), \ldots, f_{N+M}(\tau), \\ &b_1(\tau), b_2(\tau), \ldots, b_{N+M}(\tau)\}, \end{aligned} \tag{25}$$

where $o_i(\tau)$ denotes the association variable whether ES $j \in \mathcal{N}$ or $\mathcal{M}$ is selected to serve SDV $i \in \mathcal{V}$. $c_i(\tau)$ is the caching variable that whether the content is cached in ES. Note that $o_i(\tau)$ and $c_i(\tau)$ are both the binary discrete actions that determine the decision-making of offloading and caching. $f_i(\tau)$ and $b_i(\tau)$ denote the computation resources and the bandwidth resources that ES $j \in \mathcal{N}$ or $\mathcal{M}$ allocates to SDVs, which are the continuous variables.

*c) Reward:* After taking action $a(\tau)$ based on the environment states, the agent obtains the reward $R(s(\tau), a(\tau))$. Meanwhile, the reward function should correspond with the optimization problem (19). The reward function based on the optimization problem minimizing the system latency is defined as

$$R(s(\tau), a(\tau)) = -\Theta(\tau). \tag{26}$$

Based on the states, actions, and reward function, the conventional methods (e.g., convex optimization and game theory) are difficult to solve such problems due to the environment with the dynamic states and state-to-action space. Therefore, the DRL approach is utilized to obtain the optimal solutions. Generally, several DRL algorithms are available, e.g., DQN [28] and DDPG [29]. DQN is a traditional approach that can find the optimal policy based on the Q value. By implementing the deep neural network (DNN) to evaluate the Q value to explore the action compared with the Q-learning algorithm, the DQN algorithm learns the policy with DNN on the high dimension states, but can only handle the discrete actions. Nevertheless, we consider the high dimension state and action with mixed continuous and discrete space in this paper. To this end, the DDPG algorithm is selected, where DDPG is a model-free and actor-critic scheme with two kinds of network architecture that can tackle the high dimension and continuous states and actions by the policy gradient method.

Specifically, to maximize the reward value and obtain the optimal actions, a DDPG-based collaborative computation offloading, caching, and resource allocation scheme is designed. The DDPG agent is responsible for the optimal policy of resource assignment, ES selection, and content placement decision, which involves a replay memory and two components, i.e., actor and critic. In particular, the replay memory is utilized to store experience tuples, consisting of four components: current state, action, next state, and reward, which can be sampled for the actor to train and update the network parameters. The detailed process of the DDPG-based algorithm is explained as follows.

The information of the observed environment is collected by the agent, consisting of the status of SDVs and the status of available resources of ESs. Then the explored policy maps the current state to the action to obtain the resource dispatch and ES selection policy. The primary actor neural network selects the action $a(\tau)$ by adding the noise for better performance of exploring the behavior policy, where action $a(\tau)$ is given by

$$a(\tau) = \mu(s(\tau) \mid \theta^{\mu}) + \mathcal{N}(\tau), \tag{27}$$

where $\mu(s(\tau)|\theta^{\mu})$ denotes the explored policy by the actor and $\mathcal{N}(\tau)$ denotes the random noise. After obtaining the action $a(\tau)$, the immediate reward can be calculated by the agent, and the next state $s(\tau+1)$ is returned.

In the each training step, the four-tuple transitions are extracted from the experience replay buffer as training data sets for the actor and critic. The critic updates the parameter $\theta^{Q}$ by decreasing the cost function, which is given by

$$L(\theta^{Q}) = \boldsymbol{E}\left[\left(y(\tau) - Q(s(\tau), a(\tau) \mid \theta^{Q})\right)^2\right], \tag{28}$$

where $Q(s(\tau), a(\tau)|\theta^{Q})$ denotes the action-value function, which can present the score to evaluate the policy. Therefore, for each action, it requires the state $s(\tau)$ and $s(\tau+1)$ to calculate the action-value function.

Besides, the target value $y(\tau)$ can be calculated by $Q'$ and $\mu'$ of the target critic network, which is calculated as

$$\begin{aligned} y(\tau) = &R(s(\tau), a(\tau)) \\ &+ \lambda Q'\left(s(\tau+1), \mu'\left(s(\tau+1) \mid \theta^{\mu'}\right) \mid \theta^{Q'}\right). \end{aligned} \tag{29}$$

According to the $Q(s(\tau), a(\tau)|\theta^{Q})$ and the sampling experience transition tuples, the parameter $\theta^{\mu}$ updates by the actor neural network by utilizing policy gradient. Such that the policy

**Algorithm 2:** CoPace.

```
/* Initialization phase */
Initialize the replay memory ;
Initialize the μ(s(τ)|θ^μ) and Q(s(τ), a(τ)|θ^Q) ;
Initialize the actor and the critic ;
/* Execution phase */
for each episode do
    Initialize the system state s ;
    for time step = 1, 2, ..., T do
        Pridict the future content request distributions
          and then derive the popularity level of
          contents by Alg. 1 ;
        Collect the s(τ) ;
        Select the a_t based on the μ(s_t|θ^μ) and s(τ) ;
        Obtain the R(s(τ), a(τ)) and s(τ+1) ;
        Store the four-tuple transition into replay
          memory ;
        Sample N transitions from the experience
          replay buffer ;
        Calculate the y(τ) by Eq. (29) ;
        Update the θ^Q to minimize the loss function by
          Eq. (28) ;
        Update the μ(s|θ^μ) ;
        Update the actor and critic by Eq. (31) and Eq.
          (32) ;
        Calculate the system latency by Eq. (18) ;
    end
end
```

gradient is presented as [23]

$$\nabla_{\theta^\mu} J \approx \boldsymbol{E}\left[\nabla a Q\left(s, a|\theta^Q\right)\big|_{s=s(\tau),a=\mu(s(\tau))} \nabla_{\theta^\mu}\mu\left(s|\theta^\mu\right)\Big|_{s=s(\tau)}\right]. \tag{30}$$

At each iteration, the network parameters $\theta^Q$ and $\theta^\mu$ updates utilizing the soft update scheme, which is presented as

$$\theta^{\mu'} \leftarrow \omega\theta^\mu + (1-\omega)\,\theta^{\mu'}, \tag{31}$$

and

$$\theta^{Q'} \leftarrow \omega\theta^Q + (1-\omega)\,\theta^{Q'}, \tag{32}$$

where $\omega \in [0, 1]$ is an update coefficient.

The procedure of CoPace is illustrated in Algorithm 2. First, the initialization phase is conducted for initializing phase. For each episode, based on the predicted content popularity level, the system state needs to be initialized. To maximize the reward value, the system agent interacts with the EC-enabled environment. Based on the behavior policy and states, the action is selected by the system agent in each time slot $\tau$. Then, the reward and the next state are obtained. A four-tuple transition is stored into the replay memory. When the replay buffer is full, a batch of transitions is sampled for training. After that, the system agent updates key network parameters. At the end of each time step, the system latency is calculated.

### D. Complexity Analysis

The complexity of the proposed OSTP and CoPace is based on the computation complexity of neural networks. For OSTP, let $L$ denote the number of multiplications of propagation and backpropagation in neural networks and $N$ denote the number of epochs in the training. The complexity of training phase of OSTP is $O(LN)$. During the prediction phase, the propagation computation of neural networks is represented by $L_k$ and the number of mini-batch is represented by $|I_k|$ of content $k$. The complexity of prediction phase is $O(\sum_{k=1}^{K} L'_k|I_k|)$. The complexity of OSTP is $O(LN + \sum_{k=1}^{K} L'_k|I_k|)$. For CoPace, let $L'$ denote the number of multiplications of propagation and backpropagation in neural networks by DRL agent and let $N'$ denote the number of episodes. Each episode performs $T$ time steps. Within one episode, the OSTP is executed to predict the future content popularity level. Note that we perform the training process of OSTP at the beginning of CoPace, and for each time step the prediction is conducted. Therefore, the complexity of CoPace is $O(LN + N'TL' \sum_{k=1}^{K} L'_k|I_k|)$.

## V. Experimental Evaluation

In this section, extensive simulations of the CoPace is conducted to evaluate the performance.

### A. Datasets

The performance of the CoPace is evaluated on two real-world datasets in Shanghai, China. First, the telecom datasets from Shanghai are utilized, which includes more than 7.2 million accessing histories through more than 3200 base stations from Jun 1, 2014 to Nov 30, 2014 [30]–[32]. In addition, we resort to the taxi trajectory datasets with GPS reports on Feb 2007 in Shanghai, China, which contains 4316 taxis with the movement of longitude and latitude of the location [33]. Inspired by the work [32], these two real-world datasets are combined with the synthetic dataset, and the simulation area is of the latitude from $30°9'$ to $31°4'$ and longitude from $121°12'$ to $121°72'$ in this paper. Fig. 3 illustrates the distribution of base stations by telecom datasets with the records selected from Jul 1 to Jul 31 and the mobility of vehicles by taxi trajectory datasets.

All the simulations are conducted on a PC of Windows 10 with Intel i7-8700 (6 cores, 3.20 GHz), 32 GB DRAM, and NVIDIA GTX 1080 GPU. The Python 3.7 with TensorFlow 1.15.0 and Keras 2.1.5 is implemented as the simulation environment. Besides, the key parameters in the experiment are illustrated in Table II.

### B. Performance Evaluation of OSTP

In this subsection, we conduct the performance evalution of OSTP. Specifically, the simulation area is divided into $5\times6$ regions for model training and prediction. The metrics of root mean squared error (RMSE) and mean absolute error (MAE) are utilized to evaluate the accuracy of prediction for the spatial-temporal distributions. In addition, we perform the OSTP in comparison with the other three baselines as follows:

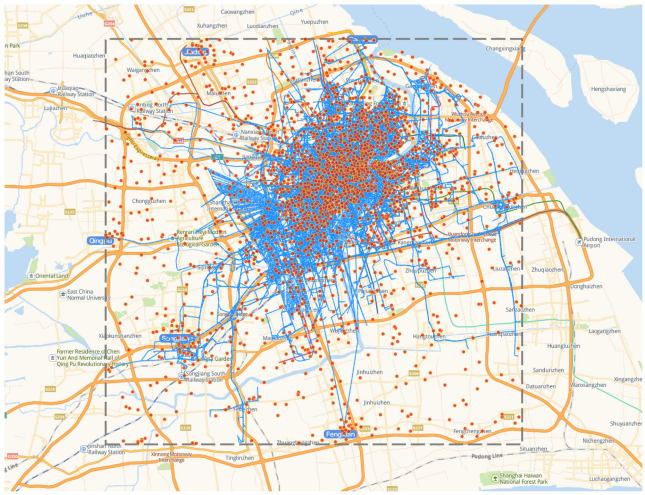


Fig. 3. The distribution of synthetic dataset combined with the real-world telecom datasets and taxi trajectory datasets in Shanghai, China.

TABLE II
KEY PARAMETERS

| Description | Value |
|---|---|
| Number of BSs | 4 |
| Number of RSUs | 18 $\sim$ 30 |
| Number of SDVs | 60 $\sim$ 100 |
| Number of type of contents | 10 |
| Bandwidth of RSUs and BSs | 20, 40 MHz [34] |
| Size of each required data | 60 $\sim$ 100 MB [34] |
| Storage capacity of RSUs and BSs | 0.5, 0.8 GB |
| Computation capacity of RSUs and BSs | 1, 2 GHz [35] |
| Required CPU cycles | 10 $\sim$ 90 Megacycles [35] |
| Gauss white noise power | -174 dBm/Hz [36] |

- Convolution neural network (CNN). A traditional convolution based neural network which can capture the spatial attributes.
- DeepST [37]. The first DNN model utilizes the neural network to capture the spatial-temporal data dependence.
- ST-ResNet [38]. A classical convolution based residual network, which can model the neighboring and distant spatial-temporal features.

Fig. 4 illustrates the comparison of prediction error on baselines with different batch sizes from two evaluation metrics, i.e., RMSE and MAE. The OSTP is better than CNN, DeepST, and ST-ResNet on both metrics. It can be observed in Fig. 4(a) that when the batch size is higher, the RMSE of OSTP decreases linearly. In particular, OSTP outperforms 16.7 % than ST-ResNet, which is the model that can effectively capture the spatial-temporal features, when batch size is 32. MAE means the absolute error between the ground-truth and the predicted one. As shown in Fig. 4(b), OSTP outperforms other baselines on different batch sizes. When batch size is 32, OSTP is 27.1 % lower on MAE than DeepST. Generally, the bigger size of the batch can improve the efficiency of training. Nevertheless, the higher batch size does not mean better performance. It can be

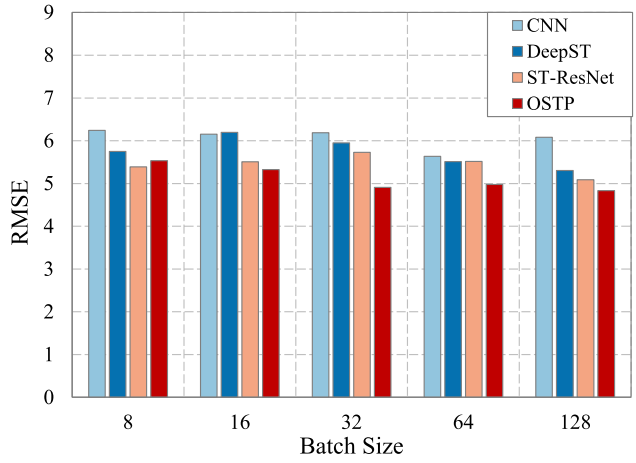



Fig. 4. Prediction error with different batch size. (a) Comparison on RMSE. (b) Comparison on MAE.

seen that the OSTP of batch size 32 achieves the lowest RMSE and MAE.

Fig. 5 elaborates the predicted performance of the content popularity. It can be seen that the predicted popularity level based on the OSTP is very close to the ground-truth value. The main reason is that with the OSTP, the spatial-temporal attributes of requirements from SDVs are more efficiently to be captured and higher prediction performance is acquired. First, with the consideration of 10 types of contents, the content popularity level is based on the Eq. (22). In addition, the request distributions are predicted and then the future content popularity level is derived. Due to the spatial-temporal dependence of content requests, it is effective to achieve the features of request distribution by deep learning technology. For example, the ground-truth content popularity of content 6 is 7.98e-2, and the predicted value of content 6 is 8.36e-2.

### C. Performance Evaluation of CoPace

CoPace is proposed to optimize the coordination of offloading decisions, content placement, and resource allocation. The

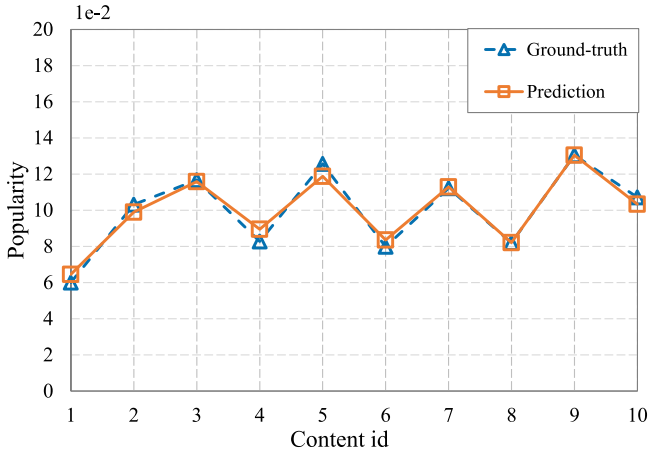



Fig. 5. The prediction performance of the content popularity by OSTP.

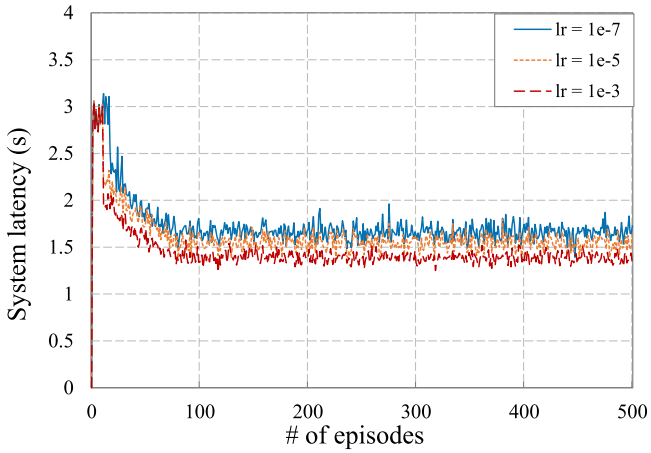



Fig. 6. Convergence performance of the CoPace with different learning rate.

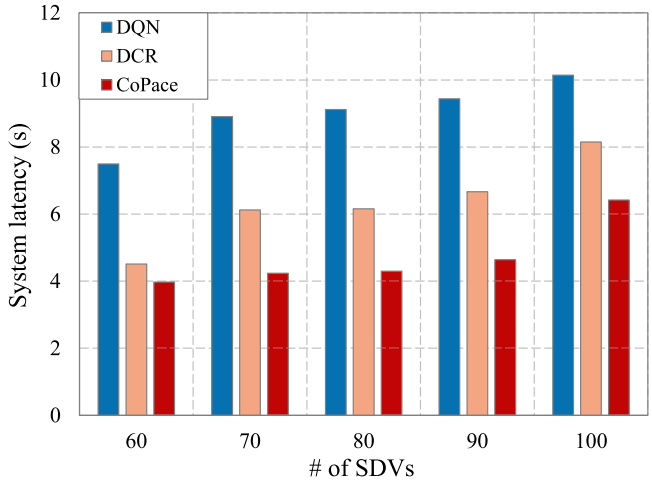



Fig. 7. The latency with different number of SDVs.

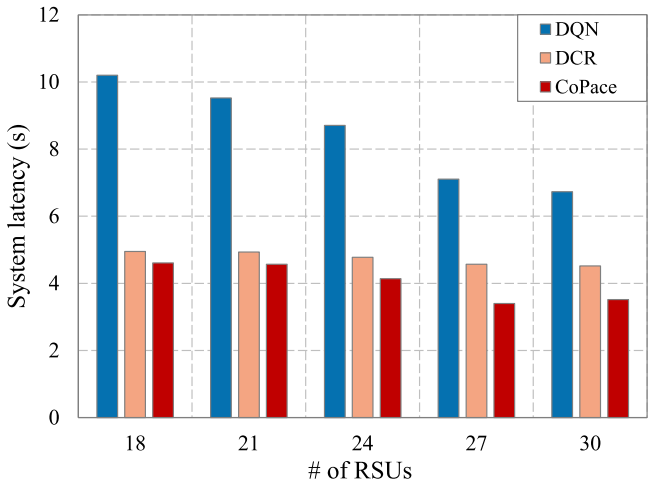



Fig. 8. The latency with different number of RSUs.

performance of CoPace is evaluated and compared with the other two baselines as

- DQN-based algorithm (DQN) [28]. In this scheme, offloading, caching, and computing and communication resources allocation policies are generated by the DQN agent.
- DDPG-based computation offloading and resource allocation without caching (DCR). In DCR, ES has no caching capabilities. The computation offloading and resource allocation strategies are derived from the DDPG agent. In addition, the network architecture of utilized DDPG in DCR is same to the CoPace.

First, Fig. 6 presents the convergence performance of our proposal with different learning rates. In the CoPace method, the actor is responsible for the selection of action by the policy, as well as the critic estimates the performance of the selected action. The learning rate can affect the convergence performance and convergence speed of the whole network, which is the crucial parameter for network training. It is observed that when the episode is from 0 to 100, the system latency drops dramatically. As the episode exceeds 100, the curve tends to be stable. Intuitively, we can see that the learning rate is lower, the better convergence performance of the CoPace is achieved. Furthermore, as the learning rate is 1e-3, the convergence speed is faster than where the learning rate is 1e-5 or 1e-7.

Fig. 7 illustrates the latency comparison with different numbers of SDVs. It is observed that the system latency of these three methods increases when the edge nodes serve more SDVs. This is because more SDVs share the computation and communication resources with ES. When the ES has the caching capabilities, the offloading time is reduced if the requested content is cached on the corresponding ES. As the number of SDVs rises, more high-frequency required data are cached on the RSUs that the offloading delay is reduced. When the number of SDVs is 60, CoPace outperforms 88.8 and 13.6 % than DQN and DCR, respectively. Moreover, in the DQN scheme, due to the limited capabilities on high-dimensional state and action space, it is difficult to explore and generate the optimal joint offloading, caching, and resource dispatching policies. In addition, CoPace achieves a better performance than DCR with the number of SDVs from 60 to 100.

Fig. 8 shows the latency comparison under the different number of RSUs. We can see that the system latency decreases linearly when more RSUs are deployed for performing computation tasks. This is because more RSUs can provision more computation resources, caching storage, and bandwidth. Especially, when the number of RSUs is 18, the performance of system

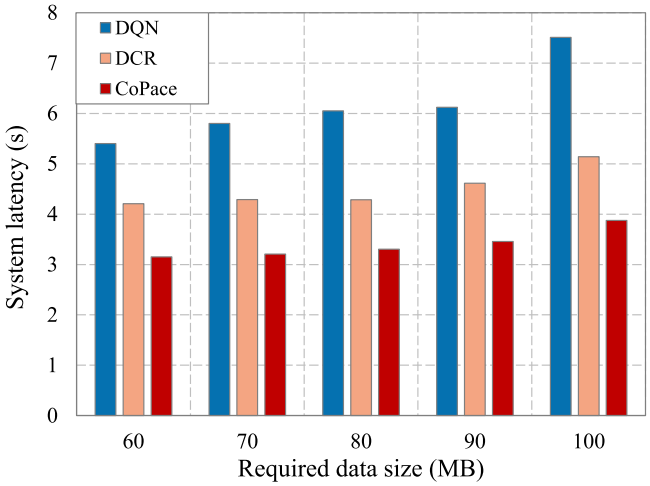



Fig. 9. The latency with different size of content.

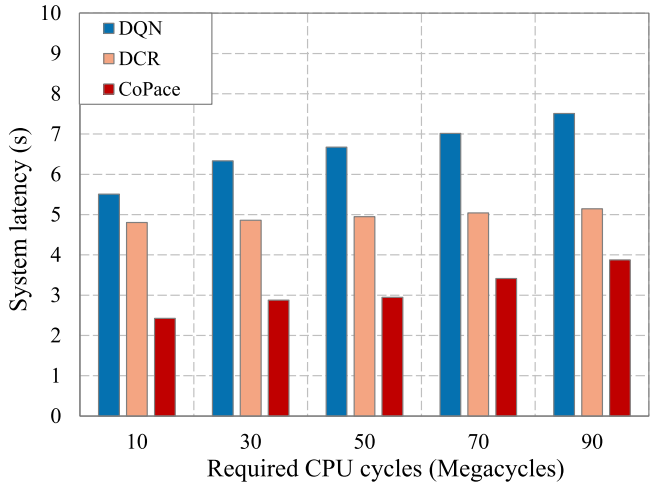



Fig. 10. The latency with different CPU cycles for execution.

latency for CoPace relative to DCR is about 7.4 %. Meanwhile, CoPace performs 2.2 times better than the DQN scheme. As the number of RSUs increases, the system latency reduces and more SDVs meet the demand for delay-sensitive vehicular applications. For example, as the number of RSUs increase to 30, the CoPace decreases the latency by 91.4 % and 28.5 % as compared with the DQN and DCR, respectively. Furthermore, with the caching capabilities, CoPace realizes the offloading latency decrease if the required content result is cached.

Fig. 9 provides the comparison of performance for the latency with different sizes of required data. It is observed that the increased size of data mainly affects the network transmission time, leading to the rise of system latency. The required data should be delivered to the ESs for execution, which causes time consumption during the transmission. The larger the size of data is, the offloading delay can be taken more. When the required data gets larger, the CoPace can obtain a better improvement of network latency than DQN. This is mainly because the DDPG-based approach gains better exploration performance in the action space, which derives the better action with a higher reward value. Particularly, when the size of required data is from 60 to 100 MB, the latency reduction ratio of the CoPace compared with DQN and DCR minimize from 71 % to 94 % and 34 % to 33 %, respectively. This is because that if the requested content is cached, the offloading delay can be sharply reduced. When requested content is cached on the ES, the corresponding processed results are returned to SDVs directly rather than offloading them.

Fig. 10 elaborates the latency with different required CPU cycles. It is seen that the latency of the three schemes increases when more CPU cycles are required. The key reason is that SDVs can hardly satisfy the computation-intensive vehicular services for executing them with more processing resources. Compared to the local processing on SDVs, computation resources are deployed on ESs in proximity to SDVs to meet the ultra-low response time requirements of delay-sensitive applications. When the computation requires 10 megacycles, the latency reduction ratio of the CoPace compared with DQN and DCR is 2.2 and 1.9 times, respectively. As the number of required CPU resources achieves, the CoPace is also superior to DQN and DCR on system delay. Furthermore, CoPace can outperform offloading latency over DCR due to the caching capabilities imposed on ESs that only deliver the processed results to SDVs if the required data is cached. Therefore, the offloading time is minimized.

## VI. Conclusion

In this paper, we proposed CoPace, a collaborative computation offloading and content caching method. Specifically, an offline spatial-temporal content popularity prediction algorithm, OSTP, is designed first to predict the future popularity level. According to OSTP, we then introduced the DDPG-based offloading, caching, and resource dispatch algorithm to optimize the latency. Based on two real-world datasets (i.e., telecom access records and taxi trajectories) in Shanghai, China, extensive experiments are conduct and numerical results demonstrate that CoPace is both more effective and well-performed than baselines.

## References

[1] K. Xiong, S. Leng, C. Huang, C. Yuen, and Y. L. Guan, "Intelligent task offloading for heterogeneous v2x communications," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 4, pp. 2226–2238, Apr. 2021.

[2] Z. Zhou, P. Liu, J. Feng, Y. Zhang, S. Mumtaz, and J. Rodriguez, "Computation resource allocation and task assignment optimization in vehicular fog computing: A contract-matching approach," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 3113–3125, Apr. 2019.

[3] S. Liu, L. Liu, J. Tang, B. Yu, Y. Wang, and W. Shi, "Edge computing for autonomous driving: Opportunities and challenges," *Proc. IEEE*, vol. 107, no. 8, pp. 1697–1716, Aug. 2019.

[4] L. U. Khan, I. Yaqoob, N. H. Tran, S. M. A. Kazmi, T. N. Dang, and C. S. Hong, "Edge-computing-enabled smart cities: A comprehensive survey," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 10 200–10 232, Oct. 2020.

[5] H. Yuan and M. Zhou, "Profit-maximized collaborative computation offloading and resource allocation in distributed cloud and edge computing systems," *IEEE Trans. Automat. Sci. Eng.*, vol. 18, no. 3, pp. 1277–1287, Jul. 2021.

[6] Z. Yang, Y. Liu, Y. Chen, and N. Al-Dhahir, "Cache-aided NOMA mobile edge computing: A reinforcement learning approach," *IEEE Trans. Wireless Commun.*, vol. 19, no. 10, pp. 6899–6915, Oct. 2020.

[7] Y. Liu, Q. He, D. Zheng, M. Zhang, F. Chen, and B. Zhang, "Data caching optimization in the edge computing environment," in *Proc. IEEE Int. Conf. Web Serv.*, 2019, pp. 99–106.

[8] S. Bi, L. Huang, and Y. A. Zhang, "Joint optimization of service caching placement and computation offloading in mobile edge computing systems," *IEEE Trans. Wireless Commun.*, vol. 19, no. 7, pp. 4947–4963, Jul. 2020.

[9] T. Zhang, X. Fang, Y. Liu, G. Y. Li, and W. Xu, "D2D-enabled mobile user edge caching: A multi-winner auction approach," *IEEE Trans. Veh. Technol.*, vol. 68, no. 12, pp. 12 314–12 328, Dec. 2019.
[10] Q. Li, Y. Zhang, Y. Li, Y. Xiao, and X. Ge, "Capacity-aware edge caching in fog computing networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 8, pp. 9244–9248, Aug. 2020.
[11] A. Ndikumana, N. H. Tran, D. H. Kim, K. T. Kim, and C. S. Hong, "Deep learning based caching for self-driving cars in multi-access edge computing," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 5, pp. 2862–2877, May 2021.
[12] S. Aradi, "Survey of deep reinforcement learning for motion planning of autonomous vehicles," *IEEE Trans. Intell. Transp. Syst.*, to be published. doi: 10.1109/TITS.2020.3024655.
[13] Y. Liu, H. Yu, S. Xie, and Y. Zhang, "Deep reinforcement learning for offloading and resource allocation in vehicle edge computing and networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 11, pp. 11 158–11 168, Nov. 2019.
[14] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis, "Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4005–4018, Jun. 2019.
[15] G. Qiao, S. Leng, S. Maharjan, Y. Zhang, and N. Ansari, "Deep reinforcement learning for cooperative content caching in vehicular edge computing and networks," *IEEE Internet Things J.*, vol. 7, no. 1, pp. 247–257, Jan. 2020.
[16] X. Hou *et al.*, "Reliable computation offloading for edge-computing-enabled software-defined IoV," *IEEE Internet Things J.*, vol. 7, no. 8, pp. 7097–7111, Aug. 2020.
[17] X. Pei, H. Yu, X. Wang, Y. Chen, M. Wen, and Y.-C. Wu, "NOMA-based pervasive edge computing: Secure power allocation for IoV," *IEEE Trans. Ind. Informat.*, vol. 17, no. 7, pp. 5021–5030, Jul. 2021.
[18] J. Zhao, X. Sun, Q. Li, and X. Ma, "Edge caching and computation management for real-time internet of vehicles: An online and distributed approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 4, pp. 2183–2197, Apr. 2021.
[19] J. Gao, S. Zhang, L. Zhao, and X. Shen, "The design of dynamic probabilistic caching with time-varying content popularity," *IEEE Trans. Mobile Comput.*, vol. 20, no. 4, pp. 1672–1684, Apr. 2021.
[20] Z. Yu, J. Hu, G. Min, Z. Zhao, W. Miao, and M. S. Hossain, "Mobility-aware proactive edge caching for connected vehicles using federated learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 8, pp. 5341–5351, Aug. 2021.
[21] X. Xiong, K. Zheng, L. Lei, and L. Hou, "Resource allocation based on deep reinforcement learning in IoT edge computing," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 6, pp. 1133–1146, Jun. 2020.
[22] Q. Luo, C. Li, T. H. Luan, and W. Shi, "Collaborative data scheduling for vehicular edge computing via deep reinforcement learning," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9637–9650, Oct. 2020.
[23] H. Ke, J. Wang, L. Deng, Y. Ge, and H. Wang, "Deep reinforcement learning-based adaptive computation offloading for MEC in heterogeneous vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 7, pp. 7916–7929, Jul. 2020.
[24] H. Chai, S. Leng, Y. Chen, and K. Zhang, "A hierarchical blockchain-enabled federated learning algorithm for knowledge sharing in internet of vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 7, pp. 3975–3986, Jul. 2021.
[25] L. Qi, X. Zhang, S. Li, S. Wan, Y. Wen, and W. Gong, "Spatial-temporal data-driven service recommendation with privacy-preservation," *Inf. Sci.*, vol. 515, pp. 91–102, 2020.
[26] Z. Lin, J. Feng, Z. Lu, Y. Li, and D. Jin, "DeepSTN: Context-aware spatial-temporal neural network for crowd flow prediction in metropolis," in *Proc. AAAI Conf. Artif. Intell.*, 2019, vol. 33, pp. 1020–1027.
[27] H. Xing, L. Liu, J. Xu, and A. Nallanathan, "Joint task assignment and resource allocation for D2D-enabled mobile-edge computing," *IEEE Trans. Commun.*, vol. 67, no. 6, pp. 4193–4207, Jun. 2019.
[28] V. Mnih *et al.*, "Playing atari with deep reinforcement learning," in *Proc. NIPS Deep Learn. Workshop*, 2013.
[29] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," in *Proc. 4th Int. Conf. Learn. Representations (ICLR)*, 2016.
[30] Y. Li, A. Zhou, X. Ma, and S. Wang, "Profit-aware edge server placement," *IEEE Internet Things J.*, to be published. doi: 10.1109/JIOT.2021.3082898.
[31] Y. Guo, S. Wang, A. Zhou, J. Xu, J. Yuan, and C.-H. Hsu, "User allocation-aware edge cloud placement in mobile edge computing," *Soft.: Pract. Exper.*, vol. 50, no. 5, pp. 489–502, 2020.
[32] S. Wang, Y. Guo, N. Zhang, P. Yang, A. Zhou, and X. Shen, "Delay-aware microservice coordination in mobile edge computing: A reinforcement learning approach," *IEEE Trans. Mobile Comput.*, vol. 20, no. 3, pp. 939–951, Mar. 2021.
[33] S. Liu, Y. Liu, L. M. Ni, J. Fan, and M. Li, "Towards mobility-based clustering," in *Proc. 16th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2010, pp. 919–928.
[34] H. Wu, J. Zhang, Z. Cai, F. Liu, Y. Li, and A. Liu, "Toward energy-aware caching for intelligent connected vehicles," *IEEE Internet Things J.*, vol. 7, no. 9, pp. 8157–8166, Sep. 2020.
[35] X. Wang, Z. Ning, S. Guo, and L. Wang, "Imitation learning enabled task scheduling for online vehicular edge computing," *IEEE Trans. Mobile Comput.*, to be published. doi: 10.1109/TMC.2020.3012509.
[36] M. Masoudi and C. Cavdar, "Device vs edge computing for mobile services: Delay-aware decision making to minimize power consumption," *IEEE Trans. Mobile Comput.*, vol. 20, no. 12, pp. 3324–3337, Dec. 2021.
[37] J. Zhang, Y. Zheng, D. Qi, R. Li, and X. Yi, "DNN-based prediction model for spatio-temporal data," in *Proc. 24th ACM SIGSPATIAL Int. Conf. Adv. Geographic Inf. Syst.*, 2016, pp. 1–4.
[38] J. Zhang, Y. Zheng, and D. Qi, "Deep spatio-temporal residual networks for citywide crowd flows prediction," in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017, pp. 1655–1661.

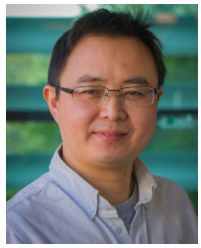
**Hao Tian** (Graduate Student Member, IEEE) received the B.S. degree in computer science and technology engineering in 2020 from the Nanjing University of Information Science and Technology, Nanjing, China, where he is currently working toward the master's degree in software engineering. His research interests include mobile edge computing, big data, Internet of Things, and machine learning.

**Xiaolong Xu** (Member, IEEE) received the Ph.D. degree in computer science and technology from Nanjing University, Nanjing, China, in 2016. He was a Research Scholar with Michigan State University, East Lansing, MI, USA, from 2017 to 2018. He is currently a Full Professor with the School of Computer and Software, Nanjing University of Information Science and Technology, Nanjing, China. He has authored or co-authored more than 100 peer-review articles in international journals and conferences, including IEEE TITS, IEEE TII, ACM TOIT, ACM TOMM, IEEE IOT, IEEE TCC, IEEE TBD, IEEE TCSS, IEEE TETCI, IEEE ICWS, ICSOC, etc. His research interests include edge computing, the Internet of Things (IoT), cloud computing, and big data. Prof. Xu is a Fellow of EAI (European Alliance for Innovation). He was the recipient of the Best Paper Award from the CBD 2016, IEEE CPSCom 2020, and SPDE 2020, the Distinguish Paper Award from EAI Cloudcomp 2019, the Best Student Paper Award from EAI Cloudcomp 2019, and the Best Session Paper Award from IEEE DSAA 2020.

**Lianyong Qi** (Member, IEEE) received the Ph.D. degree from the Department of Computer Science and Technology, Nanjing University, China, in 2011. He is currently a Full Professor with the School of Information Science and Engineering, Chinese Academy of Education Big Data, Qufu Normal University, China. He has authored or co-authored more than 50 articles, including the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, the IEEE TRANSACTIONS ON CLOUD COMPUTING, TBD, FGCS, *the Journal of Computational Social Science*, CCPE, ICWS, and ICSOC. His research interests include services computing, big data, and the Internet of Things.

**Xuyun Zhang** received the B.S. and M.E. degrees in computer science from Nanjing University, Nanjing, China, in 2011 and 2008, respectively, and the Ph.D. degree from the University of Technology Sydney, Ultimo, NSW, Australia, in 2014, He worked as a Postdoctoral Fellow with the Machine Learning Research Group, NICTA (currently, Data61, CSIRO), Sydney, NSW, Australia. He is a Senior Lecturer with the Department of Computing, Macquarie University, Sydney. His primary research interests include Internet of Things and smart cities, big data, cloud computing, scalable machine learning and data mining, data privacy and security, and web service technology.

**Wanchun Dou** (Member, IEEE) received the Ph.D. degree in 2001. He is a Full Professor with the State Key Laboratory for Novel Software Technology, Nanjing University. To date, he has Chaired four National Natural Science Foundation of China projects and authored or co-authored more than 100 articles in international journals and conferences. His research interests include big data, cloud computing, and service computing.



**Shui Yu** (Senior Member, IEEE) received the Ph.D. degree from Deakin University, Geelong, VIC, Australia, in 2004. He is currently a Professor with the School of Computer Science, University of Technology Sydney, Sydney, NSW, Australia. He has authored or co-authored two monographs and edited two books, more than 300 technical papers, including top journals and top conferences, such as the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, the IEEE TRANSACTIONS ON COMPUTERS, the IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, the IEEE TRANSACTIONS ON MOBILE COMPUTING, the IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, the IEEE TRANSACTIONS ON EMERGING TOPICS IN COMPUTING, the IEEE/ACM TRANSACTIONS ON NETWORKING, and INFOCOM. His research interests include security and privacy, networking, big data, and mathematical modeling. Prof. Yu is currently the Editorial Boards of IEEE COMMUNICATIONS SURVEYS & TUTORIALS, *IEEE Communications Magazine*, the IEEE INTERNET OF THINGS JOURNAL, IEEE COMMUNICATIONS LETTERS, and the IEEE TRANSACTIONS ON COMPUTATIONAL SOCIAL SYSTEMS.

**Qiang Ni** (Senior Member, IEEE) received the B.Sc., M.Sc., and Ph.D. degrees in engineering from the Huazhong University of Science and Technology, China. He is currently a Professor and the Head of the Communication Systems Group, School of Computing and Communications, Lancaster University, Lancaster, U.K. He has authored or co-authored more than 200 articles in these areas. His research interests include the area of future generation communications and networking, including green communications and networking, millimeter-wave wireless communications, cognitive radio network systems, non-orthogonal multiple access (NOMA), heterogeneous networks, 5G and 6G, SDN, cloud networks, energy harvesting, wireless information and power transfer, IoTs, cyber physical systems, AI and machine learning, big data analytic, and vehicular networks. He was an IEEE 802.11 Wireless Standard Working Group Voting Member and a contributor to various IEEE wireless standards.