Computation offloading through mobile vehicles in IoT-edge-cloud network

Jun Long
Central South University
Yueyi Luo
Central South University
Xiaoyu Zhu (zhuxiaoyu@csu.edu.cn)
Central South University https://orcid.org/0000-0002-6815-3754
Entao Luo
Hunan University of Science and Engineering
Mingfeng Huang
Central South University

Research

Keywords: Computation offloading, Mobile vehicles, IoT-edge-cloud network, deep reinforcement learning

Posted Date: September 23rd, 2020

DOI: https://doi.org/10.21203/rs.3.rs-41747/v2

Version of Record: A version of this preprint was published on December 1st, 2020. See the published version at https://doi.org/10.1186/s13638-020-01848-5.

RESEARCH

Computation offloading through mobile vehicles in IoT-edge-cloud network

Jun Long[1,2], Yueyi Luo[1,3], Xiaoyu Zhu[1*], Entao Luo[4] and Mingfeng Huang[1]

*Correspondence: zhuxiaoyu@csu.edu.cn
[1]School of Computer Science and Engineering, Central South University, 410083 Changsha, China
Full list of author information is available at the end of the article

Abstract

With the developing of Internet of Things (IoT) and mobile edge computing (MEC), more and more sensing devices are widely deployed in the smart city. These sensing devices generate various kinds of tasks, which need to be sent to cloud to process. Usually, the sensing devices do not equip with wireless modules, because it is neither economical nor energy saving. Thus, it is a challenging problem to find a way to offload tasks for sensing devices. However, many vehicles are moving around the city, which can communicate with sensing devices in an effective and low-cost way. In this paper, we propose a computation offloading scheme through mobile vehicles in IoT-edge-cloud network. The sensing devices generate tasks and transmit the tasks to vehicles, then the vehicles decide to compute the tasks in the local vehicle, MEC server or cloud center. The computation offloading decision is made based on the utility function of the energy consumption and transmission delay, and the deep reinforcement learning technique is adopted to make decisions. Our proposed method can make full use of the existing infrastructures to implement the task offloading of sensing devices, the experimental results show that our proposed solution can achieve the maximum reward and decrease delay.

Keywords: Computation offloading; Mobile vehicles; IoT-edge-cloud network; deep reinforcement learning

1 Introduction

Wireless sensor networks (WSNs) is widely deployed in the smart city. There exists a great amount of sensing devices in the urban environment, the sensing devices in WSNs can sense the environmental data [1–4]. The collected sensing data can be processed and sent to the remote server to make decisions, which can help improve the city management [5–7].

The quickly developing micro-processing technologies are applied into the sensing devices, the computation and communication capacities of the sensing devices are becoming more and more stronger [8–10]. Different kinds of sensing devices deployed in smart city, such as water pollution sensor, temperature sensor, humidity sensor, PM2.5 sensor, intelligent garbage can, intelligent streetlight. The sensing devices are the basis of the Internet of Things (IoT) [11–14]. The sensing devices can be applied into a large number of application scenarios, such as the environmental pollution monitoring and traffic monitoring. For example, the water pollution sensors are deployed into urban water pipelines, which are used to sense and collect the status of water resources in the urban environment. Usually, the large amount of sensing data needs to be processed before making decisions. The sensing data processing can help people understand urban environment.

Processing the sensing data can generate a large number of real-time computing tasks. For example, the data deduplication task can remove the duplicated data. Furthermore, the data compression task can reduce the communication overhead in the process of data transmission. In addition, there is a large number of artificial intelligence algorithms [15, 16], such as augmented reality (AR), face recognition, task classification, etc. However, the sensing devices exist shortcomings such as low computing power, limited storage capacity and limited battery power. Therefore, it is important to offload the computation tasks to servers with strong computing power, the energy of sensing devices can be saved through computation offloading.

However, the sensing devices contain some communication problems. (1) The sensing devices are usually very small, and the communication range is limited. (2) There is a large amount of sensing devices which are widely distributed in the city, the sensing devices can be added or moved easily. Therefore, it is impractical to deploy fixed networks for sensing devices. For example, the garbage cans can be added or moved to other locations due to the demands. Thus, it is not cost-effective to apply an expensive wired network for these sensing devices. (3) The mobile communication technology is quickly developing, such as 5G technology. However, the SIM card cost and communication fee are too high for a sensing device. Therefore, many existing and deployed sensing devices usually do not use the cellular network. Totally, it is difficult to find a cost-effective way for sensing devices to complete the computation tasks.

The sensing devices have limited computation power and communication range, if the tasks are processed in the sensing devices, the sensing devices may consume a lot of energy and enter the sleepy mode. Furthermore, the deployment of communication system for these sensing devices has a large cost. Researchers proposed to adopt mobile vehicles to collect data sensed by the sensing devices distributed in the city. Similarly, there is a possible way to solve the task offloading problem, the moving mobile vehicles in the city can act as the task mules. There is a large number of vehicles moving around the city. When the vehicles pass by the sensing devices, the sensing devices send its computation task to the vehicle. The sensing devices and vehicles can communicate with each other with a low energy cost, which leads to a low-cost and effective way to offload the computation tasks to the vehicles.

Numerous sensing devices and moving vehicles generate the IoT in smart city. The mobile edge computing (MEC) server on the roadside provides conditions for short-distance edge computing, while cloud computing can provide powerful computing capabilities remotely. In order to support computation offloading for a large number of IoT devices, the cooperation between IoT, edge and cloud is required. Therefore, it is very important to study the computation offloading problem in IoT-edge-cloud network. In this paper, we attempt to apply vehicles to help offload the computation tasks of sensing devices. There are many mobile vehicles travelling around the city. In the future, the vehicles will gradually develop into the intelligent platform integrating various communication functions. The vehicles can communicate with the sensing devices using short range communication, the mobile vehicles can communicate with more powerful servers. Thus, it is a possible solution to use mobile vehicles to complete sensing devices' computation offloading tasks.

In order to make full use of vehicles in computation offloading of mobile edge network, we consider three layers in our system architecture: IoT device layer, mobile

edge computing server and cloud center layer. The sensing devices offload the computation tasks to mobile vehicles through short range communication technology. When mobile vehicles receive the computation tasks, vehicles choose to compute the tasks locally or transmit the tasks to the MEC servers. The MEC server can compute the tasks transmitted by the mobile vehicles.

The mobile edge computing is proposed to provide computation services for edge devices. However, a large number of sensing devices cannot connect to Internet, thus it is impossible to offload the computation tasks to MEC servers. In order to solve this problem, we propose to offload the computation tasks to vehicles, and the vehicles can offload the computation tasks to MEC server when the vehicles cannot process the tasks. Our objective is to design a computation offloading scheme for sensing devices. Through jointly coordinate with sensing devices, mobile vehicles and MEC servers, we propose a computation offloading scheme through vehicles in IoT-edge-cloud network.

Artificial intelligence [17] is an important tool to solve the computation offloading problem efficiently, the deep reinforcement learning plays an important role in artificial intelligence. Deep reinforcement learning has a strong cognitive ability to the real environment and has been widely used in vehicular network. In large-scale network, deep reinforcement learning can significantly improve the decision-making speed and realize the task offloading for a large number of vehicles. With the explosive growth of sensing data, deep reinforcement learning can quickly learn knowledge from the environment, especially in MEC-enabled systems, which can quickly select the appropriate MEC server for vehicles to offload tasks. Therefore, it is very important to develop a deep reinforcement learning-based computation offloading method in heterogeneous network architecture.

The actor-critic (AC) framework is widely used in reinforcement learning algorithms. The AC framework integrates the advantages of policy-based solution and value-based solution, AC algorithm is the most commonly considered algorithm when solving practical problems. In order to speed up the training process of the AC algorithm, asynchronous advantage actor-critic (A3C) algorithm is proposed. A3C puts AC into multiple threads for synchronous training, which can effectively use the resources of computer and improve the training efficiency. The server has multiple cores which can run multiple AC algorithms, the results are sent to the main network from multiple cores. The main network combines the results and decreases the relevance, which is better for the convergence of the algorithm. In order to minimize the offloading cost of tasks, this paper proposes a computation offloading scheme based on deep reinforcement learning in the IoT-edge-cloud network.

The main contributions of this paper are summarized as follows:

(1) A computation offloading scheme through mobile vehicles (COTV) in IoT-edge-cloud network is proposed in this paper, which solves the computation task offloading problem for widely deployed sensing devices in the smart city. The COTV scheme can find a solution for the computation offloading problem. There are a lot of sensing devices deployed in the smart city, it will consume a lot of energy to process these tasks. In this paper, we define the computation offloading problem for sensing devices. As many vehicles are moving around the city, we attempt to apply the mobile vehicles to communicate with sensing devices. The sensing devices can send their tasks to vehicles in a low-cost and energy-efficient way.

(2) The system architecture is studied in COTV scheme, which consists of sensing device layer, mobile vehicle layer, MEC server layer and cloud center layer. The tasks are computed at mobile vehicle, MEC servers or cloud center. The total system cost includes the processes of task collection, task transmission and task execution. The energy consumption and latency are jointly considered in our COTV scheme.

(3) The deep reinforcement learning method is adopted to train the model, the reward is the total cost of the whole system in a long time period. Through extensive experiments, our COTV scheme achieves better performance than the other compared schemes.

The rest of this paper is organized as follows: In Section 2, we introduce the related work. The network model and problem statement are presented in Section 3. Then, the method is introduced in Section 4. The experiment results of COTV scheme is presented in Section 5. Finally, Section 6 gives the conclusions.

2 Background and related work

In order to provide low processing delay for devices, MEC is proposed to improve the performance of the system [18–20]. The MEC servers can process many tasks. Meanwhile, the MEC servers has smaller distance with devices, thus the processing delay can be largely decreased compared with cloud server. The IoT-edge-cloud network can make full use of IoT, MEC computing and cloud computing [21–26]. Wang et al. [27] proposed a task offloading and resource allocation algorithm in a heterogeneous network. The network divided different MEC servers into multiple layers, the objective was to minimize the overall system latency. Through assigning tasks and allocating resources in heterogeneous layers, the latency was minimized and the processing rate was improved. In [28], the authors aimed at task offloading and service orchestration based on software defined network technology, the tasks were offloaded to MEC server or cloud server according to the required resource and allowed latency. The service orchestration technology could decrease the transmission data amount, the propose scheme can decrease the latency and consumed energy. In [29], the authors proposed a task offloading and resource allocation algorithm based on software defined network technology in ultra-dense network, the battery capacity of the devices was considered as an impact factor of task offloading decision. However, none of these works consider the offloading costs of sensing devices, it is better to design a cost-effective way to solve the computation offloading problem for sensing devices.

Mobile vehicles have been used in different kinds of application scenarios [30, 31]. Liu et al. [32] took the vehicles as mobile edge servers, which could process the latency-sensitive tasks of some hot spots. The objective was to maximize the number of finished tasks. In 5G network, the subchannel assignment and computation offloading generate new challenging problems for vehicles, the authors in [33] proposed to offload tasks through cellular network or vehicle to vehicle according to the estimated transmission delay. Hoang et al. [34] studied the task offloading problem in a MEC-vehicular network, which considered that the vehicles were random moving. The offloading decision was made according to the estimated cost by predicting the vehicle trajectory. Peng et al. [35] proposed a resource management solution based on MEC in autonomous vehicular networks. Zhuang et al. [36] surveyed a large

number of software-defined networking and network function virtualization solutions in internet of vehicles. Misra et al. [37] generated a task offloading problem and a computation results downloading problem separately, because the vehicles were continuously moving. A number of road site units were selected as the path to transmit the task to the fog server, the vehicle mobility was predicted to enable the path selection of downloading the task results.

A large number of sensing devices are deployed in smart city, they generate a lot of tasks. In order to make intelligent decisions, the tasks need to be transmitted to the cloud server. However, the sensing devices usually can not directly connect with cloud center. Thus, the sensing devices should find a way to transmit the task to cloud center. Recently, some researches proposed to make vehicles as data mules to transmit data. Luo et al. [38] proposed a data collection scheme in smart city using vehicles, the sensing devices were widely deployed to detect the status of basic infrastructures, the vehicles sent sensing data from sensing devices to data center, then the data center could make scientific decisions. The authors of [39] proposed to collect data through unmanned aerial vehicle in a trust way, the energy consumption of system was decreased by optimizing the unmanned aerial vehicle's trajectory. Liu et al. [40] orchestrated sensing data as service in edge-cloud network, which could reduce the service response delay and data redundancy. The vehicles were used to transmit services to the base stations. He et al. [41] considered the collaboration of cloud computing and local computing and proposed vehicle cloud computing. Through coordinating the computing resources of vehicle cloud and remote cloud, the proposed scheme provided real-time service for users. The computation resources of road side units were also considered in the vehicle cloud. However, the computation offloading problem in IoT-edge-cloud network is quite complex, none of these researches focus on the IoT-edge-cloud network.

Recently, many researchers adopt machine learning techniques in the computation offloading problem. Lu et al. [42] considered the fine-grained task offloading problem in the heterogeneous MEC network. The mobile applications can be divided into multiple subtasks according to data dependencies, the subtasks can be offloaded to different servers for execution. The task offloading based on deep reinforcement learning can reduce the energy consumption and execution delay. Qi et al. [43] focused on the internet of vehicles and proposed a knowledge-driven offloading scheme based on A3C algorithm. In order to solve the cooperative computation offloading problem in blockchain-enabled MEC system, Feng et al. [44] formulated the joint optimization as a Markov decision process and solved the problem based on A3C algorithm. However, the machine learning solutions adopted in these researches do not consider the computation offloading problem in IoT-edge-cloud network. The computation offloading techniques can be further explored considering the blockchain [45, 46] or privacy preserving technique [47, 48].

Many researchers proposed many task offloading researches. However, most researches focused on the task offloading problem of edge devices in MEC or cloud computing, the task offloading problem of sensing devices is not discussed. Different from these works, we propose a task offloading scheme through mobile vehicles. Our research focuses on the IoT-edge-cloud network, there are many sensing devices in heterogeneous network. The computation tasks are intensive, and the tasks require

low latency. There are three different platforms that can be selected to offload, which are mobile vehicles, MEC servers and cloud center, respectively. Thus, this problem is quite different from other existing researches.

3 The system model and problem statement

3.1 The energy consumption model

The network model contains IoT devices, MEC servers and cloud center. As Fig. 1 shows, our COTV model contains three layers.

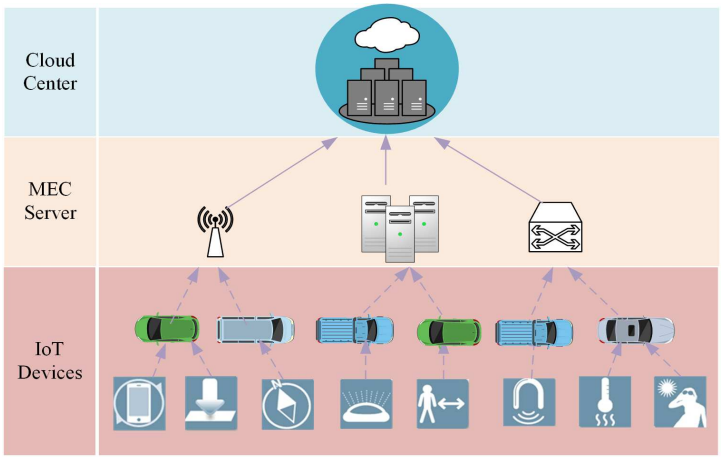

Figure 1 The structure of COTV

(1) IoT Device Layer (IDL). The IDL layer contains sensing devices and mobile vehicles. The sensing devices upload computation tasks. The sensing devices are usually deployed all over the city, such as the intelligent trash can, streetlamp. These sensing devices sense and collect data in the city, and upload data to cloud center, the cloud center makes scientific decision based on these data. However, as the sensing devices are widely deployed in the city, and it will result a large cost to deploy a communication network for these sensing devices. The mobile vehicles are moving along the roads. The vehicles can process the computation tasks locally, or they can upload the computation tasks to the MEC servers. The mobile vehicles act as the task mules to receive the computation tasks of sensing devices. The vehicles move in the city as its predefined route, which can cover most areas of the city, and they can receive the computation tasks effectively. We assume that the vehicles are integrated with short range communication module and wireless communication module. When the vehicles communicate with sensing devices, they use short range communication which consumes smaller energy. When the vehicles communicate with MEC servers or cloud center, they use wireless communication.

(2) Mobile Edge Computing Server Layer (MSL). The MSL layer contains MEC servers. The MEC server can process the computation tasks. The MEC servers close to the vehicles, which can process the latency-sensitive tasks, and they contain wireless and wired communications. When the MEC servers communicate with vehicles, they use wireless communication, when they communicate with cloud center, they use wired communication.

(3) Cloud Center Layer (CCL). The CCL layer contains one cloud center. The cloud server has large computation power and can process large number of computation tasks. The cloud center is far away from the vehicle, thus it needs longer time to process the task.

3.2 The computation offloading model

In this paper, we propose a computation offloading scheme through vehicles (COTV) in IoT-edge-cloud network. In IoT-edge-cloud network, there are many IoT device and mobile vehicles in IoT, which generate a large number of computation tasks. However, the devices in IoT have small computation power. The cloud can provide large computation resources, which can help process the computation tasks. However, the data transmission of long distances will result in some delay. In addition, the edge can provide low-delay computation resources for IoT devices.

In our computation model, mobile vehicle can receive, process and transmit computation tasks. Assume that there are some computation tasks. For each task, the vehicles should make a computation offloading decision, the task can be processed in the vehicle, MEC server, or cloud center. The computation offloading operation is made based on the deep reinforcement learning algorithm.

3.3 Problem statement

The overall objective is to design a computation offloading scheme through vehicles. Each task is executed in the vehicle, MEC server or cloud center. Different from the other optimization problems, we consider the overall energy consumption and delay of the task in our optimization problem. We design a utility function to improve the quality of service, which is the weight sum of the energy consumption and delay. Assume that the total energy consumption is E, which includes the communication energy consumption and task execution consumption. The total latency is D, which includes the communication latency and task execution latency. Our optimization objective is to minimize the total energy consumption and latency. Thus, the total optimization function is defined as

$$min(F) = min(\lambda E + (1 - \lambda)D). \tag{1}$$

4 Methods

4.1 The design of COTV scheme

In this paper, we propose a computation offloading scheme through vehicles (COTV) in IoT-edge-cloud network. We consider a network consisted of sensing devices, mobile vehicles, MEC servers and cloud center. The sensing devices can transfer the computation tasks to mobile vehicles. We propose to use vehicles to make computation offloading decisions, the mobile vehicles receive the tasks from sensing devices through opportunistic communication. In this paper, we propose the computation offloading scheme through mobile vehicles in IoT-edge-cloud network. The tasks can be processed in the vehicle, MEC server or cloud center.

(1) The sensing devices transmit the tasks to the vehicles through opportunistic communication. The vehicles should make task decisions when they receive tasks. We assume that the task is an independent object, which can only be fully processed

by vehicle, MEC server or cloud center. The task can not be divided into several parts.

(2) The task is processed in the vehicle. The vehicle first receives the task from the sensing device, if the vehicle decides to process the task locally, the task is added into the cache queue. When the task is processed locally, the final results of the tasks are sent to the cloud center.

(3) The task is offloaded to the MEC server. If the vehicle decides to offload the task to the MEC server. Then the vehicle needs to select one MEC server to offload the task, and sends the required resource to the MEC server. After selecting the object MEC server, the vehicle offloads the task to it. Finally, the task results are sent to the cloud center.

(4) The task is offloaded to the cloud center. If the vehicle decides to offload the task to cloud center, the vehicle transmits the required resource to cloud center, the task is added into the cache queue of cloud center. Then the task is processed in the cloud center. Finally, the cloud center obtains all the processing results of the tasks.

4.2 Tasks transferring through vehicles

In the smart city, the sensing devices are widely deployed in the basic infrastructures. The sensing devices are responsible for generating different kinds of tasks. These tasks need to be offloaded and processed in an appropriate platform. We assume that each sensing device generates a task, the processing results of these tasks are aggregated in the cloud center. The cloud center is responsible for making scientific decisions based on task results.

Assume that all the sensing devices are partitioned into different areas, each area contains some sensing devices, the sensing devices in the same area can communicate with each other. The sensing devices in one area can be divided into two categories: direct sensing devices and indirect sensing devices. For the direct sensing devices, they are within the communication range of the vehicles, and they act as the relay to help other sensing devices to transmit tasks. When the vehicles passing by the direct sensing devices, they can transmit the tasks to the vehicles through short range communication. For the indirect sensing devices, they need to transmit their tasks to the direct sensing devices through multi-hop communication. Each indirect sensing device needs to know the next-hop sensing device. The task transmission will cause the route cost. However, the computation cost is much larger, thus we do not consider route cost in our COTV scheme.

Many sensing devices cannot connect to Internet, some sensing devices do not have wifi adaptor due to economic considerations. In addition, it consumes a lot of energy when transmitting tasks through wireless communication, which leads to the death of the sensing devices. The sensing devices are equipped with short range communication module, which consume smaller energy through opportunistic communication. There are many vehicles moving in the city, which are equipped with short range communication modules.

The sensing devices do not process the tasks, we make full use vehicles to receive and process tasks. The vehicles are moving along the roads, when the vehicle is within the communication of the vehicle, the sensing device transmit its task to

the vehicle through short range communication. Let T be the duration time, which can be divided into multiple time slots. At time slot t, we assume that each sensing device has one computation task $S_t = \{l_t, x_t\}$. l_t is the needed input data to process the task, such as the program code and input parameters. x_t is the computation resources needed to finish the computation task. l_t and x_t can not only obtain the basic attributes of task, such as the computation and communication requirements, but also simplify the evaluation of execution delay and energy consumption.

We adopt mobile vehicles to receive the computation tasks from sensing devices. The vehicles can communicate with MEC servers and cloud center, the MEC servers can communicate with the cloud center. When the vehicle receives the task, the vehicle should make the offloading decision. The task is processed in the vehicle, MEC server or cloud center, and transmit the task to the cloud center. Finally, the cloud center gathers all the task processing results and makes scientific decision. In this paper, we focus on the total cost of the task from the sensing device to the cloud center.

4.3 Energy consumption and delay modeling

The execution delay and energy consumption are very important. In the offloading process, we mainly consider two main steps: i) transfer the input data, ii) compute tasks. The main objective of our approach is: minimize the energy consumption and execution latency. Hereafter, we will evaluate with close formulas the above parameters.

(1) Sensing device computation model

The sensing devices contain the trash can, streetlamp, and so on. In our network, the sensing devices are responsible for generating tasks. The sensing devices locate at the first level of the network. Each sensing device generate one task, the sensing device is not responsible for process task.

The sensing device transfers its computation task to the vehicle through opportunistic communication. The task is denoted as $S_t = \{l_t, x_t\}$. The delay of computation task S_t in the sensing device level is denoted as D_t^s, the data uplink transmission rate from the sensing device to the vehicle is denoted as $\alpha_t^{s,v}$. Thus, the delay D_t^s is the time of uploading computation tasks from sensing device to vehicle, which is computed as

$$D_t^s = l_t / \alpha_t^{s,v}. \tag{2}$$

The consumed energy E_t^s is the transmission cost of uploading computation tasks from sensing device to vehicle, E_t^s is given by

$$E_t^s = C_s D_t^s, \tag{3}$$

where C_s is the transmission power consumption from sensing device to vehicle per unit time.

(2) Vehicle computation model

The vehicles play very important roles in the network. First, the vehicle layer is between the sensing layer and MEC server layer, the vehicles can receive the tasks

from sensing devices in a low-cost and effective way. Then, the vehicles can make full use of its own computation power and compute some part of tasks for sensing devices. Finally, the vehicles transmit the processing results to the cloud center.

If a vehicle receives task $S_t = \{l_t, x_t\}$, the vehicle decides to compute the task locally. The delay of the computation task is denoted as D_t^v, the delay contains two parts: i) the delay of processing tasks in the vehicle, ii) the delay of uploading results from vehicle to cloud center.

We denote f_v as the computation power of the vehicle, the computation power is the CPU cycles in one second. Thus, the delay of processing tasks in the vehicle $D_t^{v,p}$ is denoted as

$$D_t^{v,p} = x_t / f_v. \tag{4}$$

Assume the amount of processed results is $x_{t,p}$, the data uplink transmission rate from vehicle to cloud center is denoted as $\alpha_t^{v,c}$. The delay of uploading results from vehicle to cloud center $D_t^{v,u}$ is denoted as

$$D_t^{v,u} = x_{t,p} / \alpha_t^{v,c}. \tag{5}$$

Thus, the total delay in the vehicle is given by

$$D_t^v = D_t^{v,p} + D_t^{v,u}. \tag{6}$$

The consumed energy E_t^v includes the computation cost in the vehicle and transmission cost of uploading computation results from vehicle to cloud center, E_t^v is denoted as

$$E_t^v = C_v^a {D_t}^{v,p} + C_v^u D_t^{v,u}, \tag{7}$$

where C_v^a is the computation power consumption of vehicle per unit time, C_v^u is the transmission power consumption from vehicle to cloud center per unit time.

(3) MEC computation model

The MEC server layer is between the vehicle layer and cloud center level, the MEC servers bring the computation resources to the edge network. The computation power of MEC server is larger than the vehicle, but smaller than the cloud center. The MEC server can share the responsibility of vehicles, because the vehicles have smaller computation power. The MEC servers can communicate with vehicles and cloud center through wireless communication. After receiving the computation task from the vehicles, the MEC server sends the task to the cloud center. As the MEC servers are closer to the vehicles and sensing devices, thus it takes smaller time to transmit the task from vehicle to MEC server than cloud center.

If the vehicle chooses to offload the task to a MEC server, the delay of the computation task is denoted as D_t^m, the delay contains two parts: i) the delay of processing tasks in the MEC server, ii) the delay of uploading results from MEC server to cloud center.

We denote f_m as the computation power of the MEC server, the computation power is the CPU cycles in one second. Thus, the delay of processing tasks in the MEC server $D_t^{m,p}$ is denoted as

$$D_t^{m,p} = x_t / f_m. \tag{8}$$

Assume the amount of processed results is $x_{t,p}$, the data uplink transmission rate from MEC server to cloud center is denoted as $\alpha_t^{m,c}$. The delay of uploading results from MEC server to cloud center $D_t^{m,u}$ is denoted as

$$D_t^{m,u} = x_{t,p} / \alpha_t^{m,c}. \tag{9}$$

Thus, the total delay in the MEC server is denoted as

$$D_t^m = D_t^{m,p} + D_t^{m,u}. \tag{10}$$

The consumed energy E_t^m includes the computation cost in the MEC server and transmission cost of uploading computation results from MEC server to cloud center, E_t^m is given by

$$E_t^m = C_v^\beta D_t^{m,p} + C_v^\epsilon D_t^{m,u}, \tag{11}$$

where C_v^β is the computation power consumption of MEC server per unit time, C_v^ϵ is the transmission power consumption from MEC server to cloud center per unit time.

(4) Cloud computation model

The cloud center can receive and process the tasks from the vehicles through wireless communication, and the computation power of cloud center is the largest in all the layers.

If the vehicle chooses to offload the task to a cloud center, the delay of the computation task is denoted as D_t^c, the delay only includes the time of processing tasks in the cloud center.

We denote f_c as the computation power of the cloud server, the computation power is the CPU cycles in one second. Thus, the total delay in the cloud center D_t^c is denoted as

$$D_t^c = x_t / f_c. \tag{12}$$

The consumed energy E_t^c includes the computation cost in the cloud center, which is denoted as

$$E_t^c = C_v^\gamma D_t^c, \tag{13}$$

where C_v^γ is the computation power consumption of cloud center per unit time.

The cloud center also receives the task processing results from vehicles and MEC servers. Based on all the processing results, the cloud center can make scientific decisions.

(5) Total cost optimization model

The overall objective is to design a computation offloading scheme for IoT sensing devices through vehicles. Each task is executed in the vehicle, MEC server and cloud center. Our optimization objective is to minimize the total energy consumption and latency.

The offloading policy of task is $a_t = \{a_t^v, a_t^m, a_t^c\}$, in which $a_t = \{a_t^v = 1, a_t^m = 0, a_t^c = 0\}$ denotes that the task is processed in the vehicle, $a_t = \{a_t^v = 0, a_t^m = 1, a_t^c = 0\}$ denotes that the task is offloaded to MEC server, $a_t = \{a_t^v = 0, a_t^m = 0, a_t^c = 1\}$ denotes that the task is offloaded to cloud center.

The total latency is D_t, which includes the transmission latency and task execution latency. D_t is computed as

$$D_t = D_t^s + a_t^v D_t^v + a_t^m D_t^m + a_t^c D_t^c. \tag{14}$$

Assume that the total energy consumption is E_t, which includes the transmission energy consumption and task execution consumption. E_t is computed as

$$E_t = E_t^s + a_t^v E_t^v + a_t^m E_t^m + a_t^c E_t^c. \tag{15}$$

The total optimization function is defined as

$$F_t = \lambda D_t + (1 - \lambda) E_t, \tag{16}$$

where λ is a parameter to dynamic change the weight between delay and energy consumption in the optimization function.

Thus, the goal and constraints are defined as:

$$\begin{aligned} & min \sum_{t \in T} F_t \\ \text{s.t. } & C_1 : a_t^v, a_t^m, a_t^c \in (0, 1) \\ & C_2 : a_t^v + a_t^m + a_t^c = 1, \end{aligned} \tag{17}$$

where C_1 and C_2 means that each task can be processed at vehicle, MEC server or cloud center.

4.4 Computation offloading algorithm

Assume that the sensing devices generate multiple tasks, the sensing devices transfer tasks to vehicles. The vehicles determine the offloading decisions of these tasks. The tasks can be processed in vehicle, offloaded to MEC server or cloud center. The system information is transferred to the agent to obtain the feedback of the offloading strategy. In the task offloading process using deep reinforcement learning, the reward is obtained after taking an action. We will introduce the state, action and reward as follows.

State s_t: The state s_t is defined to describe environment state for the task of vehicle in time slot t, which includes the vehicle information, MEC server information and cloud server information. When the agent generates a computation task in time

slot t, it records the task status as well as the available computational resource information. The MEC server information is the computation capacity. Hence, we use $s_t = \{l_t, x_t, f_m, f_c\}$ to represent the vehicular edge computing state. The status contains the following:

(1) Vehicle information $\{l_t, x_t\}$: l_t is the input data of task, x_t is the needed computing resource for executing the task.

(2) MEC server information f_m: f_m is the computation capacity of MEC server.

(3) Cloud server information f_c: f_c is the computation capacity of cloud server.

Action a_t: The agent takes an action a_t at time slot t according to the current state s_t, the computation policy π implies that the vehicle computes the task locally, or offloaded to the MEC server, or offloaded to the cloud center.

Reward r_t: For each step, the agent will get a reward r_t in a certain state s_t after executing each possible action. In general, the reward function should be related to the objective function. Consequently, the objective of our optimization problem is to get the minimal sum cost and the goal of deep reinforcement learning is get the maximum reward, so the value of reward should be negatively correlated to the size of the sum cost, we define the immediate reward as normalized r_t, where A_1 and A_2 are constants. r_t is computed as

$$r_t = \frac{A_1}{A_2 + F_t}. \tag{18}$$

The task offloading environment has many system states, which needs to be determined by the current situation of the system. However, it is difficult to use traditional method to solve this complex task. Asynchronous advantage actor-critic (A3C) is a novel deep reinforcement technique, which can receive high dimensional data and output optimal operation for each input data. A3C has a good performance in solving high-dimensional state and action decision. Compared with the other deep reinforcement learning algorithms, such as actor-critic and Q-learning, A3C takes the advantages of these two algorithms. As A3C uses asynchronous participants to learn, which can accelerate the learning speed and make the training strategy more diverse.

The data center can collect status from each vehicle and MEC server. Meanwhile, the data center combines the information into the system state. In addition, the data center sends the state to the agent and receives the feedback of the optimal policy, then the data center selects appropriate server to offload task. In A3C algorithm, the task offloading experiences are stored in the replay. Algorithm 1 shows the asynchronous advantage actor-critic algorithm in task offloading. In order to obtain the optimal policy in task offloading, we should determine the reward, state and action in A3C algorithm.

5 Results and discussion

In the simulation settings, we consider a scenario as follows. The real vehicle dataset of Rome city is used in the experiment for verification. The network contains 100 sensing devices, 6-20 vehicles, 50 MEC servers and 1 cloud center. The sensing devices are randomly deployed, the vehicles are moving around along the roads, the

Algorithm 1 : A3C Algorithm in Task Offloading

Set global parameters θ,θ_v
Set parameters θ',θ'_v
Set shared counter $T = 0$
Set counter $t = 1$
repeat
  Reset global gradients $d\theta = 0$, $d\theta_v = 0$
  Synchronize parameters $\theta' = \theta$, $\theta'_v = \theta_v$
  $t_{start} = t$
  Observe from Env and obtain state s_t
  repeat
    Perform a_t according to $\pi(a_t|s_t;\theta')$
    Receive reward r_t and obtain next state s_{t+1}
    $t = t + 1$
    $T = T + 1$
  until s_t terminal or $t - t_{start} = t_{max}$
  if s_t is the terminal state **then**
    $R = 0$
  else
    $R = V(s_t|\theta'_v)$
  end if
  for $i \in \{t-1, \cdots, t_{max}\}$ **do**
    $R = r_i + \gamma R$
    Compute gradient $\theta' : d\theta = d\theta + \bigtriangledown_{\theta'} \log \pi(a_i|s_i;\theta')(R - V(s_i|\theta'))$
    Compute gradient $\theta'_v : d\theta_v = d\theta_v + (\partial(R - V(s_i|\theta'_v)))^2/\partial\theta'_v$
  end for
  Asynchronous update θ using $d\theta$
  Asynchronous update θ_v using $d\theta_v$
until $T > T_{max}$

Table 1 Experimental parameters

Parameter	Value
Number of sensing devices	100
Number of vehicles	6-20
Number of MEC servers	50
Number of cloud servers	1
The maximum bandwidth of MEC server	2.5MHZ
The capability of MEC server	6.5GHZ

MEC servers are randomly deployed within 100-1000 meters from the cloud center. The maximum bandwidth of MEC server is 2.5MHZ, the capability of MEC server is 6.5GHZ. About the training parameters, the actor's learning rate is 0.001, the critic's learning rate is 0.01. The CPU is Intel Core i5-8400 with 32G memory. The deep reinforcement learning environment is implemented on Python. The experimental parameters are showed in Table 1.

We compare our solution with two other solutions. The two comparison schemes are described as follows.

1) Actor Critic: the scheme that uses actor critic algorithm as the deep reinforcement learning algorithm to train the model and offload computation tasks.

2) Nearest Neighbor: the scheme that uses nearest neighbor algorithm to offload computation tasks.

5.1 Analysis of the reward

Fig. 2 illustrates the impact of learning rounds on the reward performance of COTV scheme and actor critic scheme. From the figure, we can see that COTV outperforms the actor critic scheme in the reward performance. COTV has a higher reward than actor critic scheme after 30 learning rounds. We can observe that the reward increases as the learning rounds increase, and the reward performance

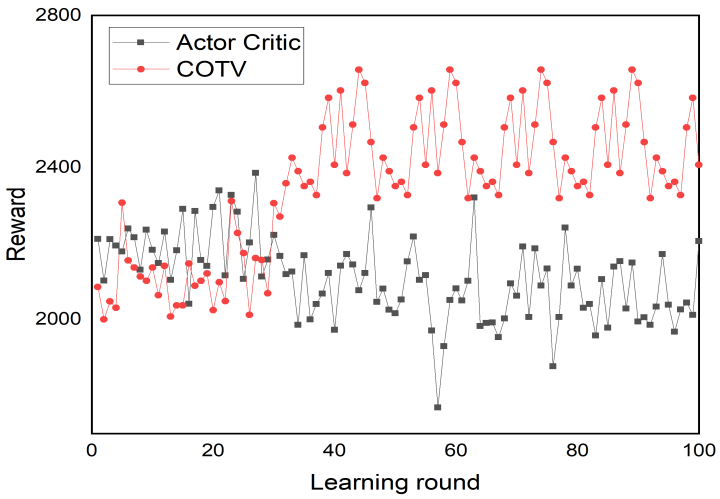

Figure 2 Reward versus learning round

converges gradually. In the learning rate performance, we do not compare the performance of nearest neighbor scheme. The reason is that the nearest neighbor scheme selects the MEC server or cloud server to offload computation tasks based on the distance, the vehicles select the nearest server to offload tasks and do not have the training process. The COTV scheme's training performance is better than actor critic algorithm, because the A3C algorithm used in COTV scheme can utilize asynchronous mechanism to accelerate the training process.

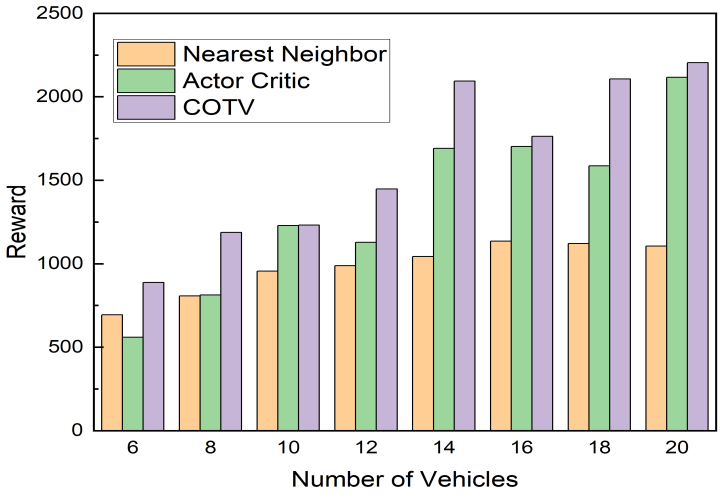

Figure 3 Reward versus number of vehicles

Fig. 3 compares the reward performance of three algorithms in different number of vehicles. The vehicles change from 5 to 20, the day is fixed as the 30th day. As we can see from the figure, the reward increases with the increase of the number of vehicles in general. Because the reward is the total reward of the whole system,

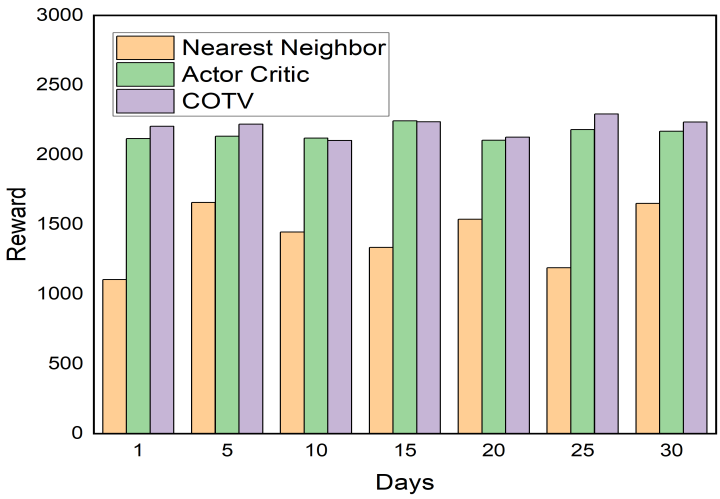

Figure 4 Reward versus days

thus the number of vehicles is an important factor to determine the total reward. Thus, we choose to fix the number of vehicles as 20 in the follow-up experiments.

Fig. 4 shows the impact of days on the performance of reward. The three algorithms are compared when the number of vehicles is 20. The real Rome dataset contains vehicles' trajectory of 30 days, we select 7 representative days to illustrate the impact of days on the reward performance. As we can see from the figure, the reward performance of COTV scheme keeps very stable in different days. The nearest neighbor scheme has a greater change compared with our COTV scheme. The experiment results show that our COTV scheme has a stable performance.

5.2 Analysis of the number of packets

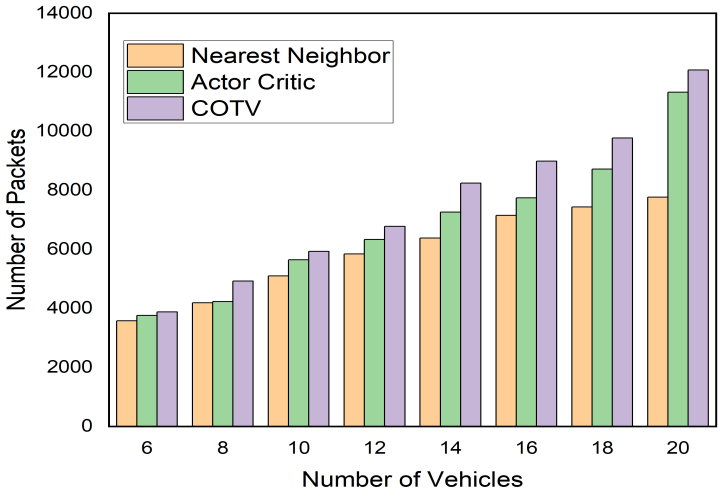

Figure 5 Packets versus number of vehicles

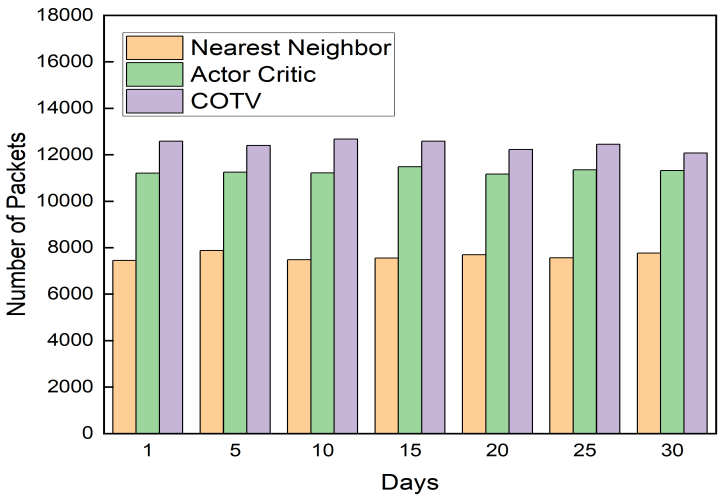

Figure 6 Packets versus days

Fig. 5 compares the packets of three algorithms in different number of vehicles. Compared with actor critic scheme and nearest neighbor scheme, COTV scheme sends more packets. The number of packets increase with the increase of the number of vehicles, because the whole system can send and process more packets with more vehicles.

Fig. 6 shows the performance comparison of packets in different days. The figure shows that COTV's packets are similar in different days, which indicates that the training process can make the packet performance better than the untrained nearest neighbor scheme.

5.3 Analysis of the delay

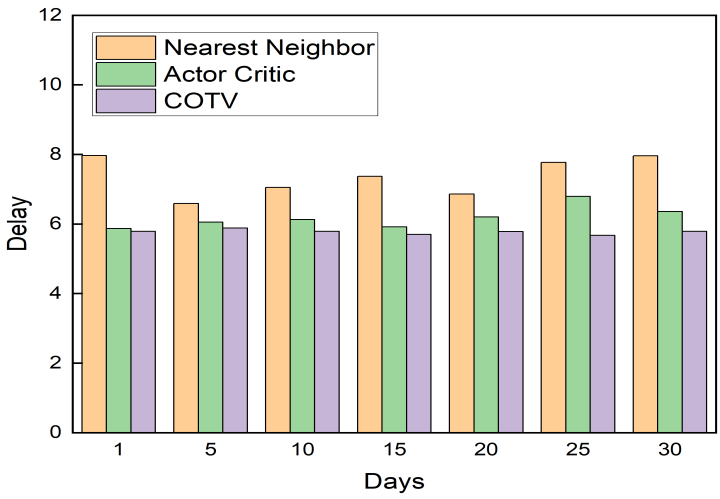

Figure 7 Delay versus days

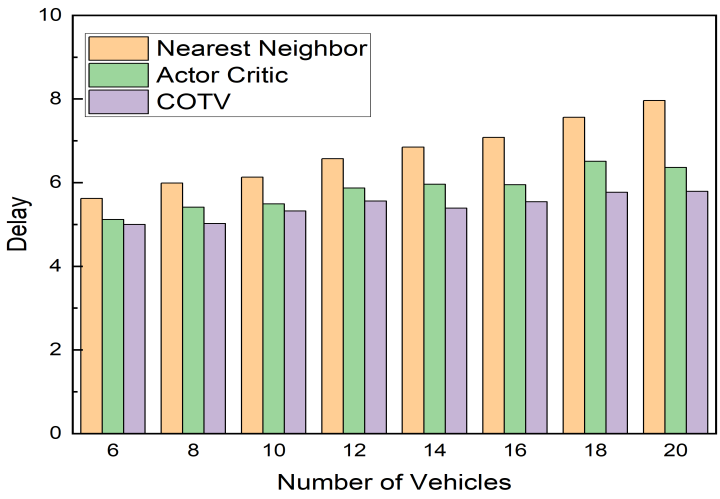

Figure 8 Delay versus number of vehicles

Fig. 7 shows the performance comparison of delay on different days. Delay is a very important factor for the computation offloading of the tasks. If the delay time is larger than the maximum tolerable time, the task may not be able to offload to the other platforms. The vehicles are continuously moving on the road, it is very important to select an efficient scheme to offload the computation tasks. Our scheme performs better than the nearest neighbor scheme and actor critic scheme. The experiment results show that our COTV algorithm can choose the appropriate server to offload tasks. From the figure, we can see that our COTV scheme performs best at the 5th day in 30 days.

Fig. 8 compares the delay time of three algorithms in different number of vehicles. The delay time of our COTV scheme is the smallest in three compared schemes. As the number of vehicles increase, the delay time increases. Thus, the number of vehicles should be selected appropriately if the task is delay-sensitive. In general, the delay time is totally acceptable.

The experimental results show that our scheme is better than the existing scheme. We can see that our performances on reward, packets and delay are better compared with the comparison schemes. The reason of our scheme is better than the compared schemes is illustrated as follows. We model the computation offloading process as a reinforcement learning process, and we use A3C algorithm to perform computation offloading. The A3C algorithm uses multiple threads for asynchronous parallel processing, which can speed up the convergence, improve learning efficiency, and enhance performances.

Since the IoT-edge-cloud network is heterogeneous and there are many devices in IoT, the computation offloading problem of IoT devices is very difficult. When selecting a platform for computation offloading, the computation offloading decisions are very important. In this article, we propose to use vehicles to support computation offloading of IoT devices, and make computation offloading decisions based on A3C algorithm. The proposed computation offloading framework can be widely applied to the computation offloading of IoT devices in smart city. In the future,

we are interested in exploring more complex offloading models, such as partial offloading. In addition, we can combine technologies such as blockchain and privacy protection to further enhance the security of computation offloading.

6 Conclusion

In smart city, there are many sensing devices deployed in the basic infrastructures. The sensing devices need to transmit the task results to the cloud center for decision making. However, some sensing devices cannot connect to the Internet, and it consumes a lot of money to deploy wireless network for these devices. In this paper, we propose a computation offloading scheme through vehicles in IoT-edge-cloud network. Through introducing mobile vehicles and offloading the tasks of the sensing devices, the energy consumption of sensing devices can be decreased. The tasks can be processed in the vehicles, MEC servers or cloud center. In our proposed algorithm, the task offloading cost considers the computation cost and the task delay. Through comparing the offloading costs of the vehicles, MEC servers and cloud center, we select the platform using deep reinforcement learning algorithm. In our experiment, we compare our proposed scheme with the other schemes, our algorithm can achieve maximum reward.

Abbreviations

MEC: Mobile edge computing; IoT: Internet of things; AR: Augmented reality; COTV: Computation offloading scheme through mobile vehicles; IDL: IoT device layer; MSL: Mobile edge computing server layer; CCL: Cloud center layer

Availability of data and materials

Not applicable.

Competing interests

The authors declare that they have no competing interests.

Funding

National Natural Science Foundation of China under Grant (61472450, 61772554), the Key Technology Research and Development Program of Hunan Province under Grant 2018GK2052.

Author's contributions

Jun Long is the main author of the current paper. Yueyi Luo contributed to the conception and design of the study. Xiaoyu Zhu, Entao Luo and Mingfeng Huang commented the work. All authors read and approved the final manuscript.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China under Grant 61772554, in part by the Key Technology Research and Development Program of Hunan Province under Grant 2018GK2052.

Author details

[1]School of Computer Science and Engineering, Central South University, 410083 Changsha, China. [2]Network Resources Management and Trust Evaluation Key Laboratory of Hunan Province, Central South University, 410083 Changsha, China. [3]School of Mathematics and Statics, Central South University, 410083 Changsha, China. [4]School of Electronics and Information Engineering, Hunan University of Science and Engineering, 425199 Yongzhou, China.

References

1. Xie, K., Ning, X., Wang, X., Xie, D., Cao, J., Xie, G., Wen, J.: Recover corrupted data in sensor networks: A matrix completion solution. IEEE Transactions on Mobile Computing **16**(5), 1434–1448 (2017)
2. Liu, X., Lin, P., Liu, T., Wang, T., Liu, A., Xu, W.: Objective-variable tour planning for mobile data collection in partitioned sensor networks. IEEE Transactions on Mobile Computing (2020). https://doi.org/10.1109/TMC.2020.3003004
3. Liu, X., Qiu, T., Dai, B., Yang, L., Liu, A., Wang, J.: Swarm intelligence-based rendezvous selection via edge computing for mobile sensor networks. IEEE Internet of Things Journal (2020). https://doi.org/10.1109/JIOT.2020.2966870
4. Huang, S., Liu, A., Zhang, S., Wang, T., Xiong, N.: Bd-vte: a novel baseline data based verifiable trust evaluation scheme for smart network systems. IEEE Transactions on Network Science and Engineering (2020). https://doi.org/10.1109/TNSE.2020.3014455
5. Wang, T., Cao, Z., Wang, S., Wang, J., Qi, L., Liu, A., Xie, M., Li, X.: Privacy-enhanced data collection based on deep learning for internet of vehicles. IEEE Transactions on Industrial Informatics **16**(10), 6663–6672 (2020)
6. Teng, H., Ota, K., Liu, A., Wang, T., Zhang, S.: Vehicles joint uavs to acquire and analyze data for topology discovery in large-scale iot systems. Peer-to-Peer Networking and Applications **13**(5), 1720–1743 (2020)
7. Ren, Y., Zeng, Z., Wang, T., Zhang, S., Zhi, G.: A trust-based minimum cost and quality aware data collection scheme in p2p network. Peer-to-Peer Networking and Applications (2020). https://doi.org/10.1007/s12083-020-00898-2
8. Zhao, Y., Wang, T., Zhang, S., Wang, Y.: Towards minimum code dissemination delay through uav joint vehicles for smart city. IET Communications (2020). https://doi.org/10.1049/iet-com.2019.1205
9. Peng, M., Liu, W., Wang, T., Zeng, Z.: Relay selection joint consecutive packet routing scheme to improve performance for wake-up radio-enabled wsns. Wireless Communications & Mobile Computing (2020). https://doi.org/10.1155/2020/7230565
10. Liu, X., Liu, A., Qiu, T., Dai, B., Wang, T., Yang, L.: Restoring connectivity of damaged sensor networks for long-term survival in hostile environments. IEEE Internet of Things Journal **7**(2), 1205–1215 (2020)
11. Li, Z., Liu, Y., Liu, A., Wang, S., Liu, H.: Minimizing convergecast time and energy consumption in green internet of things. IEEE Transactions on Emerging Topics in Computing **8**(3), 797–813 (2020)
12. Zhu, X., Luo, Y., Liu, A., Tang, W., Bhuiyan, M.Z.A.: A deep learning-based mobile crowdsensing scheme by predicting vehicle mobility. IEEE Transactions on Intelligent Transportation Systems (2020). https://doi.org/10.1109/TITS.2020.3023446
13. Wang, X., Liu, Z., Gao, Y., Zheng, X., Dang, Z., Shen, X.: A near-optimal protocol for the grouping problem in rfid systems. IEEE Transactions on Mobile Computing (2019). https://doi.org/10.1109/TMC.2019.2962125
14. Liu, X., Song, H., Liu, A.: Intelligent uavs trajectory optimization from space-time for data collection in social networks. IEEE Transactions on Network Science and Engineering (2020). https://doi.org/10.1109/TNSE.2020.3017556
15. Li, T., Zhao, M., Wong, K.K.L.: Machine learning based code dissemination by selection of reliability mobile vehicles in 5g networks. Computer Communications **152**, 109–118 (2020)
16. Ge, J., Liu, B., Wang, T., Yang, Q., Liu, A., Li, A.: Q-learning based flexible task scheduling in a global view for the internet of things. Transactions on Emerging Telecommunications Technologies, 4111. https://doi.org/10.1002/ett.4111
17. Ren, Y., Wang, T., Zhang, S., Zhang, J.: An intelligent big data collection technology based on micro mobile data centers for crowdsensing vehicular sensor network. Personal and Ubiquitous Computing, 1–17 (2020)
18. Tan, J., Liu, W., Wang, T., Zhao, M., Liu, A., Zhang, S.: A high-accurate content popularity prediction computational modeling for mobile edge computing using matrix completion technology. Transactions on Emerging Telecommunications Technologies. https://doi.org/10.1002/ett.3871
19. Wang, T., Qiu, L., Sangaiah, A.K., Liu, A., Bhuiyan, M.Z.A., Ma, Y.: Edge-computing-based trustworthy data collection model in the internet of things. IEEE Internet of Things Journal **7**(5), 4218–4227 (2020)
20. Wang, T., Liang, Y., Yang, Y., Xu, G., Peng, H., Liu, A., Jia, W.: An intelligent edge-computing-based method to counter coupling problems in cyber-physical systems. IEEE Network **34**(3), 16–22 (2020)
21. Liu, Q., Tian, Y., Wu, J., Peng, T., Wang, G.: Enabling verifiable and dynamic ranked search over outsourced data. IEEE Transactions on Services Computing (2019). https://doi.org/10.1109/TSC.2019.2922177
22. Zhu, X., Wu, J., Chang, W., Wang, G., Liu, Q.: Efficient authentication of multi-dimensional top- k queries. IEEE Access **7**, 4748–4762 (2019)
23. Liu, Q., Wang, G., Liu, X., Peng, T., Wu, J.: Achieving reliable and secure services in cloud computing environments. Computers & Electrical Engineering **59**, 153–164 (2017)
24. He, H., Shan, H., Huang, A., Ye, Q., Zhuang, W.: Edge-aided computing and transmission scheduling for lte-u-enabled iot. IEEE Transactions on Wireless Communications (2020). https://doi.org/10.1109/TWC.2020.3017207
25. Xu, Y., Ren, J., Zhang, Y., Zhang, C., Shen, B., Zhang, Y.: Blockchain empowered arbitrable data auditing scheme for network storage as a service. IEEE Transactions on Services Computing **13**(2), 289–300 (2020)
26. Liu, Q., Guo, Y., Wu, J., Wang, G.: Effective query grouping strategy in clouds. Journal of Computer Science and Technology **32**(6), 1231–1249 (2017)
27. Wang, P., Zheng, Z., Di, B., Song, L.: Hetmec: Latency-optimal task assignment and resource allocation for heterogeneous multi-layer mobile edge computing. IEEE Transactions on Wireless Communications **18**(10), 4942–4956 (2019)
28. Huang, M., Liu, W., Wang, T., Liu, A., Zhang, S.: A cloud-mec collaborative task offloading scheme with service orchestration. IEEE Internet of Things Journal **7**(7), 5792–5805 (2020)
29. Chen, M., Hao, Y.: Task offloading for mobile edge computing in software defined ultra-dense network. IEEE Journal on Selected Areas in Communications **36**(3), 587–597 (2018)
30. Li, T., Liu, W., Wang, T., Ming, Z., Li, X., Ma, M.: Trust data collections via vehicles joint with unmanned

aerial vehicles in the smart internet of things. Transactions on Emerging Telecommunications Technologies (2020). https://doi.org/10.1002/ett.3956
31. Huang, M., Zhang, K., Zeng, Z., Wang, T., Liu, Y.: An auv-assisted data gathering scheme based on clustering and matrix completion for smart ocean. IEEE Internet of Things Journal (2020). doi:10.1109/JIOT.2020.2988035
32. Liu, Y., Li, Y., Niu, Y., Jin, D.: Joint optimization of path planning and resource allocation in mobile edge computing. IEEE Transactions on Mobile Computing (2019)
33. Ning, Z., Wang, X., Rodrigues, J.J., Xia, F.: Joint computation offloading, power allocation, and channel assignment for 5g-enabled traffic management systems. IEEE Transactions on Industrial Informatics **15**(5), 3058–3067 (2019)
34. Hoang, V.H., Ho, T.M., Le, L.B.: Mobility-aware computation offloading in mec-based vehicular wireless networks. IEEE Communications Letters **24**(2), 466–469 (2019)
35. Peng, H., Ye, Q., Shen, X.: Spectrum management for multi-access edge computing in autonomous vehicular networks. IEEE Transactions on Intelligent Transportation Systems **21**(7), 3001–3012 (2020)
36. Zhuang, W., Ye, Q., Lyu, F., Cheng, N., Ren, J.: Sdn/nfv-empowered future iov with enhanced communication, computing, and caching. Proceedings of the IEEE **108**(2), 274–291 (2019)
37. Misra, S., Bera, S.: Soft-van: Mobility-aware task offloading in software-defined vehicular network. IEEE Transactions on Vehicular Technology **69**(2), 2071–2078 (2019)
38. Luo, Y., Zhu, X., Long, J.: Data collection through mobile vehicles in edge network of smart city. IEEE Access **7**, 168467–168483 (2019)
39. Jiang, B., Huang, G., Wang, T., Gui, J., Zhu, X.: Trust based energy efficient data collection with unmanned aerial vehicle in edge network. Transactions on Emerging Telecommunications Technologies (2020). https://doi.org/10.1002/ett.3942
40. Liu, Y., Zeng, Z., Liu, X., Zhu, X., Bhuiyan, M.Z.A.: A novel load balancing and low response delay framework for edge-cloud network based on sdn. IEEE Internet of Things Journal **7**(7), 5922–5933 (2020)
41. He, Y., Zhao, N., Yin, H.: Integrated networking, caching, and computing for connected vehicles: A deep reinforcement learning approach. IEEE Transactions on Vehicular Technology **67**(1), 44–55 (2017)
42. Lu, H., Gu, C., Luo, F., Ding, W., Liu, X.: Optimization of lightweight task offloading strategy for mobile edge computing based on deep reinforcement learning. Future Generation Computer Systems **102**, 847–861 (2020)
43. Qi, Q., Wang, J., Ma, Z., Sun, H., Cao, Y., Zhang, L., Liao, J.: Knowledge-driven service offloading decision for vehicular edge computing: A deep reinforcement learning approach. IEEE Transactions on Vehicular Technology **68**(5), 4192–4203 (2019)
44. Feng, J., Richard Yu, F., Pei, Q., Chu, X., Du, J., Zhu, L.: Cooperative computation offloading and resource allocation for blockchain-enabled mobile-edge computing: A deep reinforcement learning approach. IEEE Internet of Things Journal **7**(7), 6214–6228 (2020)
45. Xu, Y., Zhang, C., Wang, G., Qin, Z., Zeng, Q.: A blockchain-enabled deduplicatable data auditing mechanism for network storage services. IEEE Transactions on Emerging Topics in Computing (2020). https://doi.org/10.1109/TETC.2020.3005610
46. Xu, Y., Zhang, C., Zeng, Q., Wang, G., Ren, J., Zhang, Y.: Blockchain-enabled accountability mechanism against information leakage in vertical industry services. IEEE Transactions on Network Science and Engineering (2020). https://doi.org/10.1109/TNSE.2020.2976697
47. Zhang, S., Wang, G., Bhuiyan, M.Z.A., Liu, Q.: A dual privacy preserving scheme in continuous location-based services. IEEE Internet of Things Journal **5**(5), 4191–4200 (2018)
48. Zhang, S., Mao, X., Choo, K.-K.R., Peng, T., Wang, G.: A trajectory privacy-preserving scheme based on a dual-k mechanism for continuous location-based services. Information Sciences **527**, 406–419 (2020)

Figures

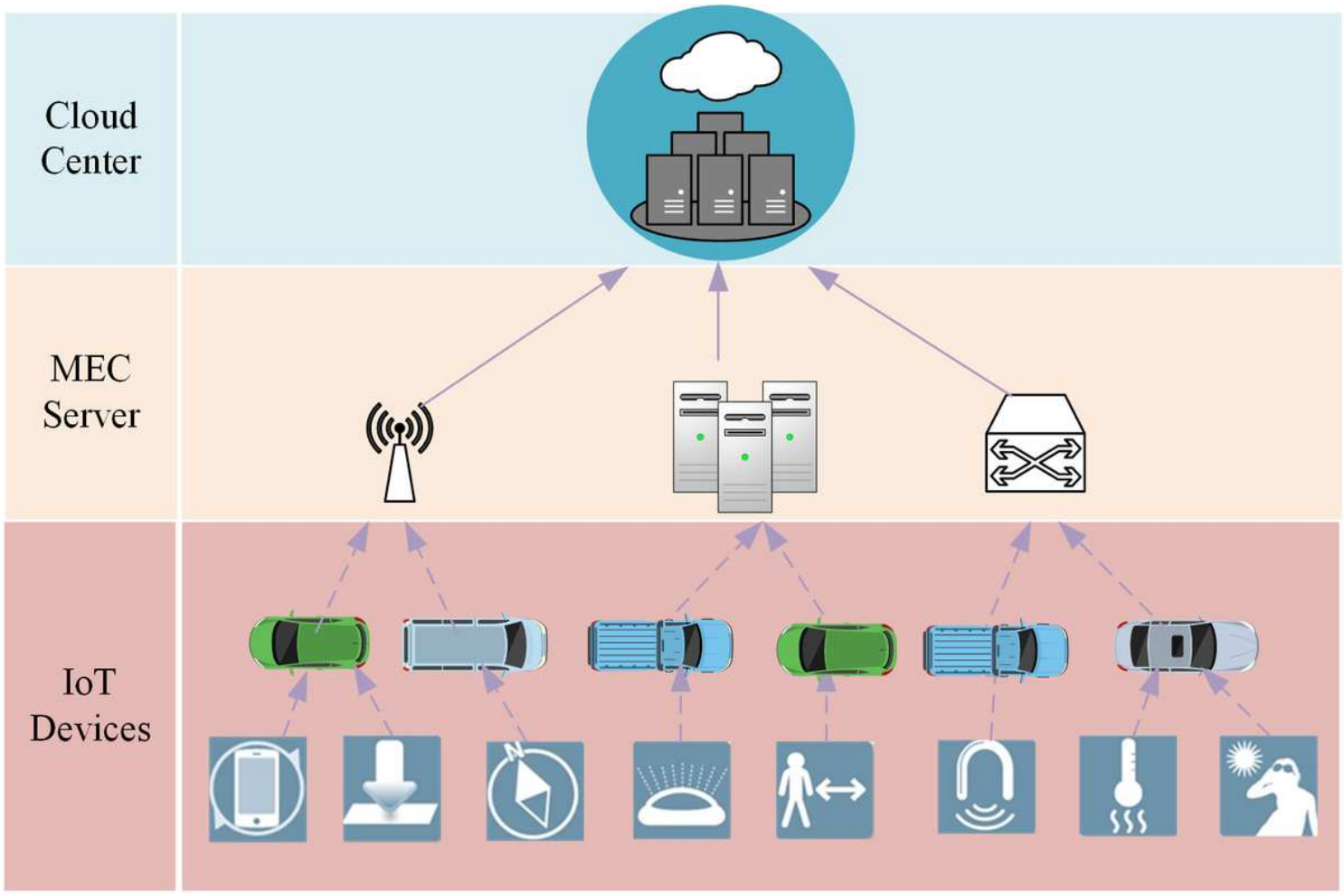

Figure 1

The structure of COTV

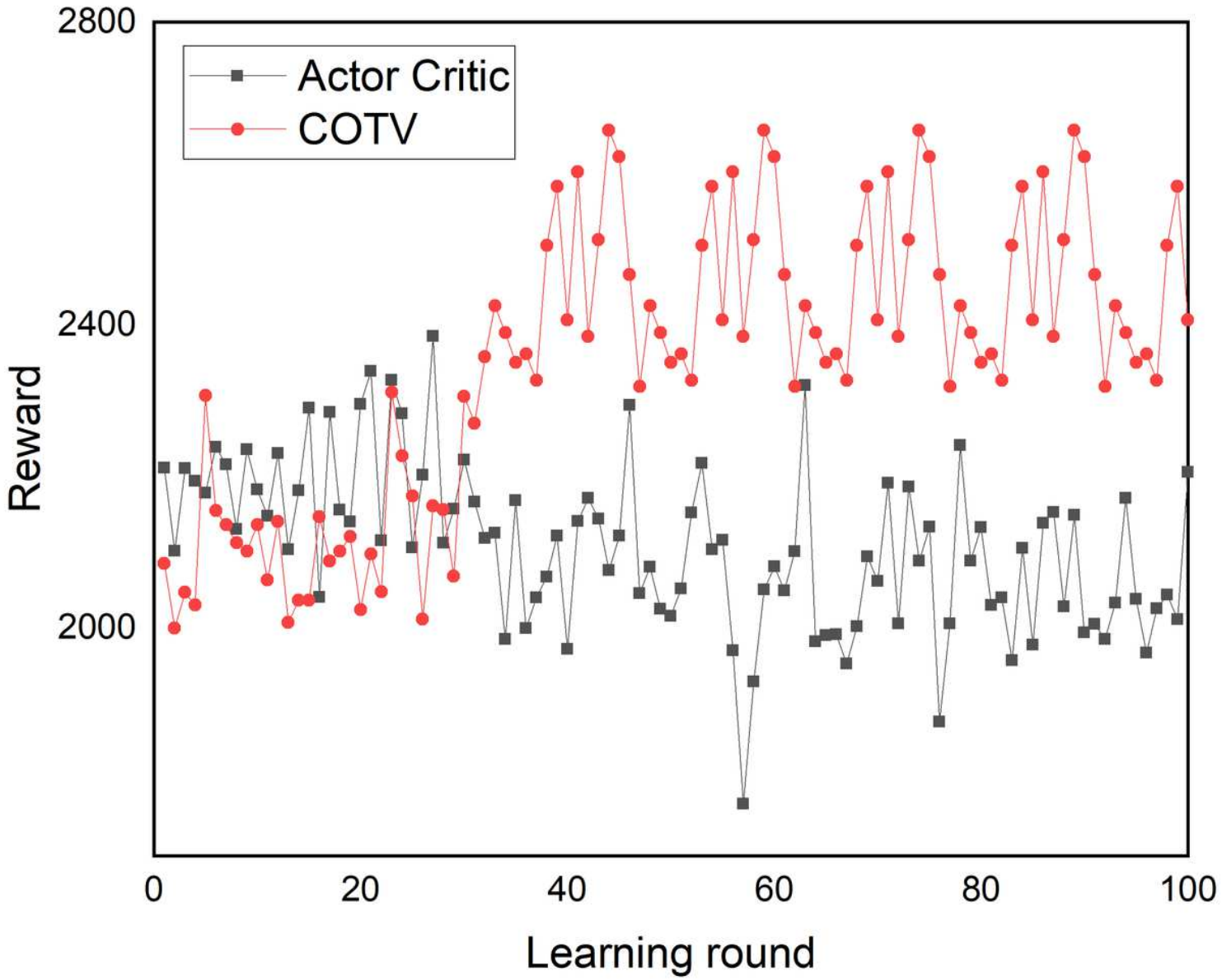

Figure 2

Reward versus learning round

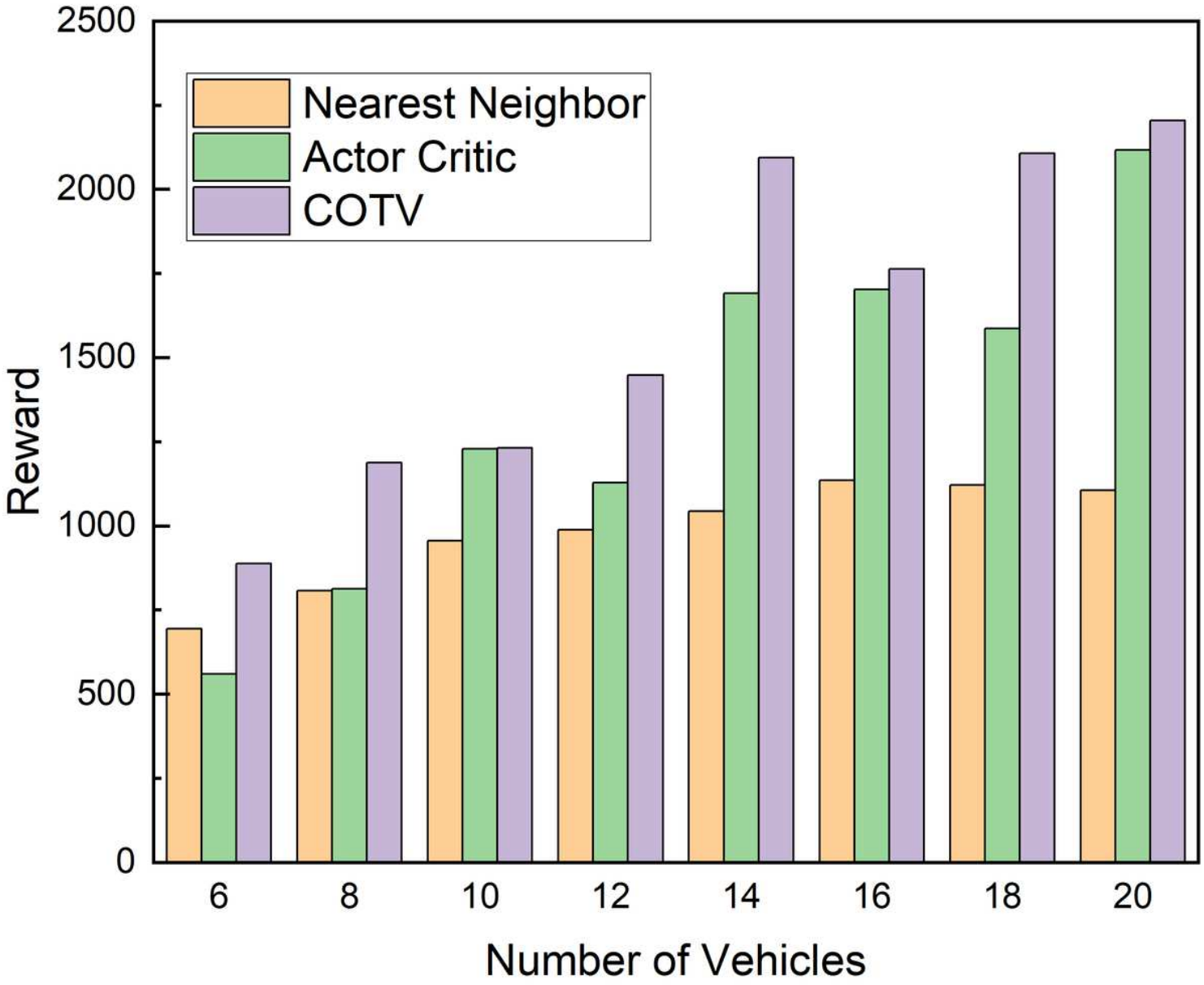

Figure 3

Reward versus number of vehicles

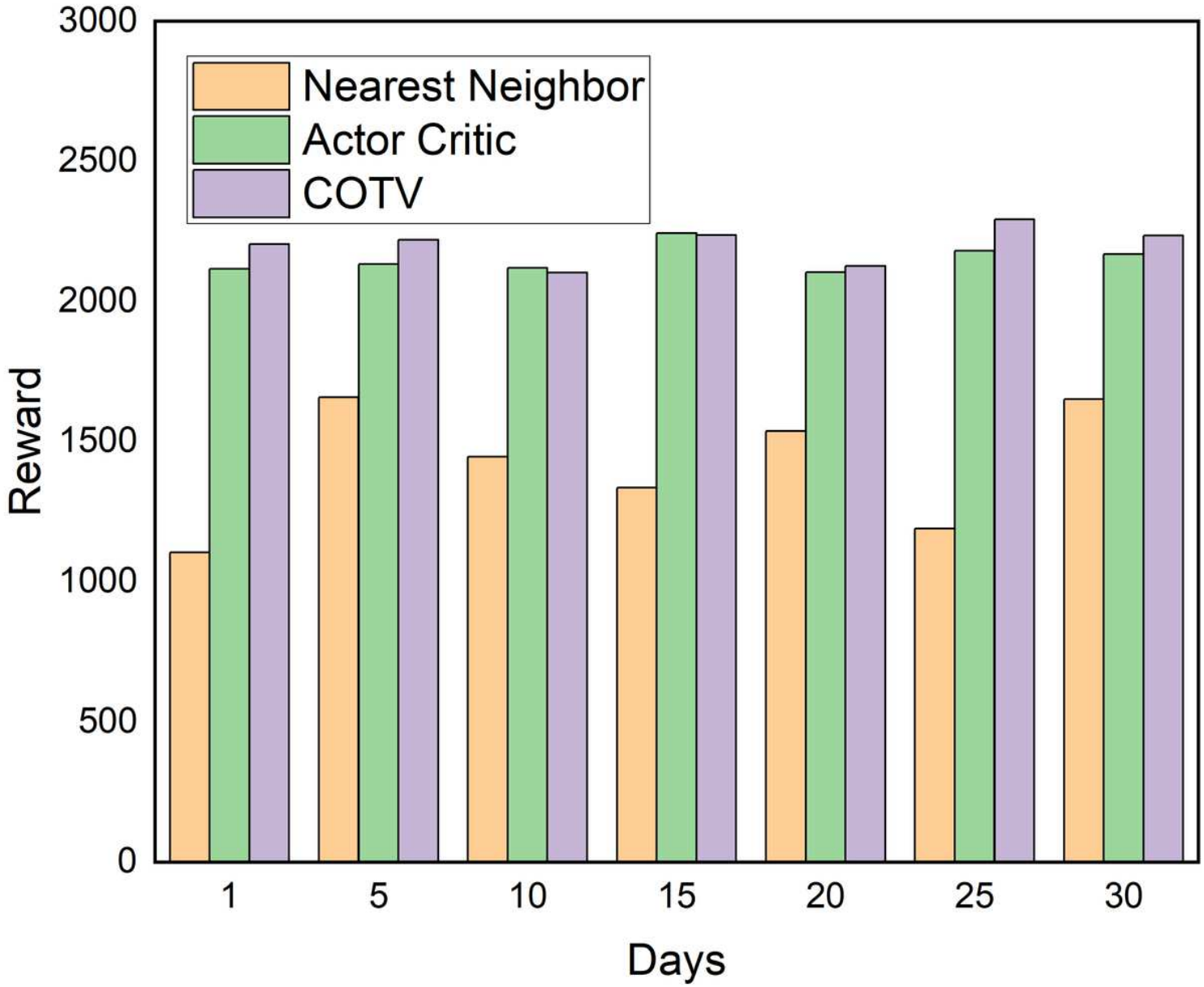

Figure 4

Reward versus days

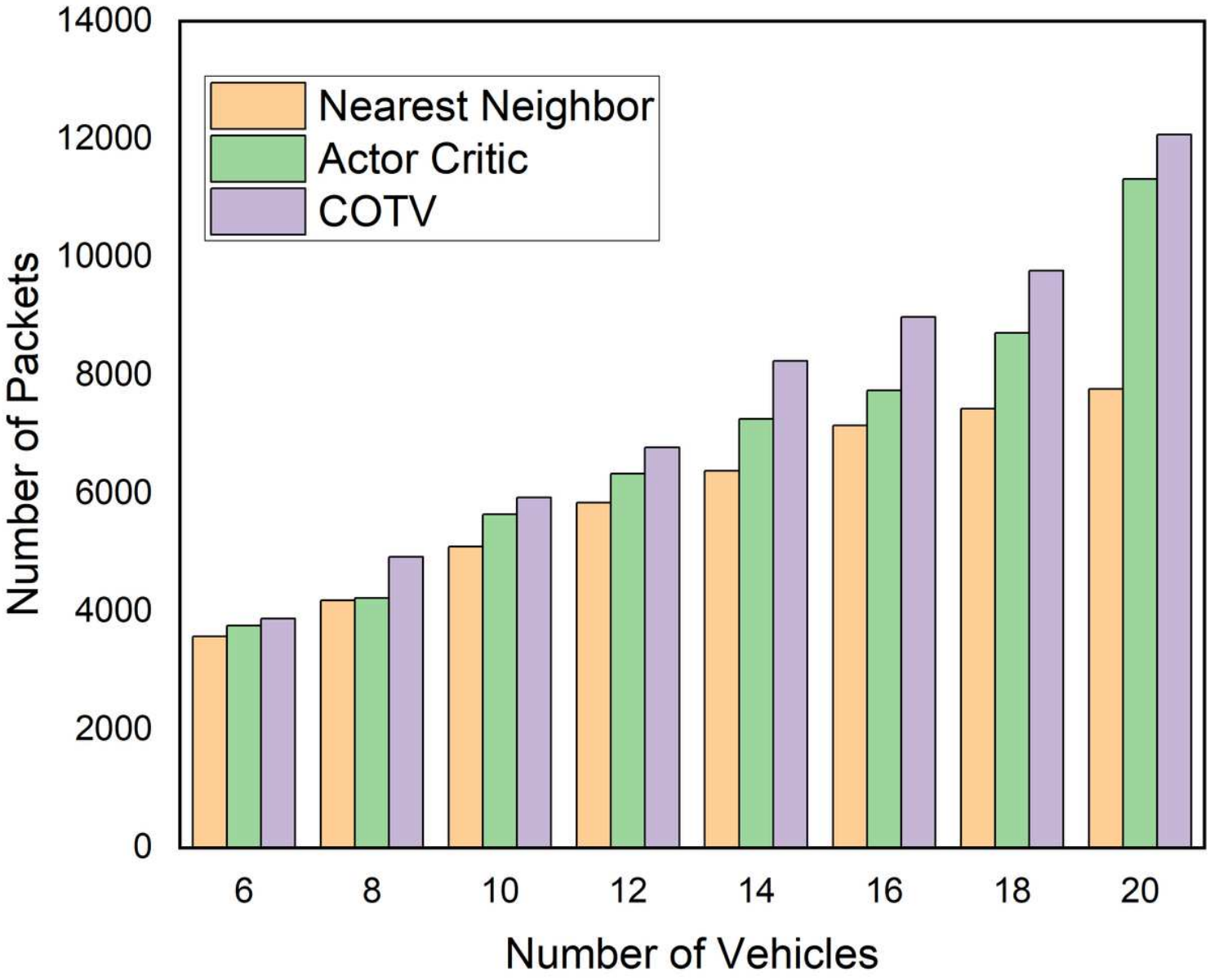

Figure 5

Packets versus number of vehicles

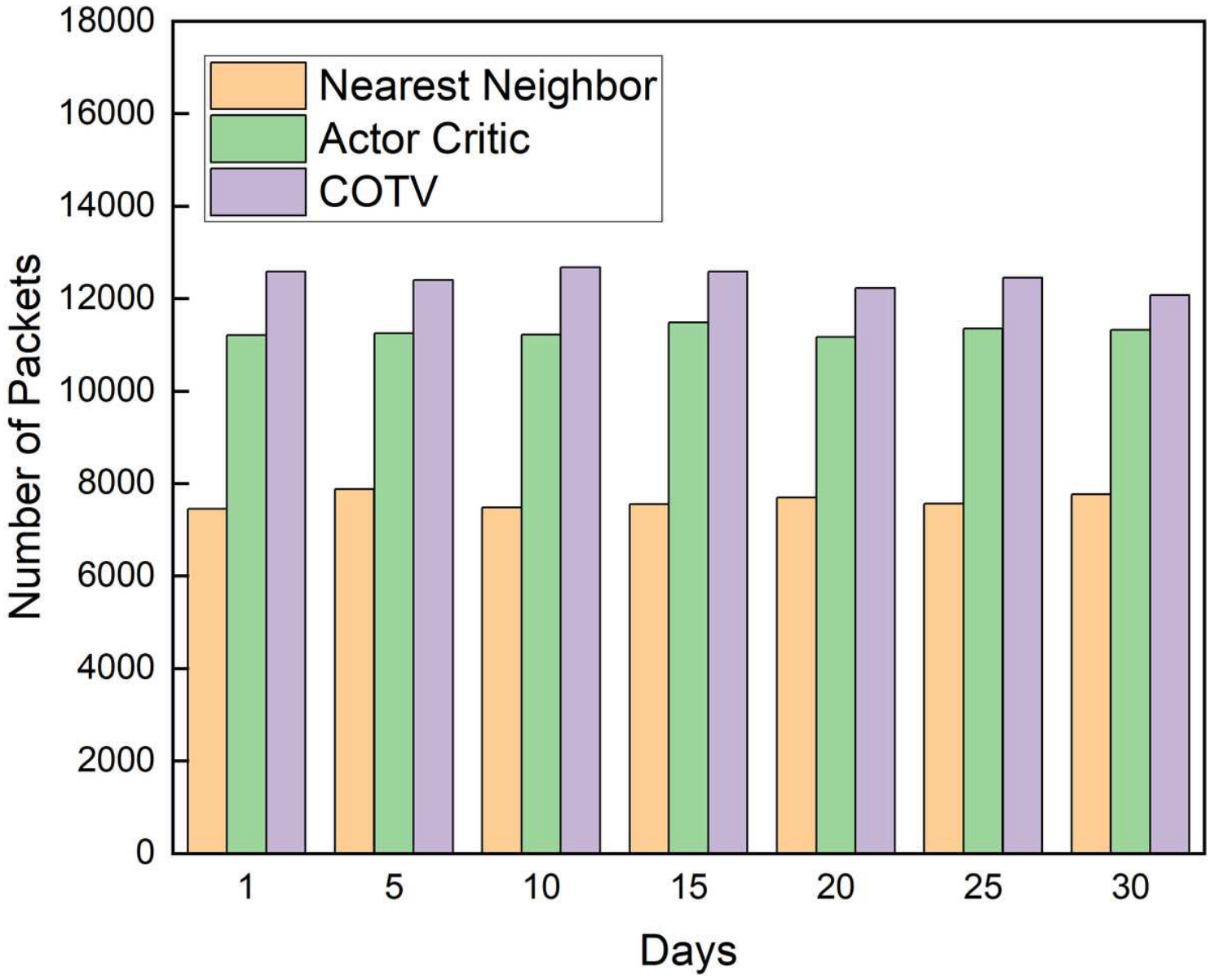

Figure 6

Packets versus days

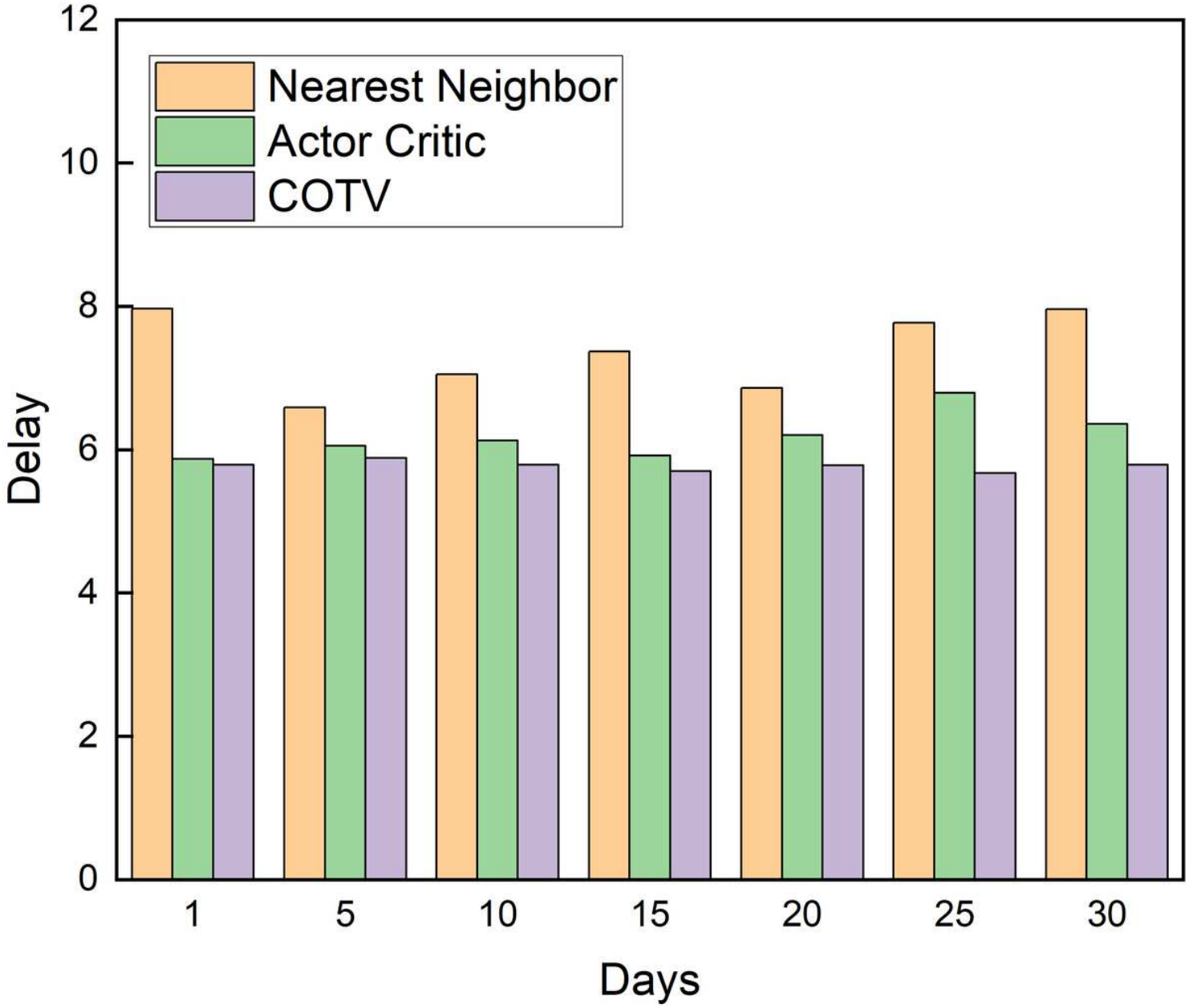

Figure 7

Delay versus days

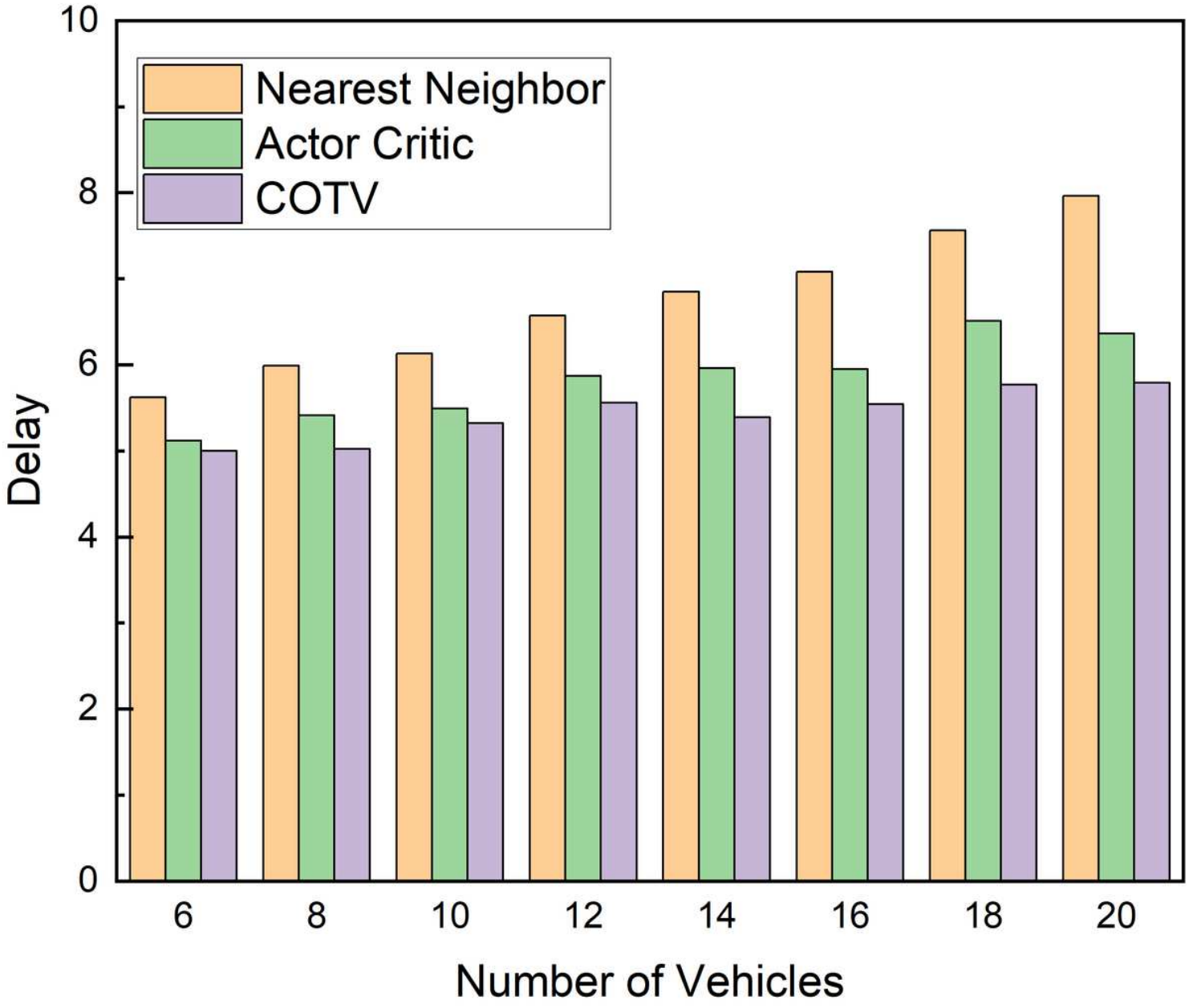

Figure 8

Delay versus number of vehicles