Cooperative mobile edge computing-cloud computing in Internet of vehicle: Architecture and energy-efficient workload allocation

Xiaohui Gu | Guoan Zhang | Yujie Cao

School of Information Science and Technology, Nantong University, Nantong, China

Correspondence
Guoan Zhang, School of Information Science and Technology, Nantong University, Nantong, Jiangsu, 226000, China. Email: gzhang@ntu.edu.cn

Funding information
This work is supported by the National Natural Science Foundation of China under Grant no. 61971245 and 61801249

Abstract
With the increasing number of vehicles, the generating vehicular data exceeds the capacity of mobile edge computing (MEC). Therefore, studying the interaction and collaboration of edge computing and cloud computing is of significance to provide vehicular users with low-latency high-rate services. This paper first proposes a MEC-cloud computing collaboration architecture for Internet of vehicles, then designs the interconnection/interaction framework between MEC and cloud computing. We consider reducing computation delay and power consumption, and formulate an energy-efficient workload allocation problem with load balancing and dynamic voltage frequency scaling technology, to obtain the optimal workload allocations of MEC and cloud computing. We then present the overall distribution optimization algorithm to solve this problem. The simulation and numerical results show that by saving communication bandwidth and reducing transmission delay, MEC significantly enhances the performance of cloud computing. Besides, the proposed workload balance scheme is better than the benchmark schemes in terms of power consumption and latency.

KEYWORDS
Cloud computing, Energy-latency tradeoff, Internet of vehicles, Mobile edge computing, Workload allocation

1 | INTRODUCTION

The intelligent transportation system (ITS) is an important part of smart cities, and the Internet of vehicles (IoV) is able to reduce traffic accidents, alleviate traffic congestion and provide various real-time convenient information, such as navigation, multimedia entertainment, and remote fault diagnosis, thus IoV plays an important role in ITS. However, with the rapid development of IoV applications, mobile devices, and the Internet of things (IoT), mobile users have become increasingly stringent in terms of quality of service (QoS). Providing users with low-latency and high-reliability vehicular services has become a research hotspot in the academia and industry.

To alleviate the computing burden of vehicles with limited resources, the road side unit (RSU) can collect and upload computing tasks to a central cloud server for cloud computing, and sends the computation results back to local users. For example, in some navigation applications, the cloud server grasps the congestion states of urban roads based on the
GPS data dynamically uploaded by vehicles, then it plans the optimal driving route for vehicle users. However, IoV is a complex heterogeneous network consisting of vehicular ad hoc networks, IoT, roadside networks, and cellular mobile networks. Exploiting the cloud computing framework for big data management in IoV still exits some disadvantages like high latency, low efficiency and poor scalability. The root cause of these problems is the centralized computing and storage model of cloud services. Therefore, researchers have proposed some improvements (such as mobile cloud\textsuperscript{2} and cloudlet\textsuperscript{3}) to promote the localization of cloud services while maintaining the dominant position of the cloud server. In addition, with the emergence of similar concepts such as mobile cloud computing, micro-cloud, and fog computing, mobile edge computing (MEC), as an extension of cloud computing in the edge network, is able to overcome the high latency obstacles when traditional cloud computing provides services.

By extending cloud computing to the edge network, MEC can provide big data computing services more economically and effectively, and it has become a hot topic today.\textsuperscript{4,5} For example, Ahmed et al\textsuperscript{6} emphasized that MEC provides high-rate and low-latency services, which is a beneficial computing platform for many IoT scenarios and big data services. Tran,\textsuperscript{7} Taleb,\textsuperscript{8} and Mao\textsuperscript{9} reviewed literatures on MEC in recent years, they discussed a number of MEC application scenarios, such as vehicular networks, the smart grid, medical electronics, telecommunication cellular networks, etc. They also discussed the current security and privacy issues faced by MEC. Applying MEC to IoV to provide vehicular services with low-latency and highly-reliability has received widespread attention. For example, Elhajj et al\textsuperscript{10} based on the concept of MEC designed a solution to reduce the user-perceived latency in IoT scenarios, and suggested to treat the computing resources of vehicles as virtualized miniature data centers for data sharing and local services providing. Yu et al\textsuperscript{11} illustrated the concept of the roadside cloud, which is composed of local servers and RSUs, to process traffic data and provide edge computing services. In addition, MEC is also used in IoV in other aspects. Zhang et al\textsuperscript{12} proposed the vehicular edge computing (VEC) architecture to simulate the computation offloading process of mobile vehicles. Deng et al\textsuperscript{13} applied software defined network (SDN) to design routing protocols for vehicular networks, where the SDN controller issues routing control information based on the global network state information, and the MEC server controls the local network. Finally, Zhou et al\textsuperscript{14} suggested that pushing the AI frontiers to MEC, is a promising solution to fully unleash the potential of the edge big data, and they reviewed the emerging efforts on AI model toward training/inference at the network edge.

On the other hand, technological advances in Internet data center (IDC) design have brought many new opportunities for power management of computing servers. At the server level, chip multiprocesing, dynamic voltage and frequency scaling (DVFS), sleep scheduling mechanisms, and virtual machine management have emerged and been gradually mature. At the data center level, efficient load balancing and task scheduling schemes can further improve the efficiency of power supplies.\textsuperscript{15,16} These technologies provide technical support for complex network and physical interactions between the MEC layer and the cloud layer. Moreover, SDN as an emerging software-based network architecture and technology can integrate logical architecture and physical architecture by coding to maximize the resources utilization. For example, by aggregating similar services and sleep scheduling (operational/hibernating) computing servers while meeting latency constraints, the energy consumption can be saved by balancing the workload of computing servers and communication networks, when the system workload increases.

This paper emphasizes the collaboration between cloud servers and MEC servers, and proposes a MEC-cloud computing collaboration architecture to support various IoV applications and users. Different from the central control of cloud computing, we apply a centralized SDN controller to monitor the entire network state, and support the interaction and collaboration between cloud servers and MEC servers. Then based on the proposed architecture, we design a hierarchical framework of interconnection/interaction between MEC and cloud computing to study the system power consumption and delay. The workload allocation problem with latency constraints is formulated and the objective function aims to minimize the system power consumption. We decompose the primary problem (PP) into subproblems (SP) corresponding to three subsystems (edge computing subsystem, cloud computing subsystem, WAN communication subsystem), use the existing optimization techniques to solve the SPs respectively, design an iterative algorithm to obtain the optimal solution of PP, and an energy-efficient workload allocation scheme is obtained. This scheme can achieve workload balancing between MEC and cloud servers, and dynamically control the power and working state (operational/hibernating) of the computing server based on DVFS technology, while meeting the certain latency requirements. Simulation and numerical results prove that in the MEC-cloud computing collaboration system, MEC is able to reduce latency, save communication bandwidth, and further enhance cloud computing performance. Moreover, the workload allocation scheme proposed in this paper can effectively reduce system power consumption and latency compared to other approaches.

The remainder of this paper is organized as follows. Section 2 presents the cooperative MEC-cloud computing architecture. In Section 3, we discuss the workload allocation model, describe the optimization problem and develop a distributed
optimization algorithm. Section 4 provides simulation results to verify the advantages of the proposed method. Finally, conclusion is given in Section 5.

2 | PROPOSED ARCHITECTURE

The development of IoTs, the increase in the users’ demands and the advance of cloud computing technology have promoted the cooperation between MEC and cloud computing. Inspired by the hierarchical architecture proposed by some classic literatures, this paper proposes the MEC-cloud computing collaboration architecture as shown in Figure 1. For example, literature\textsuperscript{17} introduced the concept of autonomous vehicle clouds as “vehicle clusters” consisting of highly autonomous vehicles form a vehicular cloud. Its computing, sensing, communication, and physical resources are coordinated and dynamically assigned to authorized users. Literature\textsuperscript{17} defined the taxonomy for multiple clouds and came up with the concept of “grid” for vehicular services. For example, the hierarchical Vehi-Cloud architecture includes the IoV physical infrastructure layer and the V2V layer, in which each layer can access the cloud computing resources; the ITS-cloud is composed of non-scalable cloud layer, communication layer, and end-user layer, which aims to optimize the security and reliability of applications. Literature\textsuperscript{18} extending the concept of layering to cloud computing, designed a layered and distributed framework of cloud computing to provide user-oriented services. Literature\textsuperscript{11} proposed a novel layered vehicular cloud architecture, including central cloud layer, roadside cloud layer, and vehicular cloud layer. It further introduced the benefits of the layered cloud architecture in managing vehicle traffic and optimizing IoV resources. The MEC-cloud computing collaboration architecture as shown in Figure 1 is a dynamic and open-layered architecture, in which vehicular users can dynamically access cloud layer and/or edge layer for computing, storage and other services.
With the continuous development of cloud computing and edge computing, this architecture combining their advantages can provide high-quality services for the entire network system in a coordinated and effective manner. Cloud computing (at the top of Figure 1) equipped with powerful computing capacity, has developed rapidly in recent years and has been widely used in education, communication, medical and other fields. However, the inherent drawbacks of cloud computing, such as high latency and network overloads, etc., hinder the deployment of new IoT services and promote the emergence of cloud-based edge computing (at the middle layer of Figure 1). Edge computing is scalable and geographically closed to end users. It can be used to support real-time applications, save communication bandwidth, reduce the workload of cloud data center and improve resource utilization. In order to adapt to future application scenarios, it is necessary to combine cloud computing with MEC technology and upgrade traditional wireless base stations to intelligent base stations. Therefore, we propose the hierarchical MEC-cloud computing collaboration architecture including the cloud layer, MEC layer, and end-users layer. Applications running on the end users can dynamically select cloud layer or MEC layer or both to provide services according to QoS requirements.

In particular, the MEC-cloud computing collaboration architecture can efficiently handle IoV big data and provide services such as mobility support, service customization, cloud resource selection, distributed computing/storage, etc. Finally, the layered architecture utilizes the cloud and MEC resources in a coordinated manner assisted by SDN technology. This architecture not only can create new services, but also improve system performance, such as execution time, parallel execution, edge processing, MEC security, resource utilization, and energy efficiency.

3 ENERGY-EFFICIENT WORKLOAD ALLOCATION

3.1 System framework

The system framework of the interaction between MEC and cloud computing is shown in the left part of Figure 2. The RSU acts as a user interaction interface for receiving vehicular requests. RSUs then input these requests to the connected
MEC servers through the local area network (LAN). Owing to the vehicle is geographically close to the MEC server, the LAN communication delay can be ignored (compared to the wide area network [WAN]). There is a group of cloud servers in the cloud computing subsystem. Each cloud server is equipped with several computers of the same type, and the vehicular requests are partially offloaded from each MEC server to the cloud server through the WAN. Due to the large coverage of the WAN and the huge amount of traffic data communicated through the WAN, the communication delay and bandwidth constraints in the WAN should not be ignored. In the rest of the paper, the power consumption and computing/communication delay are mainly considered in the three subsystems, namely the edge computing subsystem, the WAN communication subsystem and the cloud computing subsystem. We list some significant symbols and their meanings in Table 1. In addition, mathematical notations like the all-ones vector $\mathbf{1}$ and the all-zeros vector $\mathbf{0}$ are also used in the following.

### 3.1.1 MEC subsystem

**Power consumption:** For simplicity, we can model the power consumption of MEC server $i$ as a function of the computation workload $x_i$, such as quadratic functions and piecewise linear functions.\(^{19}\) Without loss of generality, we use the following monotonically increasing and convex function to denote the power consumption $P_{\text{MEC}}^i$ of MEC server $i$

$$P_{\text{MEC}}^i = a_i x_i^2 + b_i x_i + c_i, \quad (1)$$

where $a_i > 0$ and $b_i, c_i \geq 0$ are predetermined parameters.
Computation delay: With request arrival rate $x_i$ and service rate $v_i$, the MEC server $i$ is considered as a queuing system and we define its computation delay $D^\text{MEC}_i$ as

$$D^\text{MEC}_i \triangleq \frac{1}{v_i - x_i}.$$

(2)

3.1.2 Cloud computing subsystem

Power consumption: We assume that every cloud server is equipped with a set of homogeneous computers and each computer at the same cloud server has the same configurations (e.g., CPU frequency) and power consumption profile. The power consumption of each computer at cloud server $j$ is approximated as a function of the computer CPU frequency $f_j$:

$$P_j = A_j f_j^e + B_j,$$

where $A_j$ and $B_j$ are positive constants, and $e$ varies from 2.5 to 3.\textsuperscript{20}

In the cloud computing subsystem, the working state of the cloud server changes with the workload.\textsuperscript{21} The operational/hibernating state of cloud server $j$ is denoted by a binary variable $O_j$ (i.e., 1 denotes the server is operational and 0 is hibernating). In addition, the quantity of operational computers at the cloud server $j$ is denoted by an integer variable $q_j$. Therefore, the power consumption $P_j^\text{cloud}$ of cloud server $j$ is given as\textsuperscript{22}

$$P_j^\text{cloud} \triangleq O_j q_j (A_j f_j^e + B_j).$$

(3)

Computation delay: Characterized by the Erlang-C (or M/M/n queuing) model, each cloud server’s computation delay is $[(C(n, \lambda/\mu))/(n\mu - \lambda) + (1/\mu)]$, where $n$ denotes the quantity of operational computers, $\lambda$ and $\mu$, respectively, denote the request arrival rate and service rate, and $C(n, \lambda/\mu)$ denotes the Erlang’s C formula [23, Ch. 2]. Assume that $\mu_j$ is the service rate of each computer at cloud server $j$. We can usually do the conversion: $\mu_j = f_j/K$, where $K$ is in units of #/(cycles)/request.

With the operational/hibernating state $O_j$, the quantity of operational computers $q_j$ and the traffic arrival rate $y_j$, the computation delay $D_j^\text{cloud}$ of cloud server $j$ is given as

$$D_j^\text{cloud} \triangleq O_j \left[ \frac{C(q_j, y_j K/f_j)}{q_j f_j/K - y_j} + \frac{K}{f_j} \right].$$

(4)

3.1.3 WAN subsystem

Communication delay: The transmission delay of the WAN path from MEC server $i$ to cloud server $j$ is denoted by $d_{ij}$.\textsuperscript{24} Therefore, when the MEC server $i$ dispatches request rate $\lambda_{ij}$ to the cloud server $j$, the communication delay in WAN $D_{ij}^\text{comm}$ is defined as

$$D_{ij}^\text{comm} \triangleq d_{ij} \lambda_{ij}.$$

(5)

3.2 Problem formulation

3.2.1 Power consumption minimized problem formulation

The power consumption of the MEC-cloud computing system is mainly generated by all MEC servers and computers equipped at cloud servers. The system power consumption function as the objective function is defined as

$$P_{\text{sys}} \triangleq \sum_{i=1}^{N} P_{\text{MEC}}^i + \sum_{j=1}^{M} P_j^\text{cloud}.$$

(6)

Constraints include the workload balance constraints, MEC server constraints, cloud server constraints and WAN communication bandwidth constraint that are discussed in the following. The decision variables include the workload
$x_i$ of MEC server $i$, the workload $y_j$ of cloud server $j$, MEC server $i$ dispatching the request rate $\lambda_{ij}$ to cloud server $j$, the computer CPU frequency $f_j$ and the operational/hibernating state of the cloud server $j$.

While minimizing the system power consumption, it is essential to guarantee the delay requirements, including computation delay and communication delay. The system delay function is then defined as

$$D_{\text{sys}} \triangleq \sum_{i=1}^{N} D_{\text{MEC}}^i + \sum_{j=1}^{M} D_{\text{cloud}}^j + \sum_{i=1}^{N} \sum_{j=1}^{M} D_{\text{comm}}^{ij}. \quad (7)$$

The PP of this paper is to minimize the power consumption of the system while ensuring the latency requirements:

$$\begin{align*}
\min_{x_i, y_j, \lambda_{ij}, f_j} & \quad P_{\text{sys}} \\
\text{s.t.} & \quad D_{\text{sys}} \leq D_{\text{th}} \\
& \quad \text{Workload balance constraints} \\
& \quad \text{MEC server constraints} \\
& \quad \text{Cloud server constraints} \\
& \quad \text{WAN communication bandwidth constraint}
\end{align*}$$

To design an energy-efficient workload allocation scheme in the MEC-cloud computing system, we make the tradeoff between the power consumption and the vehicular users experienced delay, which will be carefully discussed in the subsystems.

### 3.2.2 Constraints

**Constraints of workload balance:** $L$ is denoted as the sum of the request arrival rate at LAN, $l_i$ is denoted as the request rate distributed by RSUs to MEC server $i$. The relationship between $L$ and $l_i$ satisfies $L \triangleq \sum_{i=1}^{N} l_i$. Vehicular requests are processed by the MEC server or the cloud server. $X$ and $Y$ are, respectively, denoted as the request rate assigned to the MEC subsystem and the cloud computing subsystem, $x_i$ and $y_j$ are, respectively, denoted as the workload of MEC server $i$ and the workload of cloud server $j$. Their relationship satisfy $X \triangleq \sum_{i=1}^{N} x_i$, $Y \triangleq \sum_{j=1}^{M} y_j$ and $L = X + Y$. Moreover, the workload balance constraints of servers are

1. **Workload balance constraint of MEC servers**
   $$\sum_{j=1}^{M} \lambda_{ij} = l_i - x_i \quad \forall i \in \mathcal{N}. \quad (9)$$

2. **Workload balance constraint of cloud servers**
   $$\sum_{i=1}^{N} \lambda_{ij} = y_j \quad \forall j \in \mathcal{M}. \quad (10)$$

**Constraints of MEC servers:** The computing capacity of MEC server $i$ is limited, thus the workload $x_i$ of MEC server $i$ should be less than its maximum computing capacity $x_i^{\text{max}}$, while not exceed the request arrival rate $l_i$:

$$0 \leq x_i \leq \min_{i} \{x_i^{\text{max}}, l_i\} \quad \forall i \in \mathcal{N}. \quad (11)$$

**Constraints of cloud servers:** Cloud server $j$ constraints include

1. **the basic constraint**
   $$y_j \geq 0 \quad \forall j \in \mathcal{M}. \quad (12)$$
b. computation rate constraint

\[ f_{j}^\text{min} \leq f_{j} \leq f_{j}^\text{max} \quad \forall j \in \mathcal{M}. \]  

(13)

where \( f_{j}^\text{min} \) and \( f_{j}^\text{max} \) are the minimum and maximum setting of the computer CPU frequency for cloud server \( j \), respectively.

c. the quantity of computers

\[ q_{j} \in \{1, 2, \ldots, q_{j}^\text{max}\} \quad \forall j \in \mathcal{M}, \]  

(14)

where \( q_{j}^\text{max} \) denotes the upper bound of the quantity of computers at cloud server \( j \).

d. working state

\[ O_{j} \in \{0, 1\} \quad \forall j \in \mathcal{M}, \]  

(15)

which indicates the operational/hibernating working state of cloud server \( j \). \( O_{j} = 1 \) indicates that the cloud server \( j \) is operational; while \( O_{j} = 0 \) indicates that the cloud server \( j \) is hibernating and the quantity of operational computers equals 0.

Constraint of WAN communication bandwidth: The request rate dispatched from MEC server \( i \) to cloud server \( j \) is \( \lambda_{ij} \). Assume that these transmission paths do not interfere with each other, and the bandwidth used by each transmission path is limited. Therefore, the constraint of WAN communication bandwidth is

\[ 0 \leq \lambda_{ij} \leq \lambda_{ij}^\text{max} \quad \forall i \in \mathcal{N}; \quad \forall j \in \mathcal{M}. \]  

(16)

3.3 Decomposition and solution

Note that in PP, the decision variables come from different subsystems and are tightly coupled with each other, which makes the relationship between the workload allocation and the power consumption-delay tradeoff not clear. Moreover, the decision variables contain integer variables and linear variables, which makes PP general nonconvex and NP-hard. To address this issue effectively, we develop an approximate approach to decompose PP into three SPs corresponding to the edge computing subsystem, cloud computing subsystem, and WAN subsystem, which can be, respectively, solved via existing optimization techniques. The decomposition and the interaction of each SP/subsystem are illustrated in the right part of Figure 2.

3.3.1 Energy-efficient workload allocation in edge computing subsystem

SP1 is obtained by minimizing the weighted sum of power consumption and computation completion time in the edge computing subsystem:

\[
\min_{x_{i}} \sum_{i=1}^{N} \left( \gamma_{i} (a_{i}x_{i}^{2} + b_{i}x_{i} + c_{i}) + \frac{\gamma_{i}' x_{i}}{v_{i} - x_{i}} \right) \\
\text{s.t.} \quad \sum_{i=1}^{N} x_{i} = X
\]  

(17)

where the adjustable parameters \( \gamma_{i}, \gamma_{i}' \in [0, 1] \) denote the weights of power consumption and computation completion time for the MEC server \( i \), respectively.

Given the workload \( X \) assigned to the edge computing subsystem, SP1 is a convex optimization problem with linear constraints, and the interior point method (References 25-27, chapter 11) is used to solve this problem in Appendix A1.
After obtaining the optimal workload allocation $x_i^*$ for MEC server $i$, the power consumption and computation delay of the edge computing subsystem can be calculated, respectively:

$$
\begin{align}
P_{\text{MEC}}(X) &= \sum_{i=1}^{N} a_i (x_i^*)^2 + b_i x_i^* + c_i \\
D_{\text{MEC}}(X) &= \sum_{i=1}^{N} \frac{1}{v_i - x_i^*}
\end{align}
$$

(18)

### 3.3.2 Energy-efficient workload allocation in cloud computing subsystem

The service response delay of cloud server $j$ should be less than a certain threshold according to the service level agreement, when it processes some delay-sensitive requests, that is $D_{\text{cloud}}^j \leq D_{\text{cloud,th}}^j$. SP2 is obtained by minimizing the power consumption and ensuring the delay constraint of the cloud computing subsystem:

$$
\begin{align}
\min_{y_j, f_j, q_j, O_j} & \sum_{j=1}^{M} O_j q_j [A_j f_j^p + B_j] \\
\text{s.t.} & \sum_{j=1}^{M} v_j = Y \\
& D_{\text{cloud}}^j \leq D_{\text{cloud,th}}^j \\
& (12) - (15)
\end{align}
$$

SP2 is a mixed-integer nonlinear programming (MINLP) problem, which is generally difficult to tackle. Since generalized Benders’ decomposition is an effective method to solve this problem with guaranteed optimality, given the workload $Y$ of the cloud computing subsystem, we use generalized Benders’ decomposition (GBD) (References 28-30, chapter 13) method to solve SP2 in Appendix B1. After obtaining the optimal workload $y_j^*$ of cloud server $j$ and its optimal CPU frequency $f_j^*$, the quantity of operational computers $q_j^*$ and the working state $O_j^*$, the power consumption and computation delay of the cloud computing subsystem can be calculated separately:

$$
\begin{align}
P_{\text{cloud}}(Y) &= \sum_{j=1}^{M} O_j^* q_j^* [A_j (f_j^*)^p + B_j] \\
D_{\text{cloud}}^j &= \sum_{j=1}^{M} D_{\text{cloud}}^j = \sum_{j=1}^{M} O_j^* D_{\text{cloud,th}}^j
\end{align}
$$

(20)

### 3.3.3 Delay minimization for dispatching in WAN subsystem

SP3 is obtained by minimizing the communication delay in the WAN subsystem to determine the optimal request rate $\lambda_{ij}$ dispatched by MEC server $i$ to cloud server $j$:

$$
\begin{align}
\min_{\lambda_{ij}} & \sum_{i=1}^{N} \sum_{j=1}^{M} d_{ij} \lambda_{ij} \\
\text{s.t.} & (9), (10), (16)
\end{align}
$$

(21)

Given the workload assigned to edge computing subsystem $X$ and cloud computing subsystem $Y$, the optimal workload $x_i^*$ for the MEC server $i$ and $y_j^*$ for cloud server $j$ are determined from (17) and (19), respectively. Therefore, SP3 is regarded as a task assignment problem. To get the optimal combination of SP3 in polynomial time using Hungarian method, the detailed Hungarian algorithm steps are illustrated in Appendix C1. After obtaining the optimal request rate $\lambda_{ij}^*$ dispatched by MEC server $i$ to cloud server $j$, the communication delay in the WAN subsystem can be calculated:
\[ D^\text{comm}(X, Y) = \sum_{i=1}^{N} \sum_{j=1}^{M} d_{ij} x_{ij}. \]  

(22)

### 3.3.4 Energy-efficient workload allocation in MEC-cloud computing system

The system power consumption comes from MEC servers and cloud servers. Therefore, the function of the system power consumption is defined as

\[ P^\text{sys}(X, Y) \triangleq P^\text{MEC}(X) + P^\text{cloud}(Y). \]  

(23)

The latency of the system includes the MEC server computation delay, the cloud server computation delay and the WAN communication delay. Therefore, the function of the system latency is defined as

\[ D^\text{sys}(X, Y) \triangleq D^\text{MEC}(X) + D^\text{cloud}(Y) + D^\text{comm}(X, Y). \]  

(24)

After solving the above three SPs, we can approximately solve PP by considering the following approximate problem named the PP-approx:

\[
\begin{align*}
\min_{X, Y} & \quad P^\text{sys}(X, Y) \\
\text{s.t.} & \quad D^\text{sys}(X, Y) \leq D^\text{th} \\
& \quad X + Y = L
\end{align*}
\]  

(25)

Combining the algorithms of the above three SPs, we design an iterative algorithm to solve this problem in Appendix D1. The approximation ratio is dependent on the choice of the adjustable parameters \( \gamma_i, \gamma^l_i \) and \( D^\text{cloud,th} \). If these parameters could be chosen appropriately, then the solution to PP-approx would be the optimal solution to PP. How to evaluate the approximation ratio of the proposed decomposition is left as our future work.

## 4 NUMERICAL RESULTS

For simplicity but without loss of generality, in the simulation, we simulate a MEC-cloud computing system with five MEC servers and three cloud servers (regarded as an IDC). The similar results can be obtained if the system includes more MEC servers and more cloud servers. Referring to literatures, let \( e = 3 \), \( K = 1 \), and some important parameter values are shown in Tables 2 and 3. The MATLAB simulation platform is used to obtain the following results. For comparison, we take the following two most related schemes as benchmarks.

- **Optimal allocation without DFS:** This scheme corresponds to solving the SPs under fixed \( f_j \). The resulting optimization problem can be solved by using similar method in this paper. Here, we set \( f_j = 0.6f_j^\text{max} \).
- **Average allocation:** In this scheme, the total request arrival rate at LAN is averagely allocated to the edge computing subsystem and the cloud system, that is, \( X = Y = \frac{L}{2} \).

We first evaluate the convergence of the proposed iterative algorithm and the impact of the workload of LAN on the system power consumption. We plot the convergence of workload \( L = 6 \times 10^4, 8 \times 10^4, 10 \times 10^4 \) by the proposed iterative algorithm for solving PP and the dynamics of their power consumption in Figure 3. We can observe that (i) the proposed

| i | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| \( l_i \) | 30000 | 15000 | 15000 | 20000 | 20000 |

| TABLE 2 Parameter setup of the mobile edge computing server |


TABLE 3  Parameter setup of the cloud server

|   | 1     | 2     | 3     |
|---|-------|-------|-------|
| $A_j$ | 3.2606 | 4.485 | 2.370 |
| $B_j$ | 68    | 53    | 70    |
| $f_{j}^{\text{min}}$ | 1     | 1     | 1     |
| $f_{j}^{\text{max}}$ | 3.4   | 2.4   | 3.0   |
| $m_{j}^{\text{max}}$ | 30 000 | 60 000 | 25 000 |
| $D_{j}^{\text{cloud,th}}$ | 1     | 1     | 1     |

FIGURE 3  Simulation dynamics of system power consumption with different workload $L$

iterative algorithm can achieve converge about 100 to 120 iterations for all workloads. An iteration takes 0.3 to 0.5ms, therefore, our iterative algorithm takes at most 60 ms to finish the heaviest workload; (ii) the heavier workload has the lower speed of convergence, that is, the workload of LAN can increase the number of iterations.

To evaluate the power consumption $P_{\text{MEC}}(X)$ and computation delay $D_{\text{MEC}}(X)$ of the edge computing subsystem, we change the workload $X$ of MEC subsystem within $0 \text{ to } 10^4$. The algorithm in Appendix A is used to solve SP1 and determine the optimal workload allocated for the MEC server $i$, then we calculate $P_{\text{MEC}}(X)$ and $D_{\text{MEC}}(X)$. The results are shown in Figure 4A. It can be seen from Figure 4A that both the power consumption and computation delay of the edge computing subsystem grow with the workload assigned for edge computing subsystem.

We then change the workload $Y$ of cloud computing subsystem within $10^4 \text{ to } 10^5$, and solve SP2 using the algorithm in Appendix B to determine the optimal workload $y_{j}^*$ allocated for the cloud server $j$ and working state $f_{j}^*$, $m_{j}^*$, $O_{j}^*$. Then, we calculate the power consumption of cloud computing subsystem $P_{\text{cloud}}(Y)$ and the computation delay $D_{\text{MEC}}(Y)$ respectively, the results are shown in Figure 4B. It can be seen from Figure 4B that the power consumption of the cloud computing subsystem increases with the workload assigned for cloud computing subsystem, and the computation delay keeps equal to the delay constraint of cloud computing subsystem.

With those determined variables, we solve SP3 using the algorithm in Appendix C to obtain the workload dispatching delay in the WAN subsystem $D_{\text{comm}}(X,Y)$ and calculate the power consumption $P_{\text{sys}}(X,Y)$ and the latency $D_{\text{sys}}(X,Y)$ of the system, respectively. The results are shown in Figure 4C. By comparing the curves in Figure 4, we note that when the workload of edge computing subsystem becomes heavy, the power consumption of MEC servers increases and the power consumption of the system also increase, while the workload of cloud computing subsystem, the communication delay of WAN subsystem and the system latency also decrease, and vice versa.

Therefore, we can make the conclusion that the system power consumption is mainly generated by the edge computing subsystem, while the system latency is mainly generated by the WAN subsystem. The reason is that, in the MEC-cloud computing system, the computing capacity of cloud servers is more abundant and more energy-efficient compared with MEC. However, MEC has the advantage of being geographically closed to end users that enables MEC to
reduce communication delay and process some computing requests without consuming WAN bandwidth. The conclusions of the simulations in this section are of great significance for designing efficient computing offloading and resource (such as communication and computing resources) management schemes in the MEC-cloud computing system.

Based on the above conclusions, we design a bipartite iterative algorithm in Appendix D1to obtain the optimal workload allocation scheme of the system, then, calculate the power consumption and the latency of the system. As shown in Figure 5, the workload allocation scheme proposed in this paper is significantly better than the other two schemes. First, our scheme can dynamically adjust the workload of edge computing subsystem and cloud computing subsystem based on their computing capacity. Then, our scheme applies DVFS technology, each cloud server can adjust its CPU frequency according to its workload, which effectively saves power consumption and reduces system latency. Finally, our scheme can control the operational/hibernation working state of the cloud server and the quantity of operational computers based on the workload of cloud computing subsystem, that further reduces system power consumption.
5 | CONCLUSION

This paper has introduced the architecture and vision of collaborative MEC-cloud computing in IoV, and designed a layered framework for interactions of MEC and cloud computing, to study the power consumption and latency of the system. Based on workload balancing and dynamic power control, we propose an energy-efficiency optimized workload allocation scheme. The PP is decomposed into three SPs corresponding to three subsystems, which can be solved within, respectively. Simulation results show that the cooperation of MEC and cloud computing can improve the performance of the system, and the workload allocation scheme proposed in this paper effectively reduces system power consumption and latency. In the future work, we will consider the computing capacity of vehicles and study the collaboration among local computing, edge computing, and cloud computing to minimize the computing latency in IoV.

CONFLICT OF INTEREST

No conflict of interest exits in the submission of this manuscript, and manuscript is approved by all authors for publication. We would like to declare that the work described was original research that has not been published previously, and not under consideration for publication elsewhere, in whole or in part. All the authors listed have approved the manuscript that is enclosed.

ORCID

Xiaohui Gu https://orcid.org/0000-0001-8406-0702
Guoan Zhang https://orcid.org/0000-0001-8439-3478

REFERENCES

1. Zhao L, Wang J, Liu J, Kato N. Routing for crowd management in smart cities: a deep reinforcement learning perspective. IEEE Commun Mag. 2019;57(4):88-93.
2. Wan J, Zhang D, Sun Y, Lin K, Zou C, Cai H. VCMIA: a novel architecture for Integrating vehicular cyber-physical systems and mobile cloud computing. Mob Netw Appl. 2014;19(2):153-160.
3. Wang C, Li Y, Jin D, Chen S. On the serviceability of mobile vehicular clouds: a large-scale urban environment. IEEE Trans Intell Transp Syst. 2016;17(10):2960-2970.
4. Abbas N, Zhang Y, Taherkordi A, Skeie T. Mobile edge computing: a survey. IEEE IoT J. 2017;5(1):450-465.
5. Mach P, Becvar Z. Mobile edge computing: a survey on architecture and computation of offloading. IEEE Commun Surv Tutor. 2017;19(3):1628-1656.
6. de Assuncao MD, da Silva VA, Buiya R. Distributed data stream processing and edge computing: a survey on resource elasticity and future directions. J Netw Comput Appl. 2018;103:1-17.
7. Tran TX, Hajisami A, Pandey P, Pompili D. Collaborative mobile edge computing in 5G networks: new paradigms, scenarios, and challenges. IEEE Commun Mag. 2017;55(4):54-61.
8. Taleb T, Dutta S, Ksentini A, Iqbal M, Flinck H. Mobile edge computing potential in making cities smarter. IEEE Commun Mag. 2017;55(3):38-43.
9. Mao Y, You C, Zhang J, Huang K, Letaief KB. A survey on mobile edge computing: the communication perspective. IEEE Commun Surv Tutor. 2017;19(4):2322-2358.
10. Salman O, Elhajj I, Kayssi A, Chehab A. Edge computing enabling the Internet of Things. Paper presented at: Proceedings of the 2015 IEEE 2nd World Forum on Internet of Things (WF-IoT); Milan, Italy: 2015:603-608; IEEE.
11. Yu R, Zhang Y, Gjessing S, Xia W, Yang K. Toward cloud-based vehicular networks with efficient resource management. IEEE Netw. 2013;27(5):48-55.
12. Zhang K, Mao Y, Leng S, Maharjan S, Vinel A, Zhang Y. Contract-theoretic approach for delay constrained offloading in vehicular edge computing networks. Mob Netw Appl. 2019;24(3):1003-1014.
13. Deng D, Lien S, Lin C, Hung S, Chen W. Latency control in software-defined mobile-edge vehicular networking. IEEE Commun Mag. 2017;55(8):87-93.
14. Zhou Z, Chen X, Li E, Zeng L, Luo K, Zhang J. Edge intelligence: paving the last mile of artificial intelligence with edge computing. Proc IEEE. 2019;107(8):1738-1762.
15. Hong J, Yin J, Liu Y, Peng J, Jiang H. Energy management and control strategy of photovoltaic/battery hybrid distributed power generation systems with an integrated three-port power converter. IEEE Access. 2019;7:82838-82847.
16. Zhao B, Wang X, Lin D, et al. Energy management of multiple microgrids based on a system of systems architecture. IEEE Trans Pow Syst. 2018;33(6):6410-6421.
17. Dana P. Consuming resources and services from multiple clouds. J Grid Comput. 2014;12(2):321-345.
18. Wang S, Yan K, Wang S, Chen C. A three-phases scheduling in a hierarchical cloud computing network. Paper presented at: Proceedings of the 2011 3rd International Conference on Communications and Mobile Computing; Qingdao, China: IEEE; 2011:114-117.
APPENDIX A. SOLVE SP1 USING THE INTERIOR POINT METHOD

**Definition 1.** Define $x, x^{\text{max}}, l$, as the vectors of $x_i, x_i^{\text{max}}, l_i$. We follow chapter 11 of Reference 27 to solve SP1 using the interior point method.

**Definition 2.** Objective function $f_0(x)$ and constraint function $f_1(x)$.

\[
\begin{align*}
  f_0(x) & \triangleq \sum_{i=1}^{N} \left( \gamma_i (a_i x_i^2 + b_i x_i + c_i) + \frac{x_i}{\gamma_i - x_i} \right) \\
  f_1(x) & \triangleq x - \min \{x^{\text{max}}, l\}. \\
\end{align*}
\]  \hspace{0.5cm} (A1)

Thus, SP1 can be denoted as

\[
\begin{align*}
  \min_{x_i} f_0(x) \\
  \text{s.t.} \quad & f_1(x) \leq 0 \\
  & Ax = X,
\end{align*}
\]  \hspace{0.5cm} (A2)

where $A = [1 \ldots 1]$. To make (A2) with inequality constraints be approximate to the one with equality constraints, (A2) can be rewritten as

\[
\begin{align*}
  \min_{x_i} f_0(x) + \lambda (f_1(x)) \\
  \text{s.t.} \quad & Ax = X,
\end{align*}
\]  \hspace{0.5cm} (A3)
where \( I_u = \begin{cases} 0 & u \leq 0 \\ \infty & u > 0 \end{cases} \), and its approximate function is \( \hat{I}_u = -\left( \frac{1}{t} \right) \log(-u) \), where \( t > 0 \) is the parameter that decides the approximate accuracy. By replacing \( I_u \) with \( \hat{I}_u \) in (A3), the approximate form of (A2) is given as

\[
\min_{x \in X} f_0(x) - \left( \frac{1}{t} \right) \log(-f_1(x))
\]

s.t. \( Ax = X \). (A4)

Then the logarithmic barrier function of SP1 is defined as

\[
\phi(x) = -\log(-f_1(x)) \quad \text{dom}\phi = \{x | f_1(x) < 0\} \tag{A5}
\]

To simplify the symbols, we multiply the objective function of (A4) by \( t \), and get its equality

\[
\min_{x \in X} tf_0(x) + \phi(x)
\]

s.t. \( Ax = X \). (A6)

The optimal solutions for (A4) and (A5) are the same. For \( t > 0 \), the optimization problem (A6) with an equality constraint can be solved using the Newton method and obtain its unique solution \( x^* \). We can iteratively calculate the corresponding \( x^* \) for a series of increasing \( t \) values until \( t \geq 1/\varepsilon \), where \( \varepsilon \) is the predetermined accuracy parameter. The iterative steps are summarized in Algorithm 1.

**Algorithm 1.** Interior point method for solving SP1

/* Initialization */

Given strictly feasible points \( x, t; t^{(0)} > 0, \mu > 1, \varepsilon > 0 \)

while \( t \geq \frac{1}{\varepsilon} \) do

/* Obtain a unique solution \( x^*(t) \) of (31) based on the Newton method */

Calculate the Newton direction \( \Delta x_{ng} \) by Equation (A11);

Calculate the Newton decrement \( \lambda(x) \) by Equation (A12);

while \( \lambda^2(x)/2 \geq \varepsilon \) do

/* Determine the step size \( g \) based on backing line search method */

Given \( \alpha \in (0, 0.5); \beta \in (0, 1); g = 1 \);

while \( f_2(x + p\Delta x_{ng}) \leq f_2(x) + \alpha p \nabla f_2(x)^T \Delta x_{ng} \) do

\( g \leftarrow \beta p \);

end while

\( x^* \leftarrow x + g\Delta x_{ng} \);

end while

\( x \leftarrow x^*(t) \);

\( t \leftarrow \mu t \);

end while

Solve (A6) using Newton Method.

**Definition 3.** Objective function \( f_2(x) \).

\[
f_2(x) = tf_0(x) + \phi(x). \tag{A7}
\]
Thus, (A6) can be denoted as

\[
\min_{x_i} f_2(x) \\
\text{s.t. } Ax = X.
\]  

(A8)

At the feasible point \( x \), to derive the Newton direction \( \Delta x_{ng} \) of problem (A8) with an equality constraint, we replace the objective function of (A8) with its second-derivative Taylor approximation around \( x \)

\[
\min_{x_i} f_2(x + v) = f_2(x) + \nabla f_2(x)^T v + (1/2)v^T \nabla^2 f_2(x)v \\
\text{s.t. } Ax = X,
\]

(A9)

where \( v \) is the variable. And (A9) is a convex quadratic minimization problem with an equality constraint. The Newton direction \( \Delta x_{ng} \) at the feasible point \( x \) is determined by

\[
\begin{bmatrix}
\nabla^2 f_2(x) \\
A
\end{bmatrix}
\begin{bmatrix}
\Delta x_{ng}
\w
\end{bmatrix}
= 
\begin{bmatrix}
-\nabla f_2(x)
\end{bmatrix},
\]

(A10)

where \( \w \) is the optimal dual variable of the quadratic problem (A9). Problem (A10) can be solved by

\[
\begin{aligned}
\nabla^2 f_2(x) \Delta x_{ng} + A^T \w &= \nabla f_2(x) \\
A \Delta x_{ng} &= 0
\end{aligned}
\Rightarrow
\begin{aligned}
\Delta x_{ng} &= -\left( \nabla f_2(x) + A^T \w \left( \nabla^2 f_2(x)^{-1} \right)^T \right) \\
\w &= (A \nabla^2 f_2(x)^{-1} A^T)^{-1} \left( -A \nabla^2 f_2(x)^{-1} \nabla f_2(x) \right).
\end{aligned}
\]

(A11)

The Newton decrement for problem (A8) is defined as

\[
\lambda(x) = (\Delta x_{ng}^T \nabla^2 f_2(x) \Delta x_{ng})^{1/2}.
\]

(A12)

**APPENDIX B. SOLVE SP2 USING GBD ALGORITHM**

**Definition 4.** Define \( y, f, q, O \) as the vectors of \( y_j, f_j, q_j, O_j \), and \( Y, F, Q, O \) as the definition domains of \( y_j, f_j, q_j, O_j \), that is, (4) to (7).

We use the GBD method to solve SP2, referring to chapter 13 of Reference 32. For MINLP SP2, continuous variables are \( y \) and \( f \), integer variables are \( q \) and \( O \); the optimal solutions are denoted as \( y^*, f^*, q^*, \) and \( O^* \). Obviously, the key to solve MINLP SP2 is to find the optimal solution \( q^* \) and \( O^* \). When we obtain the integer solution, MINLP is simplified to a linear programming (LP) problem, that is, after \( q^* \) and \( O^* \) are determined, it is not difficult to obtain \( y^* \) and \( f^* \).

The GBD algorithm is used to iteratively solve MINLP SP2. The underlying steps are described in the following. Based on the GBD method, MINLP can be decomposed into a master problem (MP) (B3), which is an integer optimization problem, and a SP (B4), which is a linear optimization problem. The integer constraints are considered in the MP to optimize the integer variables and obtain the lower bound (LB) of the original problem. After obtaining the integer solution, MINLP is reduced to SP to optimize the continuous variables and obtain the upper bound (UB) of the original problem. When the predicted LB is bigger than or equal to the UB, the algorithm terminates. Generally, the initially obtained integer variables are not the optimal solution, but we can narrow the search/feasible space by adding new integer constraints in the MP to approximate the optimum. In other words, SP is able to find the feasible solution but UB is larger than LB (ie, the determined \( q \) and \( O \) are not the optimal solution). In order to improve the optimal solution of \( q \) and \( O \), the feasible constraint (40a) is added to MP such that the new LB is larger than the previous LB. When the SP is infeasible to find a feasible solution, to prevent obtaining the infeasible solution again, an infeasible constraint (40b) is added to the MP. When it meets the stopping criterion \( |UB - LB| \leq \epsilon \), the optimal solution is converged, where \( \epsilon \) is the rate of fault tolerance. The iterative process is summarized in Algorithm 2, and the involved definitions are listed below.
Algorithm 2. GBD Algorithm for solving SP2

/* Initialization */
k ← 1, I ← ∅, J ← ∅, UB0 ← +∞;
while do
    Solve MPk based on the branch and bound method;
    if solution is feasible then
        The feasible solution \((q^k, O^k, LB^k)\) is obtained;
    else if solution is unbounded then
        Input \(q^k \in Q\) and \(O^k \in O\)
        \(LB^k ← −∞\);
    end if
    Solve \(SP(q^k, O^k)\) based on the dual decomposition method;
    if solution is feasible then
        Solution \((y^k, f^k)\) and Lagrangian multipliers \((\mu^k, \beta^k)\) are obtained;
        \(UB^k ← \min\{UB^{k-1}, F(f^k, q^k, O^k)\}\);
        if \(|UB^k − LB^k| ≤ \epsilon\) then /* Converged */
            return \((y^k, f^k, q^k, O^k)\);
        else /* Add feasible constraint */
            \(I^{k+1} ← I^k \cup \{k\}, J^{k+1} ← J^k\);
        end if
    else if solution is infeasible then
        Solve \(SPF(q^k, O^k)\) based on the dual decomposition method;
        Solution \((y^k, f^k)\) and Lagrangian multipliers \((\mu^k, \beta^k)\) are obtained;
        \(UB^k ← UB^{k-1}\);
        /* Add infeasible constraint */
        \(I^{k+1} ← I^k, J^{k+1} ← J^k \cup \{k\}\);
    end if
    \(k ← k + 1\);
end while

Definition 5. Objective function \(F(f, q, O)\) and constraint functions \(G(y, f, q, O)\)

\[
\begin{align*}
F(f, q, O) & \triangleq \sum_{j=1}^{M} O_j (A_j f^e_j + B_j) \\
H(y) & \triangleq \sum_{j=1}^{M} Y_j - Y \\
G(y, f, q, O) & \triangleq \begin{bmatrix} g_1, \ldots, g_l, \ldots, g_M \end{bmatrix}^T \\
g_i & \triangleq O_j \left[ \frac{C_i q_j f_j}{q_j f_j/k-y_j} + \frac{k}{y_j} \right] - D_{i}^{\text{cloud,th}}
\end{align*}
\]
Thus, MINLP SP2 is given as

\[
\begin{align*}
\min_{y \in Y, f \in F, q, O} & \quad \mathcal{L}(f, q, O) \\
\text{s.t.} & \quad \mathcal{H}(y) = 0 \\
& \quad \mathcal{G}(y, f, q, O) \leq 0.
\end{align*}
\]

(B2)

**Definition 6.** MP\(^k\)

\[
\begin{align*}
\min_{q \in Q, O \in O, LB} & \quad LB, \\
\text{s.t.} & \quad LB \geq \mathcal{L}(f^i, q, O) + \mu^i \mathcal{H}(y) + (\beta^i)^T \mathcal{G}(y, f, q, O) \quad \forall i \in I^k \\
& \quad 0 \geq \mu^i \mathcal{H}(y^j) + (\beta^i)^T \mathcal{G}(y^j, f^i, q, O) \quad \forall j \in J^k.
\end{align*}
\]

(B3)

**Definition 7.** SP\((q^k, O^k)\)

\[
\begin{align*}
\min_{y \in Y, f \in F} & \quad \mathcal{L}(f, q^k, O^k) \\
\text{s.t.} & \quad \mathcal{H}(y) = 0 \\
& \quad \mathcal{G}(y, f, q^k, O^k) \leq 0
\end{align*}
\]

(B4)

**Definition 8.** SP feasibility-check \(SP^F(q^k, O^k)\)

\[
\begin{align*}
\min_{y \in Y, f \in F} & \quad 1^T s \\
\text{s.t.} & \quad \mathcal{H}(y) = 0 \\
& \quad s \geq \mathcal{G}(y, f, q^k, O^k).
\end{align*}
\]

(B5)

**APPENDIX C. SOLVE SP3 USING HUNGARIAN ALGORITHM**

The Hungarian algorithm is a combinatorial optimization method which solves the task assignment problem in polynomial time. To transform SP3 into the standard form of the assignment problem equivalently, we first make the definition

\[
C_{ij} \triangleq \min\{j_i - x_i, x_{ij}^{\text{max}}, y_j\} \quad \forall i \in \mathcal{N}, \forall j \in \mathcal{M}.
\]

(C1)

Thus, the standard assignment problem is given by

\[
\begin{align*}
\min_{y \in Y, f \in F} & \quad \sum_{i=1}^{N} \sum_{j=1}^{M} d_{ij} C_{ij} a_{ij} \\
\text{s.t.} & \quad \sum_{j=1}^{M} a_{ij} = 1 \quad \forall i \in \mathcal{N} \\
& \quad \sum_{i=1}^{N} a_{ij} = 1 \quad \forall j \in \mathcal{M} \\
& \quad 0 \leq a_{ij} \leq 1 \quad \forall i \in \mathcal{N}, \forall j \in \mathcal{M}
\end{align*}
\]

where \(a_{ij}\) denotes that the assignment of MEC server \(i\) to cloud server \(j\), and the completed assignment is denoted by \(a_{ij} = 1\), otherwise \(a_{ij} = 0\). Since the constraint matrix is completely unimodular, we can always find an optimal integer solution with relaxation of \(a_{ij}\). In order to illustrate the application of the Hungarian algorithm to (C2) specifically, without loss of generality, a simple case is considered in this paper with \(|\mathcal{N}| = 5, |\mathcal{M}| = 3\). Since the rows of the cost matrix \(\mathcal{N}\) should be equal to its columns \(\mathcal{M}\), we add two additional dummy cloud servers CS\(_4\) and CS\(_5\). The above
problem can be illustrated graphically: five MEC servers MECS1, MECS2, MECS3, MECS4, and MECS5, three cloud servers, CS1, CS2 and CS3, and two dummy cloud servers CS4 and CS5. The lines from MECSi to MECSj represent the values of cost $d_{ij}C_{ij}$, with all $d_{14}C_{14}$ and $d_{15}C_{15}$ setting to 0. Without loss of generality, the cost matrix is defined as the $n \times n$ matrix

$$
C \triangleq 
\begin{bmatrix}
  d_{11}C_{11} & 0 & \ldots & d_{1n}C_{1n} \\
  \ldots & \ldots & \ldots & \ldots \\
  d_{n1}C_{n1} & 0 & \ldots & d_{nn}C_{nn}
\end{bmatrix}
$$

(C3)

The basic idea of the Hungarian algorithm: Subtract a constant from the rows and columns of the cost matrix $C$, reduce $C$ to contain $n$ zero elements located in different rows and different columns. Then, we get the optimal assignment of zeros which corresponds to the positions of the $n$ entries in the original cost matrix. Finally, the sum of the $n$ entries is the smallest possible cost.

**Algorithm 3.** Hungarian algorithm for solving (43)

Elements in each row minus the smallest element in the row;

Elements in each column minus the smallest element in the column;

while do

  Cover all zeros in the rows and columns of the cost matrix with the least horizontal or vertical lines;

  /* Check optimality */

  if the quantity of covering lines equals $n$ then

    return an optimal assignment of zeros which don’t lie in the same row or column;

  else if the quantity of covering lines is less than $n$ then

    /* An optimal assignment of zeros hasn’t obtained */

    Find the smallest element not covered by any line;

    Each uncovered row minus this element;

    Each covered column plus this element;

  end if

end while

**Theorem 1.** For the assignment problem, after adding (or subtracting) an identical number to any row (or column) of the cost matrix $C$, the optimal solution for the new cost matrix is also optimal for the original.32

Given a $n \times n$ cost matrix, Theorem 1 is applied in Algorithm 3 to find an optimal assignment. In order to clearly explain the Algorithm 3 used to solve (C2), without loss of generality, we demonstrate the matrix transformation process in the following. The detailed steps are as follows. Step 1: subtract 0 from each row and find the smallest element of each column. Step 2: each column subtract its smallest element, that is, subtract 35, 55, and 45 from column 1, column 2 and column 3, respectively, and subtract 0 from column 4 and 5. Step 3: cover all zeros in the rows and columns with the least horizontal or vertical lines. Since $n$ is bigger than the quantity of covering lines, we find the smallest entry not covered by any line is 10. Step 4: subtract 10 from all rows without covering lines, and add 10 to all columns with covering lines. Step 5: cover all zeros in the rows and columns with the least horizontal and vertical lines. Since the quantity of covering lines is 5, an optimal assignment of zeros is obtained. Start with a row or column with a minimum of 0 elements, circle all zeros with ○, and then cross out the remaining zeros in the same rows and columns. Step 6: circle the elements of the original cost matrix corresponding to
the positions of $\bigcirc$. Thus, the optimal task assignment is $\alpha^*_{12} = \alpha^*_{12} = \alpha^*_{22} = \alpha^*_{34} = \alpha^*_{45} = \alpha^*_{51} = 1$ with the smallest cost of 145.

After obtaining an optimal assignment of (C2), Algorithm 4 is used to update the corresponding parameters in SP3, then a new task assignment problem (C2) is formulated. Similarly, to get two sets of nodes with equal size and the corresponding cost, we add two additional dummy cloud servers. Again, Hungarian Algorithm 3 is applied to solve (C2), an optimal assignment is obtained, and Algorithm 4 is applied to update parameters in SP3. Repeat this process until all the unprocessed requests have been dispatched from MEC servers to cloud servers.

**Algorithm 4.** Update parameters in SP3

```plaintext
for $i \in \mathcal{N}, j \in \mathcal{M}$ do
    if $\alpha^*_y == 1$ then
        if $C_y \rightarrow l - x_i$ then /* Remove i */
            $\lambda^*_y \leftarrow l - x_i$
            $\mathcal{N} \leftarrow \mathcal{N} \setminus \{i\}$
            $y_j \leftarrow y_j - \lambda^*_y$
        else if $C_y \rightarrow \lambda_{y_{\text{max}}}$ then /* Remove i ~ j */
            $\lambda^*_y \leftarrow \lambda_{y_{\text{max}}}$
            $l_i - x_i \leftarrow l_i - x_i - \lambda^*_y$
            $y_j \leftarrow y_j - \lambda^*_y$
            $C_y \leftarrow \infty$
        else if $C_y \rightarrow y_j$ then /* Remove j */
            $\lambda^*_y \leftarrow y_j$
            $l_i - x_i \leftarrow l_i - x_i - \lambda^*_y$
            $\mathcal{M} \leftarrow \mathcal{M} \setminus \{j\}$
        end if
    end if
end for
```
APPENDIX D. APPENDIX

Algorithm 5. Iterative algorithm for solving PP

Ensure:

\( \{X^*, Y^*\} \): optimal workload allocation scheme for the system;

Input: \( L, X = Y = L/2, D^0 \), iteration index \( k \leftarrow 1 \), maximum number of iterations \( \text{Iter}_{\text{max}} \), an infinitesimal number \( \epsilon > 0 \);

while \( |X(k + 1) - X(k)| > \epsilon \) and \( k \leq \text{Iter}_{\text{max}} \) do

/* Edge computing subsystem */
Calculate \( P^{\text{MEC}} \) and \( D^{\text{MEC}} \) by Algorithm 1;
\( P^{\text{MEC}}(k + 1) = P^{\text{MEC}}(k) \);
\( D^{\text{MEC}}(k + 1) = D^{\text{MEC}}(k) \);

/* Cloud computing subsystem */
Calculate \( P^{\text{cloud}} \) and \( D^{\text{cloud}} \) by Algorithm 2;
\( P^{\text{cloud}}(k + 1) = P^{\text{cloud}}(k) \);
\( D^{\text{cloud}}(k + 1) = D^{\text{cloud}}(k) \);

/* WAN communication subsystem */
Calculate \( D^{\text{comm}}(k) \) by Algorithm 3 and 4;
\( D^{\text{comm}}(k + 1) = D^{\text{comm}}(k) \);

/* MEC-cloud computing system */
\( P^{\text{sys}}(k + 1) = P^{\text{MEC}}(k + 1) + P^{\text{cloud}}(k + 1) \);
\( D^{\text{sys}}(k + 1) = D^{\text{MEC}}(k + 1) + D^{\text{cloud}}(k + 1) + D^{\text{comm}}(k + 1) \);

/* Update parameters using dichotomy method */
if \( D^{\text{sys}}(k) \leq D^{0} \) then

\( X(k + 1) = X(k) - X(k)/2 \);

else

\( X(k + 1) = X(k) + X(k)/2 \);

end if

\( k = k + 1 \);

end while