Latency-Energy Joint Optimization for Task Offloading and Resource Allocation in MEC-Assisted Vehicular Networks

Yuliang Cong, Ke Xue, Cong Wang, *Member, IEEE*, Wenxi Sun, Shuxian Sun, and Fengye Hu, *Senior Member, IEEE*

***Abstract*—In this article, we study the task offloading problem on mobile edge in vehicular networks. Specifically, we take computational resource constraints into consideration, and aim to simultaneously reduce latency and energy consumption. For this purpose, we establish an offloading model that consists of local edge computing resources, edge server resources of both macro and subsidiary base stations, as well as cloud computing server resources. Each task can be offloaded through one of five strategies, and is evaluated via a loss function determined by its latency and energy consumption. Based on this model, our goal is to solve a mixed-integer non-linear optimization problem (MINLP) whose objective function is the weighted sum of the task-specific loss functions. To address this optimization problem, we split it into two sub-problems, referred to as resource allocation and offloading strategy. We develop a method based on Block Coordinate Descent technique combining convex optimization and Gray Wolf algorithm (BCD-CONGW) that alternatively solves the two sub-problems, until convergence. The former sub-problem is convex and can be solved in polynomial time, whereas the latter is non-convex and hence NP-hard. For the latter, we relax discrete variables and employ Gray Wolf algorithm with elite strategy to approximate its optimal point. By numerical evaluations, we show that our method outperforms existent methods in terms of latency and energy consumption.**

***Index Terms*—Vehicular networks, mobile edge computing, task offloading, resource allocation, joint optimization.**

I. Introduction

A. Background and Motivation

THE emergence of the Internet of Vehicles has promoted vigorous development of in-vehicle services, such as: autonomous driving, interactive applications, and traffic monitoring [1]. Meanwhile, the insufficient computing and storage resources of existing vehicle equipment could not meet the demands of the massive amount of data generated by these services. To address this problem, we need to re-visit the mechanism of computing and data communication for the Internet of Vehicles [2], [3], [4].

Although cloud computing can solve the problem of insufficient in-vehicle resources, it causes unpredictable latency due to its long-distance deployment. Moreover, it also inevitably increases the cost of bandwidth [5]. To compensate the shortcomings of cloud computing, the European Telecommunication Standard Institute (ETST) proposed mobile edge computing (MEC) in 2014 [6]. By sinking cloud computing and storage capabilities to the user side, it offers possibilities to simultaneously reduce data transmission latency and equipment energy consumption [7], [8], [9]. More precisely, a vehicle can optionally offload the computing tasks to some base station, and then carry out complex computation on the MEC server that is installed at the base station [10], [11], [12], [13].

B. Related Work

Recently, a large number of research results have been obtained in terms of offloading strategy. For example, a JCTO problem is proposed to realize simultaneous decision of content placement, content delivery and UAV trajectory, and a CBTL algorithm is proposed to solve the JCTO problem offline [14]. [15] proposes a SAGIN edge/cloud computing architecture that takes into account remote energy and computing constraints for offloading computationally intensive applications. Then the authors propose a joint resource allocation and task scheduling approach and uses a learning-based approach to optimize offloading strategies. [16] proposes algorithms designed by means of heuristic search, reformulation linearization technique and semi-definite relaxation. A knowledge-driven offloading framework for Internet of Vehicle uses the asynchronous advantage actor-critic (A3C) algorithm in [17]. [18] proposes a low complexity dynamic offloading algorithm using Lyapunov optimization. However, these strategies didn't consider energy consumption. In [19], [20], the authors consider the time dependence between the vehicle mobility and the task. Specifically, a heuristic algorithm is proposed by combining the task scheduling between the MEC servers and the RSU downlink energy consumption. Although the aforementioned

Manuscript received 11 July 2022; revised 11 December 2022, 1 April 2023, and 10 June 2023; accepted 20 June 2023. Date of publication 26 June 2023; date of current version 19 December 2023. This work was supported in part by the National Nature Science Foundation of Jilin Province under Grant 20230101055JC, and in part by the National Nature Science Foundation of China under Grant 62101210. The review of this article was coordinated by Prof. Zhu Han. *(Corresponding author: Cong Wang.)*

Yuliang Cong, Cong Wang, Wenxi Sun, Shuxian Sun, and Fengye Hu are with the College of Communication Engineering, Jilin University, Changchun 130012, China (e-mail: congyl@jlu.edu.cn; wangcong2020@jlu.edu.cn; wxsun19@mails.jlu.edu.cn; Sunsx20@mails.jlu.edu.cn; hufy@jlu.edu.cn).

Ke Xue is with the Changchun Institute of Optics, Fine Mechanics and Physics (CIOMP), Chinese Academy of Sciences, Changchun 130012, China (e-mail: xueke18@mails.jlu.edu.cn).

Digital Object Identifier 10.1109/TVT.2023.3289236

works properly handles the energy consumption, they do not address the problem of limited computing resources. In [21], [22], the authors optimize the latency on the basis of limited computing resources and ignore the energy consumption. Compared to [21], [22], more comprehensive methods are given in [23], [24], which balance the task offloading and resource allocation. Still, the energy consumption problem is not considered as an optimization target.

Most recent methods that simultaneously cope with latency and energy consumption are given in [25], [26]. However, the aforementioned researches mainly focus on how to offload the computing tasks to the MEC server, ignoring the situation of the cooperative processing of the MEC server and the cloud server. The introduction of cloud server can effectively solve the resource limitation problem of MEC server, thereby improving the user experience. Therefore, it is necessary to design a proper offloading scheme of cloud and MEC co-processing tasks to ensure user experience.

Meanwhile, each vehicle will get different computing tasks that require diverse computing resources due to different task sizes. Task offloading requires vehicles to communicate with MEC servers in the uplink wireless channel, leading to non-negligible energy consumption. Hence, the offloading strategy and resource allocation should be jointly optimized, and the two indicators of latency and energy consumption should be considered at the same time to obtain better network performance.

C. Main Contributions

In this article, our main contributions are as follows:

1) We propose to jointly apply cloud server and MEC server to the task offloading in vehicular networks, with appropriate consideration of resource allocation to achieve dual benefits of cloud computing and MEC. Each task can be offloaded through one of five strategies, and is evaluated via a loss function determined by its latency and energy consumption.
2) Our goal is to solve a MINLP problem whose objective function is the weighted sum of the task-specific loss functions (herein, the weights are calculated according to the nature of the tasks). To this end, we propose a new solution scheme. Specifically, we decompose the original optimization problem into two sub-problems: resource allocation problem and offloading strategy problem. By block coordinate descent technique, the two sub-problems are solved alternatively until convergence. Compared to directly solving the original MINLP with convex relaxation, the proposed scheme can find more optimal solutions.
3) Among the two sub-problems, the offloading strategy problem is a binary integer programming, which is NP-hard. For solving it, we relax discrete variables and propose an improved Gray Wolf algorithm based on elite strategy. Through numerical examples, we show that the improved Gray Wolf algorithm outperforms (i) its original version, and (ii) the classic/latest algorithms.

The rest of the article is organized as follows. The task-offloading model on cloud assisted MEC for vehicular networks is introduced in Section II. Section III mainly focuses on the solution to the aforementioned MINLP problem. Computer simulation results are presented in Section IV and concluding remarks are finally made in Section V.

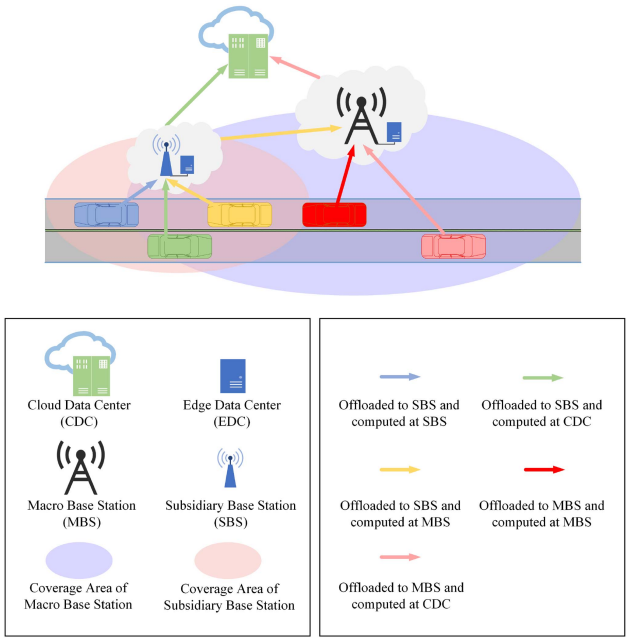

Fig. 1. Five offloading strategies in vehicular network.

II. Problem Formulation

A. System Model

We consider a heterogeneous vehicular network that has mobile edge computing functionality [27]. This network includes a macro base station (MBS) that is equipped with an edge macro data center (MDC). In addition, there are several subsidiary base stations (SBSs) within the MBS, each of which is equipped with an edge subsidiary data center (SDC). At the same time, there is also a cloud data center (CDC) that supports the network [28], [29]. As shown in Fig. 1, three offloading destinations are designed in the scenario: edge subsidiary data center deployed in subsidiary base stations, edge macro data center deployed in the macro base station and the cloud data center. For vehicle i, it can access macro base station or subsidiary base stations. There are four offloading schemes that directly reach the three offloading destinations: offloading directly to edge subsidiary / macro data center or to the cloud data center through the subsidiary / macro base stations. But considering that some tasks may need to be computed in the edge macro data center and the vehicles that are generating tasks are not within the communication range of the macro base station. We add an offloading scheme that is relayed by the subsidiary base stations and offloaded to the edge macro data center, namely the five schemes shown in the Fig. 1.

Within the coverage of the MBS, we let the number of vehicles be N, and index them by $i \in \{1, 2, \ldots, N\}$. The network bandwidth is divided into several sub-channels, and the bandwidth of each sub-channel is B. Vehicles are associated with the base station through OFDMA, and each vehicle occupies a channel.

Assuming that every vehicle generates a computing task to be executed, which is defined as $\mathbf{s_i} = (H_i, Z_i, T_i^{MAX})$. Here, H_i represents the data size of the task, Z_i represents the number of CPU cycles required to complete the task, and T_i^{MAX} represents the maximum latency that the task can tolerate [30]. Note that, we assume all computing tasks in the network are indivisible and the vehicle will not drive out of the coverage of the current server, so the case of task offload interruption is not discussed.Therefore, the base station handover will not be discussed.

The problem addressed in this article is the joint optimization of latency and energy consumption. Intuitively, when the network is optimized to a certain level, the optimization of latency and energy consumption oppose each other. More precisely, the pursuit of low energy consumption will result in high latency, and vice versa. In view of this fact, we design a loss function φ_i as follows:

$$\varphi_i = \lambda_i T_i + (1 - \lambda_i) E_i \tag{1}$$

where, T_i and E_i respectively represent the latency and energy consumption of task i. $\lambda_i \in [0, 1]$ is a weight that balances latency and energy consumption. Compared to the traditional fixed weight, λ_i is adaptive and dynamically varies according to the task itself. It becomes larger if Z_i increases and T_i^{MAX} decreases for the task. Detailed definition and normalization are given in (2) and (3), where $\xi \in (0, 1)$ is a factor affecting λ_i. The more sensitive tasks are to latency, the smaller the value of ξ.

$$\lambda_i^* = \xi \frac{Z_i}{\sum_{j=1}^{N} Z_j} + (1 - \xi) \frac{\sum_{j=1}^{N} T_j^{MAX}}{T_i^{MAX}} \tag{2}$$

$$\lambda_i = \frac{\lambda_i^* - \min\limits_i \{\lambda_i^*\}}{\max\limits_i \{\lambda_i^*\} - \min\limits_i \{\lambda_i^*\}} \tag{3}$$

For each vehicle i, there are two modes to choose from: local computation and offloading. In this article, we use $a_i \in \{0, 1\}$ to indicate the mode selection. Specifically, $a_i = 0$ means local computation; otherwise, offloading.

B. Local Computation of Task

When vehicle i selects local computation mode, its task latency consists in the processing latency expressed by (4), where f_i is the processor frequency assigned to the task. In the meantime, the energy consumption is expressed by (5). where κ is a factor determined by the processor structure. In this article, we choose $\kappa = 10^{-27}$. Given (4) and (5), the loss function associated with local computation mode reads as (6).

$$T_i^{local} = \frac{Z_i}{f_i} \tag{4}$$

$$E_i^{local} = \kappa f_i^2 Z_i \tag{5}$$

$$\varphi_i^{local}\left(T_i^{local}, E_i^{local}\right) = \lambda_i T_i^{local} + (1 - \lambda_i) E_i^{local} \tag{6}$$

C. Offloading of Task

When vehicle i selects offloading mode, its connection is denoted by $b_i \in \{0, 1\}$. Here, $b_i = 0$ indicates that the vehicle is connected to an SBS; otherwise, the vehicle is connected to the MBS. According to Shannon's theorem, the transmission capacities of SBS and MBS are:

$$R_i^s = B\log_2\left(1 + \frac{G_i^s p^s}{\sigma^2}\right) \tag{7}$$

$$R_i^m = B\log_2\left(1 + \frac{G_i^m p^m}{\sigma^2}\right) \tag{8}$$

where B is the channel bandwidth, p^s and p^m represent the transmit power of SBS and MBS, σ^2 means the noise power,G_i^s and G_i^m respectively represent the channel gain between the base station and vehicle i. In what follows, we elaborate five offloading strategies.

Note that it is unnecessary to offload tasks to a SBS when vehicles are connected to a MBS, as (i) it incurs greater latency, and (ii) the computing resource of the SBS is limited.

1) Vehicle Connected to SBS and Task Computed on SDC: When the task is computed on SDC, the latency can be divided into two parts: the latency of task transmission and processing.[1] Analogously, the energy consumption can be also divided into the fractions of task transmission and processing. As the edge side is powered by cables, this article does not consider the task-processing energy consumption of it [32]. Given the above discussion, the resulting $T_i^{sbs,s}$, $E_i^{sbs,s}$ and $\varphi_i^{sbs,s}$ are:

$$T_i^{sbs,s} = \frac{H_i}{R_i^s} + \frac{Z_i}{f_i} \tag{9}$$

$$E_i^{sbs,s} = p^s \frac{H_i}{R_i^s} \tag{10}$$

$$\varphi_i^{sbs,s}\left(T_i^{sbs,s}, E_i^{sbs,s}\right) = \lambda_i T_i^{sbs,s} + (1 - \lambda_i) E_i^{sbs,s} \tag{11}$$

2) Vehicle Connected to SBS and Task Computed on MDC: In this strategy, the vehicle is connected to SBS, but the task is offloaded to MDC for computation. Correspondingly, a transfer latency T_{sm} is incurred, and the resulting $T_i^{sbs,m}$, $E_i^{sbs,m}$ and $\varphi_i^{sbs,m}$ are:

$$T_i^{sbs,m} = \frac{H_i}{R_i^s} + \frac{Z_i}{f_i} + T_{sm} \tag{12}$$

$$E_i^{sbs,m} = p^s \frac{H_i}{R_i^s} \tag{13}$$

$$\varphi_i^{sbs,m}\left(T_i^{sbs,m}, E_i^{sbs,m}\right) = \lambda_i T_i^{sbs,m} + (1 - \lambda_i) E_i^{sbs,m} \tag{14}$$

3) Vehicle Connected to SBS and Task Computed on CDC: Similarly to the previous strategy, the transfer to CDC introduces some latency T_{sc}. Given this transfer latency, the resulting $T_i^{sbs,c}$, $E_i^{sbs,c}$ and $\varphi_i^{sbs,c}$ are:

$$T_i^{sbs,c} = \frac{H_i}{R_i^s} + \frac{Z_i}{f_i} + T_{sc} \tag{15}$$

$$E_i^{sbs,c} = p^s \frac{H_i}{R_i^s} \tag{16}$$

$$\varphi_i^{sbs,c}\left(T_i^{sbs,c}, E_i^{sbs,c}\right) = \lambda_i T_i^{sbs,c} + (1 - \lambda_i) E_i^{sbs,c} \tag{17}$$

[1]By [31], downloading latency is negligible relative to the offloading latency; hence, it is not considered.

4) Vehicle Connected to MBS and Task Computed on MDC: In this strategy, the vehicle is connected to the MBS, and the task is offloaded to MDC for computation. Herein, the resulting $T_i^{mbs,m}$, $E_i^{mbs,m}$ and $\varphi_i^{mbs,m}$ are:

$$T_i^{mbs,m} = \frac{H_i}{R_i^m} + \frac{Z_i}{f_i} \tag{18}$$

$$E_i^{mbs,m} = p^m \frac{H_i}{R_i^m} \tag{19}$$

$$\varphi_i^{mbs,m}\left(T_i^{mbs,m}, E_i^{mbs,m}\right) = \lambda_i T_i^{mbs,m} + (1-\lambda_i) E_i^{mbs,m} \tag{20}$$

5) Vehicle Connected to MBS and Task Computed on CDC: For this final strategy, the vehicle is connected to the MBS, but the task is computed on CDC. We denote the transfer latency by T_{mc} and obtain that:

$$T_i^{mbs,c} = \frac{H_i}{R_i^m} + \frac{Z_i}{f_i} + T_{mc} \tag{21}$$

$$E_i^{mbs,c} = p^m \frac{H_i}{R_i^m} \tag{22}$$

$$\varphi_i^{mbs,c}\left(T_i^{mbs,c}, E_i^{mbs,c}\right) = \lambda_i T_i^{mbs,c} + (1-\lambda_i) E_i^{mbs,c} \tag{23}$$

D. Main Problem

We denote the available computing resources by $\mathbf{F} = (F_{s\max}, F_{m\max}, F_{c\max})$, where $F_{s\max}$, $F_{m\max}$, $F_{c\max}$ represent the maximum computing resource at SDC, MDC and CDC. In addition, we use $\boldsymbol{\varepsilon}_i = (\varepsilon_i^s, \varepsilon_i^m, \varepsilon_i^c)$ to indicate offloading strategies of the task. Specifically, ε_i^s, ε_i^m, ε_i^c are binary variables, meaning whether the task is computed at SDC, MDS or CDC (1 is positive and 0 otherwise). Note that, as the task is assumed to be indivisible, its computation can be done in only one place, which leads to the following constraint:

$$\varepsilon_i^s + \varepsilon_i^m + \varepsilon_i^c = 1 \tag{24}$$

Combined with the previous content, the loss function of the vehicle can now be obtained:

$$\begin{aligned}
&\varphi_i\left(\varphi_i^{local}, \varphi_i^{mbs,m}, \varphi_i^{mbs,c}, \varphi_i^{sbs,s}, \varphi_i^{sbs,m}, \varphi_i^{sbs,c}, a_i, b_i, \boldsymbol{\varepsilon}_i\right) \\
&= (1-a_i)\varphi_i^{local} + a_i b_i\left(\varepsilon_i^m \varphi_i^{mbs,m} + \varepsilon_i^c \varphi_i^{mbs,c}\right) \\
&\quad + a_i(1-b_i)\left(\varepsilon_i^s \varphi_i^{sbs,s} + \varepsilon_i^m \varphi_i^{sbs,m} + \varepsilon_i^c \varphi_i^{sbs,c}\right)
\end{aligned} \tag{25}$$

By definitions of $\varphi_i^{local}, \varphi_i^{mbs,m}, \varphi_i^{mbs,c}, \varphi_i^{sbs,s}, \varphi_i^{sbs,m}, \varphi_i^{sbs,c}$ and constraint (24), we equivalently re-organize (25) as follows:

$$\begin{aligned}
&\varphi_i(H_i, Z_i, f_i, a_i, b_i, \varepsilon_i^s, \varepsilon_i^m, \varepsilon_i^c) \\
&= (1-a_i)\left(\lambda_i \frac{Z_i}{f_i} + (1-\lambda_i)\kappa f_i^2 Z_i\right) \\
&\quad + a_i(1-b_i)\lambda_i\left(\left(\frac{H_i}{R_i^s} + \frac{Z_i}{f_i}\right) + \varepsilon_i^m T_{sm} + \varepsilon_i^c T_{sc}\right) \\
&\quad + a_i(1-b_i)(1-\lambda_i) p^s \frac{H_i}{R_i^s} \\
&\quad + a_i b_i \lambda_i\left((\varepsilon_i^m + \varepsilon_i^c)\left(\frac{H_i}{R_i^m} + \frac{Z_i}{f_i}\right) + \varepsilon_i^c T_{mc}\right) \\
&\quad + a_i b_i (1-\lambda_i)(\varepsilon_i^m + \varepsilon_i^c) p^m \frac{H_i}{R_i^m}
\end{aligned} \tag{26}$$

Based on (26), we assume the knowledge of H_i, Z_i and formulate the optimization problem:

$$\min_{f_i, a_i, b_i, \varepsilon_i^s, \varepsilon_i^m, \varepsilon_i^c} \frac{1}{N}\sum_{i=1}^{N} \varphi_i(f_i, a_i, b_i, \varepsilon_i^s, \varepsilon_i^m, \varepsilon_i^c) \tag{27}$$

$$\text{s.t.}: a_i \in \{0,1\}, \forall i \in \mathcal{N} \tag{27a}$$

$$b_i \in \{0,1\}, \forall i \in \mathcal{N} \tag{27b}$$

$$\varepsilon_i^s \in \{0,1\}, \forall i \in \mathcal{N} \tag{27c}$$

$$\varepsilon_i^m \in \{0,1\}, \forall i \in \mathcal{N} \tag{27d}$$

$$\varepsilon_i^c \in \{0,1\}, \forall i \in \mathcal{N} \tag{27e}$$

$$\sum_{i=1}^{N} \varepsilon_i^s f_i \le F_{s,\max} \tag{27f}$$

$$\sum_{i=1}^{N} \varepsilon_i^m f_i \le F_{m,\max} \tag{27g}$$

$$\sum_{i=1}^{N} \varepsilon_i^c f_i \le F_{c,\max} \tag{27h}$$

$$\text{equation (24)}, \forall i \in \mathcal{N} \tag{27i}$$

Among the constraints: (27a) enforces that each task can be computed either locally or via offloading; (27b) represents the vehicle connection status; (27c)–(27e) indicate the offloading strategy; (27f)–(27h) are resource constraints.[2] As can be seen, (27) is a typical MINLP problem.

For simplicity of expression, we use vector $\mathbf{a} = (a_1, a_2, \ldots, a_N)$ to represent all offloading modes, and vector $\mathbf{b} = (b_1, b_2, \ldots, b_N)$ to represent all base station selections. In terms of offloading strategies, vectors $\boldsymbol{\varepsilon}^s$, $\boldsymbol{\varepsilon}^m$, and $\boldsymbol{\varepsilon}^c$ are used to indicate whether the tasks should be offloaded to the corresponding data centers for computation. With these notations, the task offloading decision variables can be collected as a single vector $\boldsymbol{\gamma} = (\mathbf{a}, \mathbf{b}, \boldsymbol{\varepsilon})$. Moreover, we use vector $\mathbf{f} = (f_1, f_2, \ldots, f_N)$ to represent the computing resources. By defining function (28), the problem (27) is compactly written as **P1**.

$$\begin{aligned}
\psi(\mathbf{f}, \boldsymbol{\gamma}) &= \psi(\mathbf{f}, \mathbf{a}, \mathbf{b}, \boldsymbol{\varepsilon}) \\
&= \frac{1}{N}\sum_{i=1}^{N} \varphi_i(f_i, a_i, b_i, \varepsilon_i^s, \varepsilon_i^m, \varepsilon_i^c)
\end{aligned} \tag{28}$$

$$[\mathbf{P1}] \quad \min_{\mathbf{f}, \boldsymbol{\gamma}} \psi(\mathbf{f}, \boldsymbol{\gamma}) \tag{29}$$

$$\text{s.t.}: \ (27\text{a}) \sim (27\text{i})$$

[2]The allocation of computing resources is generally measured by CPU frequency, which can be achieved through virtual machine technology. Noticeably, it is allowed to allocate different virtual machines to different vehicles, facilitating independent computation.

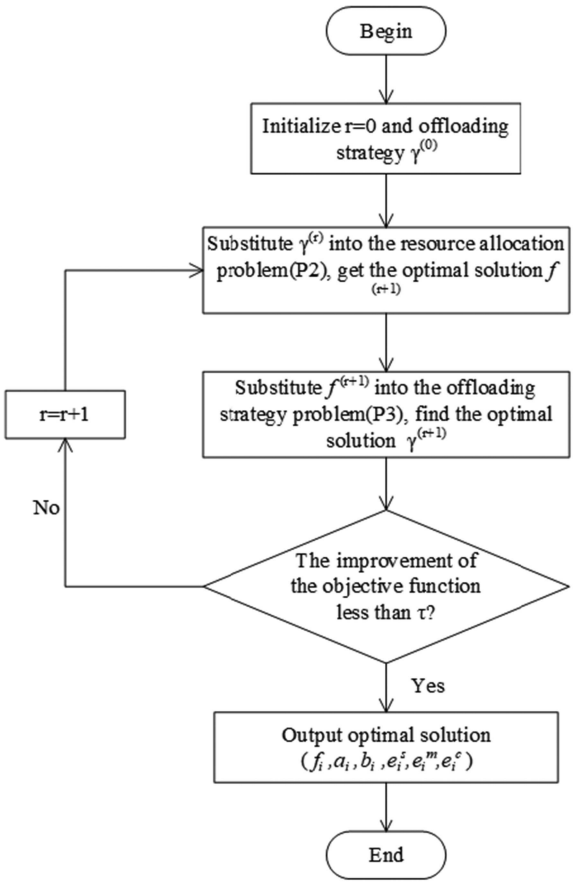

Fig. 2. Algorithm structure.

III. Scheme Design

P1 is a MINLP problem. To address it, we develop an iterative optimization scheme based on block coordinate descent technique that combines convex optimization and Gray Wolf algorithm (BCD-CONGW). More precisely, we first split **P1** into two sub-problems, as shown in Fig. 2. Among them, one optimizes the allocation of computing resources, assuming that offloading strategy is known; the other optimizes the offloading strategy, under the assumption that the computing resource allocation is known. These two problems are alternatively solved in each iteration, until convergence. In other words, when the improvement rate of the last two iterations is less than a preset threshold τ, the iteration is terminated, and the computing resource allocation and offloading strategies are finalized.

A. Resource Allocation

At the beginning, our model arbitrarily initializes the offloading strategy, and first copes with the resource allocation problem. Without loss of generality, let us denote the known offloading strategy of any iteration r by $\boldsymbol{\gamma}^{(r)} = \{\mathbf{a}^{(r)}, \mathbf{b}^{(r)}, \boldsymbol{\varepsilon}^{(r)}\}$ and plug it into (29), yielding the mathematical form of resource allocation problem **P2** as follows:

$$[\mathbf{P2}] \quad \min_{\mathbf{f}} \psi\left(\mathbf{f}, \boldsymbol{\gamma}^{(r)}\right) \tag{30}$$

$$\text{s.t.}: \sum_{i=1}^{N} \varepsilon_i^{s^{(r)}} f_i \le F_{s,\max} \tag{30a}$$

$$\sum_{i=1}^{N} \varepsilon_i^{m^{(r)}} f_i \le F_{m,\max} \tag{30b}$$

$$\sum_{i=1}^{N} \varepsilon_i^{c^{(r)}} f_i \le F_{c,\max} \tag{30c}$$

Proposition 1: **P2** is a convex problem.

Proof: According to optimization theory, when the Hessian matrix of a function is positive definite, the function is convex. By differentiating (30), we get

$$\frac{\partial \psi(\mathbf{f}, \boldsymbol{\gamma}^{(r)})}{\partial f_i} = \frac{1}{N}\left\{2\left(1-a_i^{(r)}\right)(1-\lambda_i)\kappa f_i Z_i - \left(1-a_i^{(r)} b_i^{(r)} \varepsilon_i^{s^{(r)}}\right)\lambda_i Z_i f_i^{-2}\right\} \tag{31}$$

Then, the following Hessian matrix is obtained:

$$\mathbf{H}_{N\times N} = \begin{pmatrix} \frac{\partial^2 \psi(\mathbf{f},\boldsymbol{\gamma}^{(r)})}{\partial f_1^2} & \frac{\partial^2 \psi(\mathbf{f},\boldsymbol{\gamma}^{(r)})}{\partial f_1 \partial f_2} & \cdots & \frac{\partial^2 \psi(\mathbf{f},\boldsymbol{\gamma}^{(r)})}{\partial f_1 \partial f_N} \\ \frac{\partial^2 \psi(\mathbf{f},\boldsymbol{\gamma}^{(r)})}{\partial f_2 \partial f_1} & \frac{\partial^2 \psi(\mathbf{f},\boldsymbol{\gamma}^{(r)})}{\partial f_2^2} & \cdots & \frac{\partial^2 \psi(\mathbf{f},\boldsymbol{\gamma}^{(r)})}{\partial f_2 \partial f_N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 \psi(\mathbf{f},\boldsymbol{\gamma}^{(r)})}{\partial f_N \partial f_1} & \frac{\partial^2 \psi(\mathbf{f},\boldsymbol{\gamma}^{(r)})}{\partial f_N \partial f_2} & \cdots & \frac{\partial^2 \psi(\mathbf{f},\boldsymbol{\gamma}^{(r)})}{\partial f_N^2} \end{pmatrix} \tag{32}$$

Since $f_i > 0, \forall i \in \mathcal{N}$, there is

$$\frac{\partial^2 \psi(\mathbf{f}, \boldsymbol{\gamma}^{(r)})}{\partial f_i^2} > 0, \qquad \frac{\partial^2 \psi(\mathbf{f}, \boldsymbol{\gamma}^{(r)})}{\partial f_i \partial f_j} = 0, \forall j \ne i \tag{33}$$

which implies that matrix $\mathbf{H}_{N\times N}$ is positive definite [33], and hence $\psi(\mathbf{f}, \boldsymbol{\gamma}^{(r)})$ is a convex function. Meanwhile, as the constraints in **P2** are linear, we have that **P2** is a convex optimization problem, and it can be efficiently solved. □

B. Task Offloading Strategy

By solving the resource allocation problem in iteration r, we obtain an optimal $\mathbf{f}^{(r+1)}$. Using this optimal value, the non-convex offloading strategy problem of the same iteration is formulated as follows:

$$[\mathbf{P3}] \quad \min_{\boldsymbol{\gamma}} \psi\left(\mathbf{f}^{(r+1)}, \boldsymbol{\gamma}\right) \tag{34}$$

$$\text{s.t.}: \sum_{i=1}^{N} \varepsilon_i^{s} f_i^{(r+1)} \le F_{s,\max} \tag{34a}$$

$$\sum_{i=1}^{N} \varepsilon_i^{m} f_i^{(r+1)} \le F_{m,\max} \tag{34b}$$

$$\sum_{i=1}^{N} \varepsilon_i^{c} f_i^{(r+1)} \le F_{c,\max}$$

$$(27a) \sim (27e), (27i) \tag{34c}$$

Then, we relax the discrete variables and convert **P3** into **P4**:

$$[\mathbf{P4}] \quad \min_{\mathbf{a},\mathbf{b},\boldsymbol{\epsilon}} \psi\left(\mathbf{f}^{(r+1)}, \mathbf{a}, \mathbf{b}, \boldsymbol{\epsilon}\right) \tag{35}$$

$$\text{s.t.}: 0 \le a_i \le 1, \forall i \in \mathcal{N} \tag{35a}$$

$$0 \le b_i \le 1, \forall i \in \mathcal{N} \tag{35b}$$

$$0 \le \epsilon_i^s \le 1, \forall i \in \mathcal{N} \tag{35c}$$

$$0 \le \epsilon_i^m \le 1, \forall i \in \mathcal{N} \tag{35d}$$

$$0 \le \epsilon_i^c \le 1, \forall i \in \mathcal{N} \tag{35e}$$

$$\sum_{i=1}^{N} \epsilon_i^s f_i^{(r+1)} \le F_{s,\max} \tag{35f}$$

$$\sum_{i=1}^{N} \epsilon_i^m f_i^{(r+1)} \le F_{m,\max} \tag{35g}$$

$$\sum_{i=1}^{N} \epsilon_i^c f_i^{(r+1)} \le F_{c,\max} \tag{35h}$$

(27i)

P4 minimizes a non-convex objective function subject to linear constraints. Currently, there exists a large number of intelligent algorithms to solve this problem. Through comparative research, we adopt an offloading-strategy-making scheme based on the Gray Wolf algorithm [34]. As the name suggests, Gray Wolf algorithm originates from the hierarchy and hunting behaviour of wolves. Under normal circumstances, the algorithm aims at continuous optimization problems, and there are few applications involving 0-1 programming problems. In this article, the discrete variables are relaxed and brought into the Gray Wolf algorithm model for solution. Considering that standard Gray Wolf algorithm has certain limitations and in this article proposes an improved variant of it based on elite strategy, which can effectively reduce the number of iterations and avoid falling into the local optimal solution to a certain extents easy to fall into the local optimal solution. The algorithmic steps are listed in Algorithm 1, and the details are provided in Appendix A and B.

C. Joint Resource Allocation and Task Offloading Strategy

In summary, the optimal resource allocation and offloading strategy can be obtained by alternatively fixing one variable and optimizing another variable. Specifically, the MINLP problem **P1** is divided into two sub-problems, namely the resource allocation problem **P2** and the offloading strategy problem **P3**. We have mathematically proved that **P2** is a convex optimization problem, and have also converted **P3** into a relaxed problem **P4**. According to [35], this kind of problem will eventually converge in the Block Coordinate Descent (BCD) technique. To this end, this section gives an iterative optimization scheme based on block coordinate descent technique that combines convex optimization and Gray Wolf algorithm (BCD-CONGW), for solving problem **P1**. The details are provided in Algorithm 2.

Algorithm 1: Grey Wolf Algorithm with Elite Strategy.

Input: Number of wolves M, number of iterations K.
Output: Offloading strategy $\boldsymbol{\gamma}$
1: **Initialization:** Randomly generate an initial wolf pack of size M, and relax decision variables;
2: Compute the fitness value ((35)) of each gray wolf in the initial wolf pack, and determine the division levels $\alpha, \beta, \delta, \omega$;
3: **for** $k = 1{:}K$ **do**
4: Surround, hunt, and attack prey (i.e., equations (36)–(41) in Appendix A).
5: Compute the fitness value of all gray wolves.
6: According to the fitness value, the wolves are re-divided into four levels.
7: Inferior wolves in the pack learn from elite wolves and update their positions (i.e., equations (42)-(43) in Appendix B).
8: $k = k + 1$.
9: **end for**

Algorithm 2: BCD-CONGW Scheme.

Input: Number of vehicles N, number of wolves M, number of iterations K, threshold τ.
Output: Resource allocation $\mathbf{f}^*$, offloading strategy $\boldsymbol{\gamma}^*$.
1: **Initialization:** Initialize offloading strategy $\boldsymbol{\gamma}^{(0)}$, and set the iteration number $r = 0$.
2: **while** $\tau < \varepsilon$ **do**
3: Substitute $\boldsymbol{\gamma}^{(r)}$ into **P2**, and obtain the optimal solution $\mathbf{f}^{(r+1)}$ through convex optimization algorithm.
4: Substitute $\mathbf{f}^{(r+1)}$ into **P4**, and solve the problem by Algorithm 1 to obtain offloading strategy $\boldsymbol{\gamma}^{(r+1)}$.
5: $r = r + 1$.
6: Compute the growth rate ε.
7: **end while**

IV. Numerical Examples

In order to evaluate the proposed algorithm, we carry out simulations using MATLAB. Specifically, we used the distribution of 5G base stations near Shanghai Disneyland in China, as shown in Fig. 3. There is one MBS covers the whole area, four SBS's distributed on a 1500-meter road. The communication ranges of MBS and SBS are 1.4 km and 0.5 km. The bandwidth of each channel is B=5MHz [36]. MBS transmit power $p^m = 46$ dBm, SBS transmit power $p^s = 30$ dBm, Gaussian white noise $\sigma^2 = -147$ dBm, interference between MBS and SBS $I = 100\sigma^2$, channel attenuation model $u_i = 127 + 30\log(d_i)$ (d_i is the distance between the base station and vehicle i), channel gain between the base station and vehicle i is $G_i = 10^{-u_i/10}$.

Moreover, the computation frequency on each vehicle is 0.5 GHz. Based on existing research [37], [38], the computing resources of SDC, MDC, and CDC are $F_{s\max} = 20$ GHz, $F_{m\max} = 100$ GHz, and $F_{c\max} = 300$ GHz. The latencies

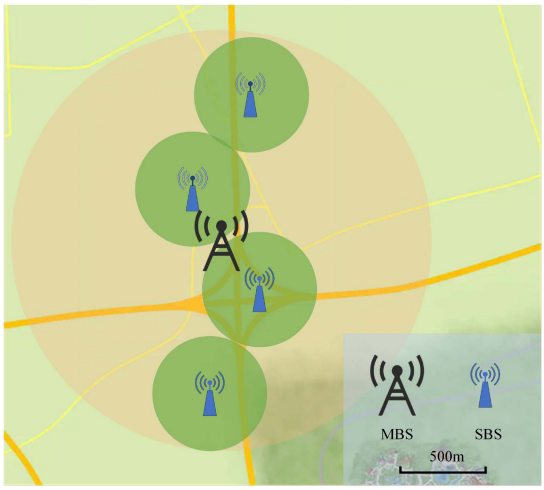

Fig. 3. BSs deployment near Shanghai Disneyland in China.

TABLE I
SIMULATION PARAMETERS

Parameter	Value
The communicaiton range of MBS(km)	1.4
The communication range of SBS(km)	0.5
The transmit power of MBS(dBm)	46
The transmit power of SBS(dBm)	30
Gaussian white noise(dBm)	-147
The computation frequency of vehicle(GHz)	0.5
The computing resource of SDC(GHz)	20
The computing resource of MDC(GHz)	100
The computing resource of CDC(GHz)	300
The number of gray wolves	30
The threshold of BCD-CONGW	0.01

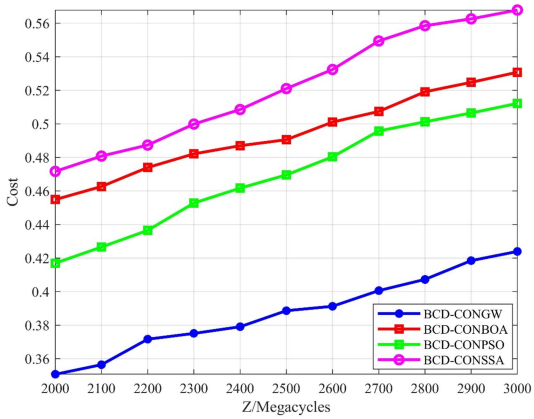

Fig. 4. Relationship between required CPU cycles and average cost.

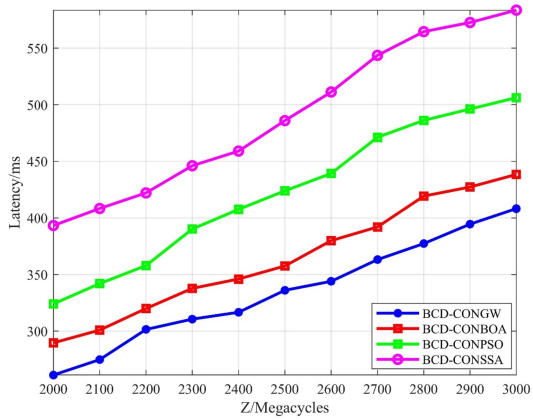

Fig. 5. Relationship between required CPU cycles and average latency.

for transferring tasks between data centers are $T_{sm} = 2$ ms, $T_{sc} = 5$ ms, and $T_{mc} = 1.5$ ms. The number of iterations for Gray Wolf algorithm is 30, and the number of gray wolves is 30. The threshold of BCD-CONGW scheme is $\tau = 0.01$.

We compare BCD-CONGW with three schemes mentioned below. Note that the maximum number for iterations of the three schemes is 150, and the number of individuals is 30.

1) *BCD-CONBOA:* The algorithm structure is the same as BCD-CONGW, but use butterfly of algorithm (BOA) to optimize problem P4 [39].
2) *BCD-CONPSO:* The algorithm structure is the same as BCD-CONGW, but use Particle Swarm optimization (PSO) to optimize problem P4 [40].
3) *BCD-CONSSA:* The algorithm structure is the same as BCD-CONGW, but use Squirell Search Algorithm (SSA) to optimize problem P4 [41].

A. Analysis of CPU Cycles Required by the Tasks

In what follows, we randomly distribute 150 vehicles, each of which generates a task. The data size per task fulfils a uniform distribution between 0.7 MB and 0.8 MB. The maximum tolerable latency for each individual task is 0.7 s. Besides, the CPU transfer required for completion of the task is between 2000 megacycles and 3000 megacycles.

After 50 simulations, average results are obtained. In Fig. 4, the relationship between the task cost and the number of required CPU cycles is analyzed. As the number of required CPU cycles increases, the cost rises up. By observation, BCD-CONSOA scheme has the worst performance. In addition, the proposed BCD-CONGW scheme is better than the BCD-CONSSA, BCD-CONBOA and BCD-CONPSO schemes, reducing the cost by 25.4%, 20.8%, 17.2%, under 2500 megacycles, respectively.

With respect to the latency, we show results in Fig. 5. Therein, the larger the task size, the greater the latency. As can be seen, the latency of three offloading schemes is smaller than that of the BCD-CONSSA, and the gap widens as the number of required CPU cycles increases. Moreover, for all CPU cycles in Fig. 5, BCD-CONGW scheme outperforms the others.

In terms of energy consumption (as shown in Fig. 6), the performance of four offloading schemes remains stable against the required CPU cycles. Furthermore, the proposed BCD-CONGW scheme generates the lowest energy consumption among all offloading schemes.

As the number of CPU cycles required by tasks increases, task offloading decision is more inclined to performing tasks on data center with richer computing resources. As can be seen from the simulation results, the average cost and delay of tasks increase and the energy consumption remains unchanged, BCD-CONGW scheme has the best effect all the time. This is because

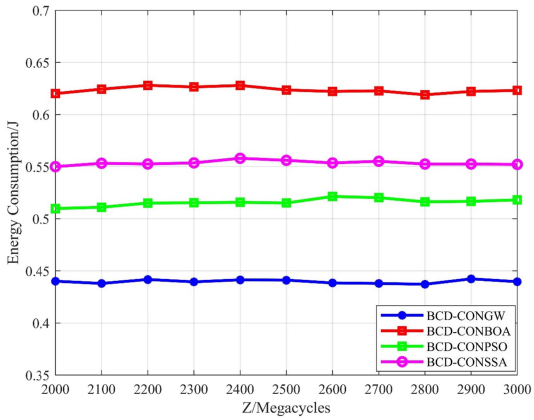

Fig. 6. Relationship between required CPU cycles and average energy consumption.

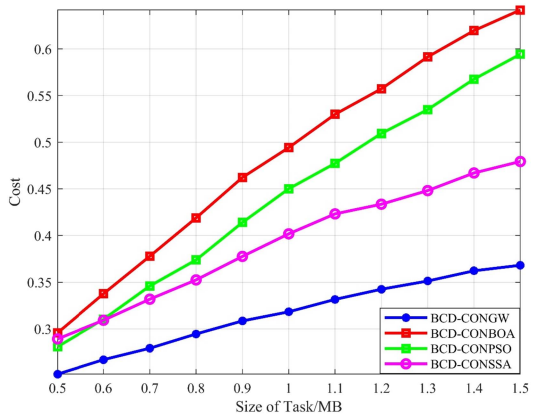

Fig. 7. Relationship between task data size and average cost.

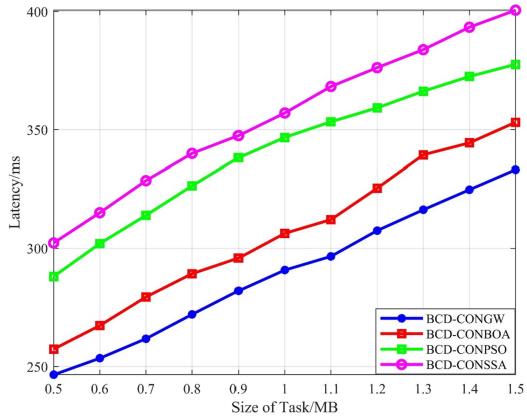

Fig. 8. Relationship between task data size and average latency.

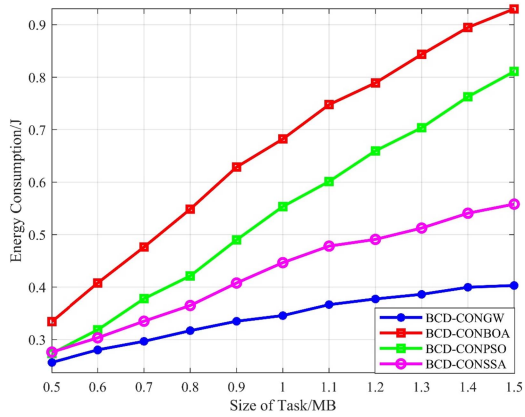

Fig. 9. Relationship between task data size and average energy consumption.

by adding the elite strategy, the iterative times of the Grey Wolf algorithm are optimized and it is less likely to fall into the local optimal. Although the cost and delay increase with the number of CPU cycles required by tasks, the BCD-CONGW scheme can still obtain better solutions than other algorithms.

B. Analysis of Data Size Per Task

Now, we assume that the number of CPU cycles required to complete a task is uniformly distributed between 700 megacycles and 800 megacycles. In addition, the amount of data per task is 0.5 MB and 1.5 MB. After 50 simulations, average results are obtained.

In Fig. 7, the impact of data size on cost is analyzed. As the amount of data increases, the cost of offloading schemes gradually increases. Notably, the proposed BCD-CONGW scheme has a better optimization effect than its counterparts.

In Figs. 8 and 9, it can be seen that our BCD-CONGW scheme has a good optimization effect in terms of latency and energy consumption.

As the amount of task data increases, the transmission delay and energy consumption of task data increase. The offloading decision is more inclined to transferring the task to the nearest data center, and the cost increase is inevitable. As can be seen from the figure, BCD-CONGW scheme has always been superior to other algorithms in terms of latency, energy consumption and cost. With the increase of task data amount, the advantages of BCD-CONGW scheme continue to expand in terms of energy consumption and cost, because BCD-CONGW scheme has higher efficiency and faster convergence speed. It can better mitigate the rise in costs and energy consumption.

C. Analysis of the Number of Tasks

In the following text, the amount of data per task is between 1 MB. Meanwhile, the number of vehicles is between 50 and 90. After 50 simulations, average results are obtained.

From Figs. 10–12, we demonstrate performance with respect to the number of tasks. As the number of tasks increases, the cost, latency, and energy consumption gradually increase. Obviously, the proposed BCD-CONGW scheme outperforms the other schemes. Indeed, when the number of tasks increases, BCD-CONGW scheme can (i) effectively allocate computing resources, (ii) regulate task offloading, and (iii) reduce the average cost, latency, and energy consumption. Intuitively, this is because the Gray Wolf algorithm based on elite strategy has faster convergence speed, which enables BCD-CONGW scheme to obtain better solutions in computing resource allocation and

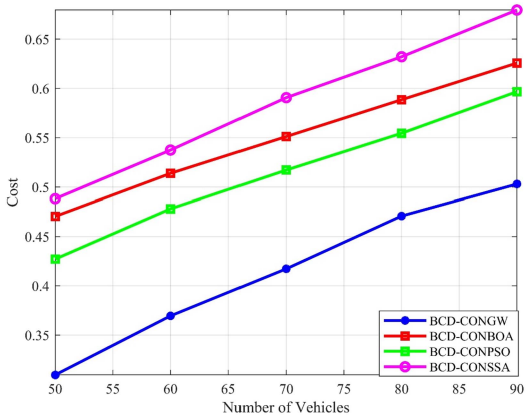

Fig. 10. Relationship between the number of tasks and average cost.

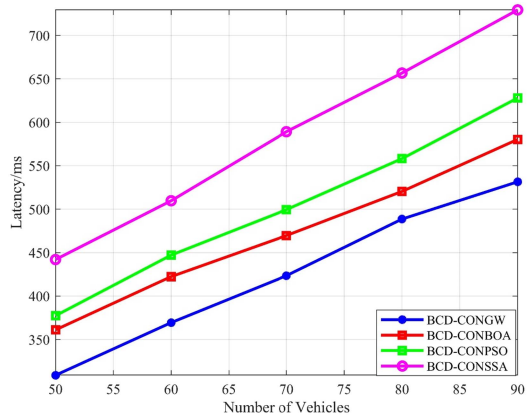

Fig. 11. Relationship between the number of tasks and average latency.

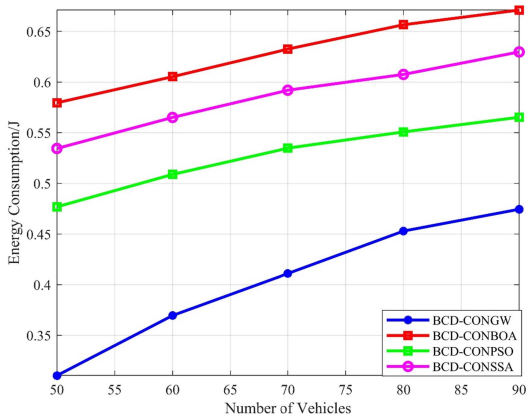

Fig. 12. Relationship between the number of tasks and average energy consumption.

offloading strategy within a limited number of cycles, thus reducing the cost, delay and energy consumption.

D. Further Evaluation

First, we validate the employment of our block coordinate descent scheme. For this purpose, we relax the discrete variables and directly solve the relaxed optimization problem by our improved GWO based on elite strategy. From Figs. 13 and 14,

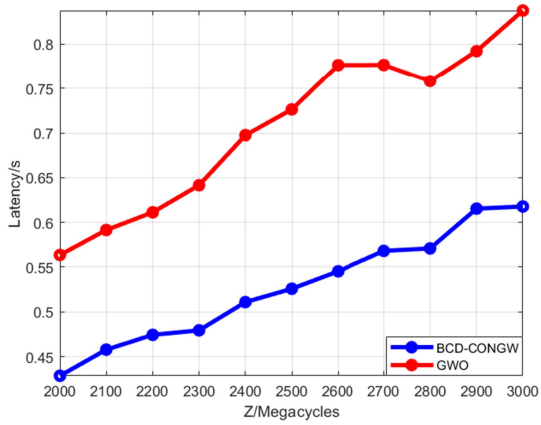

Fig. 13. Relationship between required CPU cycles and average latency.

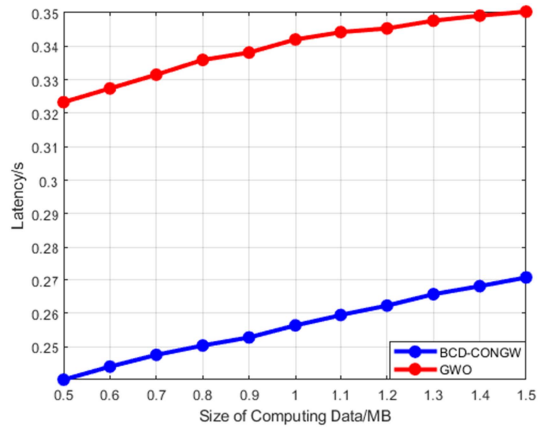

Fig. 14. Relationship between task data size and average energy consumption.

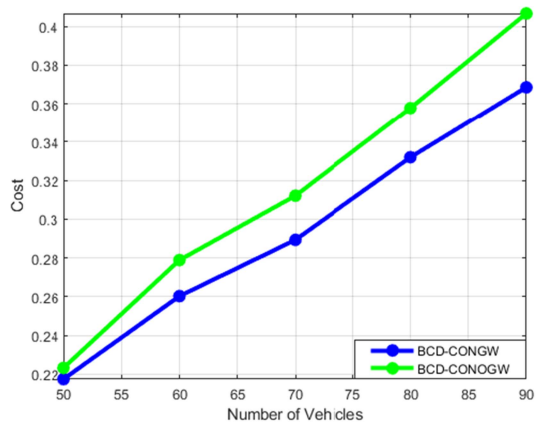

Fig. 15. Relationship between the number of tasks and average cost.

it can be seen that BCD-CONGW scheme is much better than direct application of our improved GWO in terms of latency.

Next, we aim to demonstrate that the improved GWO based on elite strategy outperforms the original GWO [42]. To this end, we replace the improved GWO in BCD-CONGW scheme by the original GWO, and refer to it as the BCD-CONOGW scheme. From the simulation results in Fig. 15, it can be seen

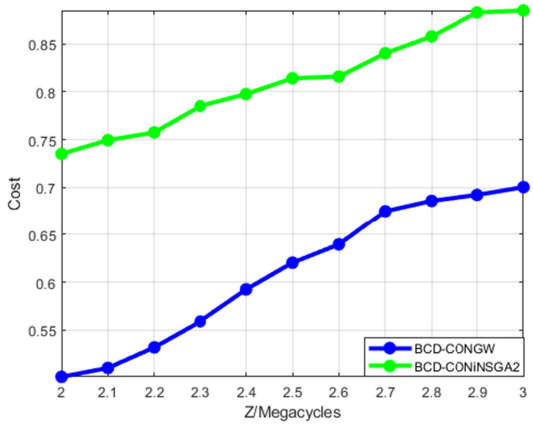

Fig. 16. Relationship between required CPU cycles and average cost.

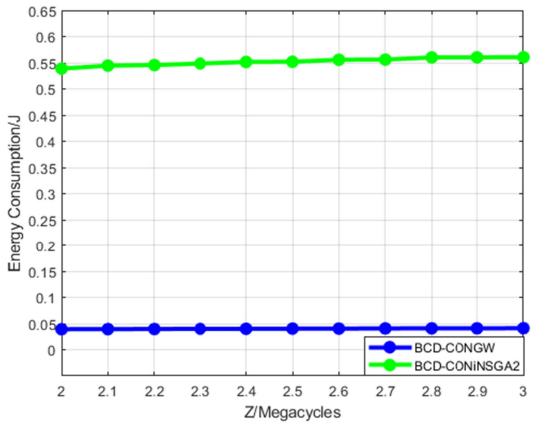

Fig. 17. Relationship between required CPU cycles and average energy consumption.

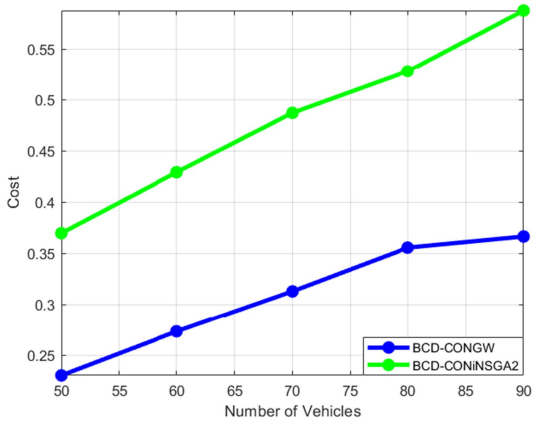

Fig. 18. Relationship between the number of tasks and average cost.

that BCD-CONGW scheme has obvious advantages over BCD-CONOGW scheme, with a performance gap rising as the number of vehicles increases. This is because the elite strategy enables the dominant individuals in each iteration to be retained, thus improving the convergence accuracy and solution efficiency of the original GWO.

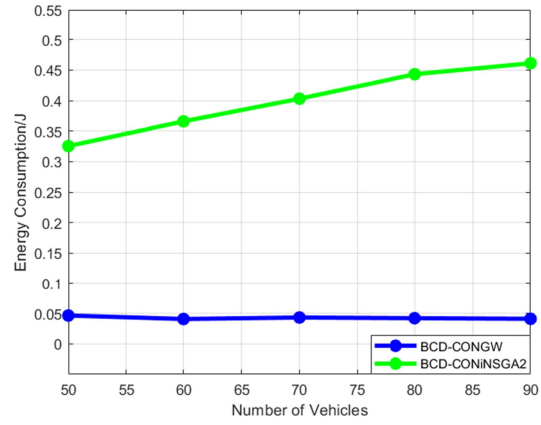

Fig. 19. Relationship between the number of tasks and average energy consumption.

Finally, we take a latest algorithm, iNSGA2 [43], to compare against our improved GWO based on elite strategy. In detail, we replace the improved GWO in BCD-CONGW scheme by iNSGA2, and refer to it as the BCD-CONiNSGA2 scheme. From Figs. 16–19, it can be seen that BCD-CONGW has a great advantage in terms of cost and energy consumption. Indeed, for our GWO based on elite strategy, both the search range and the convergence accuracy have been significantly improved. At the same time, it is less prone to falling into the local optimal solution.

V. Conclusion

In this article, an MEC system for the Internet of Vehicles has been constructed. In this system, we have proposed a BCD-CONGW offloading scheme. The scheme has strong optimization capabilities in terms of cost, latency and energy consumption, and can effectively deal with large task data and high complexity computing tasks by alternatively solving two optimization problems.

Simulation results show that, with the increase of the number of CPU cycles, task data size and the number of vehicles, the proposed scheme can effectively allocate the computing resources and adjust offloading strategy of tasks, thus, the average cost, latency and energy consumption of the system are reduced.

Appendix A
Gray Wolf Algorithm

Main steps of Gray Wolf algorithm are shown below.

a) Surround

$$\mathbf{x}(k+1) = \mathbf{x}_p(k) - \mathbf{p} \cdot \mathbf{d} \tag{36}$$

$$\mathbf{d} = |\mathbf{q} \cdot \mathbf{x}_p(k) - \mathbf{x}(k)| \tag{37}$$

Here, k is the iteration index in this algorithm and "$\cdot$" means entry-wise multiplication. $\mathbf{x}(k)$ is the position of an arbitrary gray wolf; $\mathbf{x}_p(k)$ is the position of the prey; $\mathbf{d}$ is the distance between the wolf and the prey. $\mathbf{p} = 2\varrho\mathbf{r}_1 - \varrho\mathbf{1}$ and $\mathbf{q} = 2\mathbf{r}_2$, with $\mathbf{r}_1$ and $\mathbf{r}_2$ being two random vectors whose entries all reside in [0,1]. Convergence factor ϱ is a key parameter to balance the

search and development capabilities of the algorithm. Its value decreases linearly from 2 to 0 as k increases. The calculation formula is (41).

b) Hunt

$$\mathbf{x}(k+1) = \frac{\mathbf{x}_1 + \mathbf{x}_2 + \mathbf{x}_3}{3} \tag{38}$$

$$\begin{cases} \mathbf{x}_1 = \mathbf{x}_\alpha(k) - \mathbf{p}_1 \cdot \mathbf{d}_\alpha \\ \mathbf{x}_2 = \mathbf{x}_\beta(k) - \mathbf{p}_2 \cdot \mathbf{d}_\beta \\ \mathbf{x}_3 = \mathbf{x}_\delta(k) - \mathbf{p}_3 \cdot \mathbf{d}_\delta \end{cases} \tag{39}$$

$$\begin{cases} \mathbf{d}_\alpha = |\mathbf{q}_1 \cdot \mathbf{x}_\alpha(k) - \mathbf{x}(k)| \\ \mathbf{d}_\beta = |\mathbf{q}_2 \cdot \mathbf{x}_\beta(k) - \mathbf{x}(k)| \\ \mathbf{d}_\delta = |\mathbf{q}_3 \cdot \mathbf{x}_\delta(k) - \mathbf{x}(k)| \end{cases} \tag{40}$$

The wolves are divided into four levels according to their fitness values. The top three levels are α, β, δ, each containing one wolf. All remaining wolves belong to the fourth level ω. Here, the positions of wolves in α, β, δ levels are represented by $\mathbf{x}_\alpha, \mathbf{x}_\beta, \mathbf{x}_\delta$. Meanwhile, $\mathbf{d}_\alpha$, $\mathbf{d}_\beta$, $\mathbf{d}_\delta$ are the distances between an arbitrary wolf and the top-three wolves. We use (38)–(40) to update the positions of all gray wolves. Therein, $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$ and $\mathbf{q}_1$, $\mathbf{q}_2$, $\mathbf{q}_3$ are the same as aforementioned.

c) Attack: In (41), K represents the maximum number of iterations. When ϱ decreases from 2 to 0, the entries in $\mathbf{p}$ also change within the same interval, indicating the transit from global search to local search.

$$\varrho = 2 - 2k/K \tag{41}$$

Appendix B
Improved Gray Wolf Algorithm Based on Elite Strategy

In order to avoid local optimal solution and premature convergence as much as possible, this article proposes an improved Gray Wolf algorithm based on the elite strategy. This method divides wolves into elite wolf and inferior wolf groups according to the fitness value. Wolves in the inferior group need to learn from their counterparts in the elite group.

Assume that the gray wolves are collected by $\mathcal{G} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\}$. First, let us calculate the average fitness value of $\mathcal{G}$ as Avg. Then, construct $\mathcal{G}_{inf} = \{\mathbf{x} | f(\mathbf{x}) > \text{Avg}, \mathbf{x} \in \mathcal{G}\}$ as the inferior group and $\mathcal{G}_{elite} = \{\mathbf{x} | f(\mathbf{x}) \leq \text{Avg}, \mathbf{x} \in \mathcal{G}\}$ as the elite group. Each gray wolf in $\mathcal{G}_{inf}$ should learn from a random elite wolf $\mathbf{x}_{elite}$ in $\mathcal{G}_{elite}$. The probability of elite-wolf selection is given in (42). Moreover, the inferior gray wolf learns from the elite wolf through (43), and becomes a new wolf $\mathbf{x}_{new}$. Here, $\mathbf{x}_{\max}, \mathbf{x}_{\min}$ are the upper and lower bounds of the optimization spaces, and $F \in (0, 2)$ is a variation factor.

$$P(\mathbf{x}_{elite}) = \frac{\text{Avg} - f(\mathbf{x}_{elite})}{\sum_{\mathbf{x} \in \mathcal{G}_{elite}} (\text{Avg} - f(\mathbf{x}))} \tag{42}$$

$$\mathbf{x}_{new} = \mathbf{x}_{elite} + F(\mathbf{x}_{\max} - \mathbf{x}_{\min}) \tag{43}$$

In the algorithm, let the total number of gray wolf individuals be M, the individual dimension be n. Suppose: t_0 is the time for setting the initial parameters, t_1 is the time for initializing each dimension of gray wolf individuals, $f(n)$ is the time for calculating the individual fitness value per gray wolf individual [44], [45], and t_2 is the time for the positions of wolves in α, β, δ levels. Then the time complexity of the preparation phase is:

$$T_1 = \mathcal{O}(t_0 + M \cdot (n \cdot t_1 + f(n)) + t_2) = \mathcal{O}(n + f(n)) \tag{44}$$

After entering the iteration, the max number of iterations is K. Suppose that the time for calculating coefficient vectors $\mathbf{p}$ and $\mathbf{q}$ is t_3. The time for calculating average fitness is t_4, and the time for comparing individual fitness values is t_5. If l represents the number of individuals whose fitness values are greater than the average fitness size Avg, t_6 is the time for calculating the distance between each individual in the population and the α, β, δ wolves by formula (40), t_7 is the time of position update by formula (39), t_8 is the time of average position calculation by formula (38), t_9 is the time to process the boundary of each dimension of gray wolf individuals, t_{10} is the time to update the wolf positions of α, β, δ, then the time complexity of the iteration phase is:

$$T_2 = \mathcal{O}(Mt_3 + t_4 + t_5 + l(3(t_6 + t_7) + t_8 + nt_9 + f(n) + t_{10})) = \mathcal{O}(n + f(n)) \tag{45}$$

To sum up, the total time complexity of the improved GWO algorithm is:

$$T = T_1 + KT_2 = \mathcal{O}(n + f(n)) \tag{46}$$

References

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

[2] H. Li, H. Xu, C. Zhou, X. Lü, and Z. Han, "Joint optimization strategy of computation offloading and resource allocation in multi-access edge computing environment," *IEEE Trans. Veh. Technol.*, vol. 69, no. 9, pp. 10214–10226, Sep. 2020.

[3] S. Popli, R. K. Jha, and S. Jain, "A survey on energy efficient narrowband-Internet of Things (NBIoT): Architecture, application and challenges," *IEEE Access*, vol. 7, pp. 16739–16776, 2019.

[4] C. Su, F. Ye, T. Liu, Y. Tian, and Z. Han, "Computation offloading in hierarchical multi-access edge computing based on contract theory and Bayesian matching game," *IEEE Trans. Veh. Technol.*, vol. 69, no. 11, pp. 13686–13701, Nov. 2020.

[5] W. Zhuang et al., "SDN/NFV-Empowered future IoV with enhanced communication, computing, and caching," *Proc. IEEE*, vol. 108, no. 2, pp. 274–291, Feb. 2020.

[6] A. Ahmed and E. Ahmed, "A survey on mobile edge computing," in *Proc. 10th Int. Conf. Intell. Syst. Control*, 2016, pp. 1–8.

[7] M. Azizian, S. Cherkaoui, and A. Hafid, "An optimized flow allocation in vehicular cloud," *IEEE Access*, vol. 4, pp. 6766–6779, 2016.

[8] J. Chen, Z. Chang, X. Guo, R. Li, Z. Han, and T. Hämäläinen, "Resource allocation and computation offloading for multi-access edge computing with fronthaul and backhaul constraints," *IEEE Trans. Veh. Technol.*, vol. 70, no. 8, pp. 8037–8049, Aug. 2021.

[9] Y. Yu, X. Bu, K. Yang, H. Yang, X. Gao, and Z. Han, "UAV-aided low latency multi-access edge computing," *IEEE Trans. Veh. Technol.*, vol. 70, no. 5, pp. 4955–4967, May 2021.

[10] T. K. Rodrigues, J. Liu, and N. Kato, "Offloading decision for mobile multi-access edge computing in a multi-tiered 6G network," *IEEE Trans. Emerg. Topics Comput.*, vol. 10, no. 3, pp. 1414–1427, Jul./Sep. 2022.

[11] T. Rodrigues Koketsu, J. Liu and N. Kato, "Application of cybertwin for offloading in mobile multi-access edge computing for 6G networks," *IEEE Internet Things J.*, vol. 8, no. 22, pp. 16231–16242, Nov. 2021.

[12] F. Tang, Y. Kawamoto, N. Kato, and J. Liu, "Future intelligent and secure vehicular network toward 6G: Machine-learning approaches," *Proc. IEEE*, vol. 108, no. 2, pp. 292–307, Feb. 2020.

[13] H. Guo, J. Liu, and J. Zhang, "Computation offloading for multi-access mobile edge computing in ultra-dense networks," *IEEE Commun. Mag.*, vol. 56, no. 8, pp. 14–19, Aug. 2018.
[14] H. Wu, F. Lyu, C. Zhou, J. Chen, L. Wang, and X. Shen, "Optimal UAV caching and trajectory in aerial-assisted vehicular networks: A learning-based approach," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 12, pp. 2783–2797, Dec. 2020.
[15] N. Cheng et al., "Space/Aerial-assisted computing offloading for IoT applications: A learning-based approach," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 5, pp. 1117–1129, May 2020.
[16] J. Liu and Q. Zhang, "Offloading schemes in mobile edge computing for ultra-reliable low latency communications," *IEEE Access*, vol. 6, pp. 12825–12837, 2018.
[17] Q. Qi et al., "Knowledge-driven service offloading decision for vehicular edge computing: A deep reinforcement learning approach," *IEEE Trans. Veh. Technol.*, vol. 68, no. 5, pp. 4192–4203, May 2019.
[18] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3590–3605, Dec. 2016.
[19] B. Lin et al., "Computation offloading strategy based on deep reinforcement learning for connected and autonomous vehicle in vehicular edge computing," *J. Cloud Comput.*, vol. 10, no. 1, 2021, Art. no. 33.
[20] Z. Ning, J. Huang, X. Wang, J. J. P. C. Rodrigues, and L. Guo, "Mobile edge computing-enabled internet of vehicles: Toward energy-efficient scheduling," *IEEE Netw.*, vol. 33, no. 5, pp. 198–205, Sep./Oct. 2019.
[21] C. Shu, Z. Zhao, Y. Han, G. Min, and H. Duan, "Multi-user offloading for edge computing networks: A dependency-aware and latency-optimal approach," *IEEE Internet Things J.*, vol. 7, no. 3, pp. 1678–1689, Mar. 2020.
[22] U. Saleem, Y. Liu, S. Jangsher, X. Tao, and Y. Li, "Latency minimization for D2D-enabled partial computation offloading in mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 69, no. 99, pp. 4472–4486, Apr. 2020.
[23] C. Xu, G. Zheng, and X. Zhao, "Energy-minimization task offloading and resource allocation for mobile edge computing in NOMA heterogeneous networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 12, pp. 16001–16016, Dec. 2020.
[24] H. Li, H. Xu, C. Zhou, L. Xing, and Z. Han, "Joint optimization strategy of computation offloading and resource allocation in multi-access edge computing environment," *IEEE Trans. Veh. Technol.*, vol. 69, no. 12, pp. 16001–16016, Dec. 2020.
[25] X. Gu and G. Zhang,, "Energy-efficient computation offloading for vehicular edge computing networks," *Comput. Commun.*, vol. 166, pp. 244–253, 2021.
[26] J. Zhou, X. Zhang, W. Wang, and Y. Zhang, "Energy-efficient collaborative task offloading in D2D-assisted mobile edge computing networks," in *Proc. IEEE Wireless Commun. Netw. Conf.*, 2019, pp. 1–6.
[27] L.-H. Yen, J.-C. Hu, Y.-D. Lin, and B. Kar, "Decentralized configuration protocols for low-cost offloading from multiple edges to multiple vehicular fogs," *IEEE Trans. Veh. Technol.*, vol. 70, no. 1, pp. 872–885, Jan. 2021.
[28] M. Saimler and S. C., "Ergen, Uplink/downlink decoupled energy efficient user association in heterogeneous cloud radio access networks," *Ad Hoc Netw.*, vol. 97, no. 2, 2020, Art. no. 102016.
[29] G. Wang and F. Xu, "Regional intelligent resource allocation in mobile edge computing based vehicular network," *IEEE Access*, vol. 8, pp. 7173–7182, 2020.
[30] T. Nguyen, V. D. Nguyen, V. N. Pham, N. Luan, and E. N. Huh, "Modeling data redundancy and cost-aware task allocation in MEC enabled internet-of-vehicles applications," *IEEE Internet Things J.*, vol. 8, no. 3, pp. 1687–1701, Feb. 2021.
[31] X. Q. Pham, T. D. Nguyen, V. D. Nguyen, and E. N. Huh, "Joint node selection and resource allocation for task offloading in scalable vehicle-assisted multi-access edge computing," *Symmetry*, vol. 11, no. 1, 2019, Art. no. 58.
[32] L. Yang, H. Zhang, M. Li, J. Guo, and H. Ji, "Mobile edge computing empowered energy efficient task offloading in 5G," *IEEE Trans. Veh. Technol.*, vol. 67, no. 7, pp. 6398–6409, Jul. 2018.
[33] S. P. Boyd and L. Vandenberghe, *Convex Optimization.*Cambridge, U.K.: Cambridge Univ. Press, 2004.
[34] A. Sm, B. Smm, and A. Al, "Grey wolf optimizer," *Adv. Eng. Softw.*, vol. 69, pp. 46–61, 2014.
[35] Y. Xu and W. Yin, "A block coordinate descent method for regularized multiconvex optimization with applications to nonnegativetensor factorization and completion," *SIAM J. Imag. Sci.*, vol. 6, no. 3, pp. 1758–1789, 2015.
[36] Y. He, N. Zhao, and H. Yin, "Integrated networking, caching, and computing for connected vehicles: A deep reinforcement learning approach," *IEEE Trans. Veh. Technol.*, vol. 67, no. 1, pp. 44–55, Jan. 2018.
[37] W. Jie, S. Wang, L. Zhang, Z. Ao, and F. Yang, "Minimizing data transmission latency by bipartite graph in MapReduce," in *Proc. IEEE Int. Conf. Cluster Comput.*, 2015, pp. 521–522.
[38] Y. Liu, H. Yu, S. Xie, and Y. Zhang, "Deep reinforcement learning for offloading and resource allocation in vehicle edge computing and networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 11, pp. 11158–11168, Nov. 2019.
[39] A. Prasanthi, H. Shareef, R. Errouissi, M. Asna, and A. Wahyudie, "Quantum chaotic butterfly optimization algorithm with ranking strategy for constrained optimization problems," *IEEE Access*, vol. 9, pp. 114587–114608, 2021.
[40] R. Vaze, N. Deshmukh, R. Kumar, and A. Saxena, "Development and application of quantum entanglement inspired particle swarm optimization," *Knowl.-Based Syst.*, vol. 219, no. 1, 2021, Art. no. 106859.
[41] Y. Wang and T. Du, "An improved squirrel search algorithm for global function optimization," *Algorithms*, vol. 12, no. 4, 2019, Art. no. 80.
[42] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Adv. Eng. Softw.*, vol. 69, no. 3, pp. 46–61, 2014.
[43] A. Shahidinejad and M. Ghobaei-Arani, "A metaheuristic-based computation offloading in edge-cloud environment," *J. Ambient Intell. Humanized Comput.*, vol. 13, no. 5, pp. 2785–2794, 2022.
[44] J. Liu, Y. Mao, X. Liu, and Y. Li, "A dynamic adaptive firefly algorithm with globally orientation," *Math. Comput. Simul.*, vol. 174, pp. 76–101, 2020.
[45] J.-S. Liu, Q.-Q. Liu, and F. Zuo, "A guided mutation grey wolf optimizer algorithm integrating flower pollination mechanism," *J. Comput.*, vol. 33, no. 2, pp. 51–67, 2022.

Yuliang Cong received the B.S. degree from the Department of Electrical Engineering, Jilin University, Changchun, China, in 1988, and the M.S. and Ph.D. degrees in communication and information systems from Jilin University in 1993 and 1998, respectively. In 2002, she was a Visiting Scholar with the Department of Electrical and Electronic Engineering, North Carolina State University, Raleigh, NC, USA. She is currently a Full Professor with the College of Communication Engineering, Jilin University. Her research interests include intelligent information processing, vehicular networking, array signal processing, wireless body-area networks, cognitive radio, and space–time communication.

Ke Xue received the B.S. degree from the Department of Measurement, Control Technology and Instrumentation, Jilin University, Changchun, China, in 2018, and the M.S. degree from the College of Communication Engineering, Jilin University, In 2021. He joined the Changchun Institute of Optics, Fine Mechanics and Physics, Chinese Academy of Sciences. His research interests include MEC and vehicular networks.

Cong Wang (Member, IEEE) received the B.S. degree in electronic engineering from Tsinghua University, Beijing, China, in 2013, and the Ph.D. degree in computer science from the Ecole Polytechnique Federale de Lausanne, Lausanne, Switzerland, in 2019. He is currently an Associate Professor with the College of Communication Engineering, Jilin University, Changchun, China. His research interests include computer algorithms, optimization theory, statistical learning, control theory, and networked systems.

Wenxi Sun received the B.S. degree from the College of Communication Engineering, Changchun University of Science and Technology, Changchun, China, in 2018. She is currently working toward the M.S. degree with the College of Communication Engineering, Jilin University, Changchun, China. Her research interests include Internet of Vehicles and mobile edge computing.

Shuxian Sun received the B.S. degree from the Department of Communication Engineering, Shandong Normal University, Jinan, China, in 2019. She is currently working toward the M.S. degree with the College of Communication Engineering, Jilin University, Changchun, China. Her research interests include mobile edge computing and vehicular networks.

Fengye Hu (Senior Member, IEEE) received the B.S. degree from the Department of Precision Instrument, Xi'an University of Technology, Xi'an, China, in 1996, and the M.S. and Ph.D. degrees in communication and information systems from Jilin University, Changchun, China, in 2000 and 2007, respectively. In 2011, he was a Visiting Scholar with the Department of Electrical and Electronic Engineering, Nanyang Technological University, Singapore. He is currently a Full Professor with the College of Communication Engineering, Jilin University, Changchun, China. He has authored or coauthored 50 publications in IEEE journals and conferences. His research interests include wireless body-area networks, wireless energy and information transfer, energy harvesting, cognitive radio, and space-time communication. He is the Editor of *IET Communications*, *China Communications* and Physical Communication Special Issue on Ultra-reliable Low-Latency and Low-Power Transmissions in the Era of the Internet-of-Things. He organized the first and second Asia-Pacific Workshop on Wireless Networking and Communications (APWNC 2013 and APWNC 2015). He also organized the Future 5G Forum on Wireless Communications and Networking Big Data (FWCN 2016). He was an Executive Co-Chair of the IEEE/CIC International Conference on Communications in China, China, in 2019.