Minimizing the Delay and Cost of Computation Offloading for Vehicular Edge Computing

Quyuan Luo, Changle Li, *Senior Member, IEEE*, Tom H. Luan, *Senior Member, IEEE*, and Weisong Shi, *Fellow, IEEE*

Abstract—The development of autonomous driving poses significant demands on computing resource, which is challenging to resource-constrained vehicles. To alleviate the issue, Vehicular edge computing (VEC) has been developed to offload real-time computation tasks from vehicles. However, with multiple vehicles contending for the communication and computation resources at the same time for different applications, how to efficiently schedule the edge resources toward maximal system welfare represents a fundamental issue in VEC. This article aims to provide a detailed analysis on the delay and cost of computation offloading for VEC and minimize the delay and cost from the perspective of multi-objective optimization. Specifically, we first establish an offloading framework with communication and computation for VEC, where computation tasks with different requirements for computation capability are considered. To pursue a comprehensive performance improvement during computation offloading, we then formulate a multi-objective optimization problem to minimize both the delay and cost by jointly considering the offloading decision, allocation of communication and computation resources. By applying the game theoretic analysis, we propose a particle swarm optimization based computation offloading (PSOCO) algorithm to obtain the Pareto-optimal solutions to the multi-objective optimization problem. Extensive simulation results verify that our proposed PSOCO outperforms counterparts. Based on the results, we also present a comprehensive analysis and discussion on the relationship between delay and cost among the Pareto-optimal solutions.

Index Terms—Vehicular edge computing, computation offloading, multi-objective optimization, Pareto optimality, particle swarm

1 Introduction

The connected and automated vehicles (CAVs) have recently attracted increasing interests from both academia and industry [1], [2], [3]. By integrating the computation and communication, CAVs can support a variety of novel vehicular applications, such as autonomous driving, precise fleet management and real-time video analytics, which plays a crucial role toward a safer and more convenient road experience to people [4]. However, the powerful and resource-hungry applications always require intensive computation, which poses significant challenges on resource-constrained CAVs [5]. For example, 4 TB of data per day will be generated for the autonomous driving application according to Intel [6]. For Level-5 autonomous driving, $500+$ TOPS[1] of processing capability are required [7].

1. TOPS (Tera Operations Per Second) is the unit of processor computing capability, 1 TOPS represents one trillion (10^{12}) operations per second by the processor.

- Quyuan Luo is with the School of Information Science and Technology, Southwest Jiaotong University, Chengdu, Sichuan 611756, China, with the State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an, Shaanxi 710071, China, and also with the Department of Computer Science, Wayne State University, Detroit, MI 48202 USA. E-mail: qyluo@swjtu.edu.cn.
- Changle Li is with the State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an, Shaanxi 710071, China. E-mail: clli@mail.xidian.edu.cn.
- Tom H. Luan is with the School of Cyber Engineering, Xidian University, Xi'an, Shaanxi 710071, China. E-mail: tom.luan@xidian.edu.cn.
- Weisong Shi is with the Department of Computer Science, Wayne State University, Detroit, MI 48202 USA. E-mail: weisong@wayne.edu.

Manuscript received 9 Aug. 2020; revised 19 Feb. 2021; accepted 3 Mar. 2021. Date of publication 9 Mar. 2021; date of current version 7 Oct. 2022.
(Corresponding author: Changle Li.)
Digital Object Identifier no. 10.1109/TSC.2021.3064579

The Vehicular Edge Computing (VEC) represents a practical and effective approach to support the large-scale CAVs [8], [9]. By offloading the computation-intensive tasks to roadside units (RSUs) equipped with edge servers (RES) [10], the VEC can significantly save the computation workload of vehicles yet reduce the processing latency of the computation tasks toward more efficient CAV applications. However, note that an RSU needs to serve multiple vehicles at the same time [11], how to effectively and economically use the limited edge resources and provide the maximal system welfare is a key issue.

A number of works have been developed on the allocation of edge resources in CAVs [1], [5], [10], [12], [13]. In the above works, the computation offloading is typically formulated as an optimization problem to either minimize the total processing delay or energy consumption or maximize the system utility. Distributed resource allocation methods (such as game-theoretic approach) or centralized resource allocation methods using optimization or heuristic algorithms are developed. More recently, artificial intelligence or deep reinforcement learning-based methods are proposed to solve the problems [14], [15]. The existing works mainly focus on one performance index. However, facing diverse applications, different requirements and system performance indexes during the offloading should be jointly considered. For example, to reduce delay, more cost would also be produced if tasks are offloaded to RSU to be processed, including the communication cost and computation cost. In this regard, it is imperative for CAVs to consider comprehensive performance and achieve multi-objective optimization.

As motivated, we aim to provide a detailed analysis of the delay and cost of computation offloading for VEC and minimize the delay and cost from the perspective of multi-objective optimization. To this goal, we first develop an offloading framework with communication and computation for VEC. Then, we formulate the joint resource allocation problem as a multiple-objective optimization problem, considering optimizing both delay and cost during computation offloading. The formulated problem is also a mixed-integer non-linear programming (MINLP) problem, which can not be solved effectively by traditional optimization methods. By utilizing the concept of *Pareto optimality* [16], we propose a particle swarm optimization based computation offloading (PSOCO) algorithm to solve the multi-objective optimization problem and obtain the Pareto-optimal solutions. The contributions of this paper are summarized as follows.

1) *Model*: We establish an offloading framework with communication and computation, where tasks with different computation capability requirements are considered. Under the framework, we elaborate on the detailed delay and cost of computation offloading.
2) *Multiple-Objective Optimization*: Considering the comprehensive performance for CAVs, we formulate a multi-objective optimization problem to minimize both the delay and cost, where the offloading decision, local processing capability, communication resource, and RES processing capability are jointly considered.
3) *Algorithm Design*: To solve the formulated multiple-objective optimization problem, which is also a MINLP problem, we introduce the concept of *Pareto optimality*. Motivated by the computational intelligence, we propose a particle swarm optimization based computation offloading (PSOCO) algorithm to obtain the Pareto-optimal solutions.
4) *Validation and Discussion*: Based on the real-world vehicular trace, extensive simulation results are provided to demonstrate the effectiveness of our proposed PSOCO over counterparts. Based on the obtained Pareto-optimal solutions, we provide a comprehensive analysis and discussion on the relationship between delay and cost among the Pareto-optimal solutions.

The remainder of this paper is organized as follows. We present the related work in Section 2. The system model is depicted in Section 3. The proposed PSOCO algorithm is presented in Section 4. Extensive simulation results are discussed in Section 5. We conclude this paper in Section 6.

2 Related Work

In this section, we survey the existing literature on the allocation of communication and computation resources during computation offloading.

We first present the literature specifically on resource allocation for vehicles. Du *et al.* in [5] exploit Lyapunove optimization theory and propose a DDORV algorithm to minimize the cost on the vehicle side and the RSU side, respectively. Considering the mobility of vehicles, Zhang *et al.* in [10] propose a predictive-mode transmission scheme to minimize offloading cost, by focusing on both edge server selection and transmission management. In [13], they further propose to use backup computing servers to assist the mobile edge computing (MEC) server. And a Stackelberg game-based method is adopted to maximize the utilities on both the vehicle side and the MEC server side. To reduce latency, Liu *et al.* in [12] propose a distributed computation offloading scheme through formulating the computation offloading decision-making problem as a multi-user game. The Nash equilibrium of the game is further proved.

More recently, some marvellous works adopt machine learning-based methods in this area. Dai *et al.* in [1] propose an architecture that dynamically allocates computation and caching resources. Based on the architecture, a deep reinforcement learning-based method is exploited to maximize system utility. Zhang *et al.* in [15] utilize the cognitive radio (CR) to alleviate the spectrum scarcity problem during computation offloading. To reduce transmission costs among vehicle-to-infrastructure (V2I) and vehicle-to-vehicle (V2V) communications, they propose a deep Q-learning method to schedule the communication modes and resources. Different from traditional works that study communication, caching, and computation technologies separately, He *et al.* in [14] propose an integrated framework that can orchestrate the three aspects dynamically. To solve the joint optimization problem, they utilize a deep reinforcement learning method to maximize the reward function, which is defined as the comprehensive revenue from communication, caching, and computing.

In addition to literature specifically for vehicles, other researches on MEC resource allocation are also emerging. To reduce energy consumption during computation offloading, Zhang *et al.* in [17] propose a three-stage energy-efficient computation offloading scheme through priority assignment and type classification. You *et al.* in [18] consider the edge cloud with both infinite and finite computation capacities respectively and the access mode with both TDMA and OFDMA. Wang *et al.* in [19] introduce the wireless power transfer (WPT) method and propose a unified MEC-WPT design. Dinh *et al.* in [20] formulate a distributed computation offloading problem based on game theory and propose a model-free reinforcement learning method. To minimize both delay and energy consumption, Messous *et al.* in [21] and Dinh *et al.* in [22] both transfer the multi-objective optimization problem into a single-objective optimization problem by weighting coefficients. To minimize the costs on both the user side and service provider side, Kim *et al.* in [23] propose a dual-side optimization algorithm for MEC. To maximize the total revenue, Wang *et al.* in [24] formulate an optimization problem by jointly considering the offloading decision, resource allocation, and caching in heterogeneous wireless cellular networks and propose a distributed algorithm based on alternating direction method of multipliers (ADMM).

All those works above are marvellous solutions. They often try to formulate a single-objective optimization problem such as minimizing delay, cost, or energy consumption. As for the multi-objective optimization problem, they often adopt a weighted sum method and transfer it to a single single-objective optimization problem. Moreover, most of the existing computation offloading solutions do not consider the differential requirements of computation tasks. In light

TABLE 1
Major Notations

Notation	Explanation
I_n	An indicator indicating the type of task T_n
W_1, W_2	Bandwidths of uplink and downlink channels
k_n, k_e	Coefficients related to power in CAVs and RES
D_n	Data size of task T_n
d_n	Distance between vehicle n and RSU
α_n^l, β_n^m	Indicators indicating weather RBs l and m is allocated to CAV n
K	Number of task types
λ_n	Offloading decision variable of task T_n
ϑ	Path loss exponent
h_n^l, h_n^m	Power gains between CAV n and RSU over RBs l and RB m
ρ	Price for RSU to process each unit CPU cycle
μ, ν	Price of transmission per bit data in uplink and downlink
f_n^l, F_n	Processing capability for task T_n and maximum processing capability of CAV n
f_n^e, F_e	Processing capability for task T_n and maximum processing capability of RES
c_n	Processing density of task T_n
γ_n	Ratio of the output data size to the input data size of task T_n
$\mathcal{N}$, N	Set and number of CAVs
$\mathcal{B}_1$, L	Set and number of uplink RBs
$\mathcal{B}_2$, M	Set and number of downlink RBs
p_n^l, p_n^m	Uplink and downlink transmission power between CAV n and RSU over RBs l and m
σ^l, σ^m	White Gaussian noise powers over RBs l and m

of the existing works, we take the research a step further. Specifically, we establish the computation offloading in a unified framework with communication and computation, where tasks with different computation capability requirements are considered. We also consider comprehensive system performance indexes and formulate a multi-objective optimization problem to minimize both delay and cost during computation offloading. Furthermore, Pareto-optimal solutions to the problem are obtained through our proposed PSOCO algorithm.

3 System Model

This section presents the system model, including the offloading framework with communication and computation for VEC, task local processing model, computation offloading model, and problem formulation. For convenience, we summarize the major notations in Table 1.

3.1 Offloading Framework With Communication and Computation for VEC

Fig. 1 shows the offloading framework for VEC. The road is partitioned into segments, and each is covered by a roadside

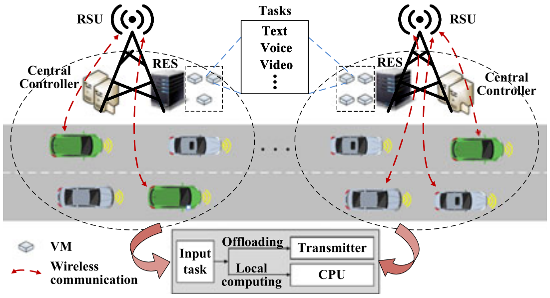


Fig. 1. Offloading framework for VEC.

unit (RSU) with a roadside edge server (RES). In this paper, we consider a coverage area of one RSU and a set of $\mathcal{N} = \{1, 2, \ldots, N\}$ CAVs (hereinafter referred to as *vehicle* for short). Various task data would be generated from the on-board applications of vehicles for entertainment (e.g., face recognition and augmented reality) or safety (e.g., LiDAR and high-definition camera) purpose [25]. We denote the number of task types by K. The RSU can provide powerful computing capability due to the deployed RES. Each vehicle n ($n \in \mathcal{N}$) has a computation task T_n to be processed. We use four items to describe T_n as $T_n \triangleq \{D_n, \gamma_n, c_n, I_n\}$, where D_n stands for the input data size of T_n, γ_n is a ratio of output data size to input data size, c_n stands for the processing density (in CPU cycles/bit) of the task, and I_n is an indicator that stands for the type of task T_n, and different type of tasks have different processing densities. We use W_1 and W_2 to represent the total bandwidths of uplink and downlink of V2I channels, respectively. By leveraging the non-orthogonal multiple access (NOMA) technique, which has been considered as a key enabling technique for 5G networks due to its potentially superior spectral efficiency [26], the communication resource is divided into resource blocks (RBs),[2] expressed as $\mathcal{B}_1 = \{1, 2, \ldots, L\}$ for uplink and $\mathcal{B}_2 = \{1, 2, \ldots, M\}$ for downlink. It is noting that there is a control center at each RSU, which gathers task processing requirements from vehicles while scheduling the communication and computation resources allocation through a dedicated control channel [15].

For ease of analysis, we consider the system to be quasi-static so that the topology and wireless channels keep unchanged during the task processing period [5]. We define λ_n ($0 \le \lambda_n \le 1$) as the offloading decision variable of task T_n, which stands for the ratio of the amount of bits offloaded to RSU to D_n. Accordingly, the amount of bits that would be offloaded to RSU is $\lambda_n D_n$ bits and that would be processed locally is $(1 - \lambda_n) D_n$ bits. In the following, we will elaborate on the local processing part and offloading part, respectively.

3.2 Task Processed Locally

We use f_n^l ($0 \le f_n^l \le F_n$) to denote the processing capability (in CPU cycles/s) at vehicle n assigned for local computation, where F_n is the maximum processing capability of vehicle n. The power consumption of vehicle n is then

2. This can be easily achieved by integrating network function virtualization (NFV) and software defined networking (SDN) technologies, by which various radio spectrum resources can be abstracted and sliced to the RSUs and then be allocated to CAVs by each RSU [27].

calculated as $p_n^l = k_n(f_n^l)^3$, where k_n is a coefficient related to power in CAV n [28]. For the $(1-\lambda_n)D_n$-bits of task T_n, it's local processing time is formulated as $t_n^l = \frac{c_n\left(1-\lambda_n\right)D_n}{f_n^l}$. Accordingly, the energy consumption of vehicle n for processing the local part of task T_n is then formulated as $E_n^l = k_n c_n\left(1-\lambda_n\right)D_n\left(f_n^l\right)^2$.

3.3 Task Offloaded to RSU

For the offloaded part of task T_n, there exist three procedures to accomplish the task computing, that is, $\lambda_n D_n$-bits of T_n should be first transmitted to RSU through V2I channels, then computed by RSU, and finally, RSU will return the result to vehicle n.

3.3.1 Task Transmission

We denote α_n^l $(l \in \mathcal{B}_1)$ as the uplink binary RB allocation indicator, where $\alpha_n^l = 1$ means l is allocated to vehicle n, and $\alpha_n^l = 0$ otherwise. We denote p_n^l as the uplink transmission power of vehicle n over RB l, h_n^l as the power gain between vehicle n and RSU over RB l. Accordingly, the uplink data rate of vehicle n over RB l after performing successive interference cancellation (SIC)[3] can be formulated as $r_n^l = \frac{W_1}{L}\log_2(1+\frac{\alpha_n^l p_n^l h_n^l}{\zeta_n^l+\sigma^l(d_n)^\vartheta})$, where σ^l denotes the White Gaussian noise power over RB l [30], $\zeta_n^l = \sum_{i\in\mathcal{N}, i\neq n, h_n^l<h_i^l}\alpha_i^l p_i^l h_i^l$ is the interference signal power from other vehicles over RB l, d_n and ϑ are the distance from vehicle n to RSU and the path loss exponent, respectively. Through SIC technology, the interference signal from vehicle $i \in \mathcal{N}\setminus\{n\}$ will be decoded and removed at the RSU side if $h_i^l < h_n^l$ [31]. The uplink data rate from vehicle n is then formulated as $R_1^n = \sum_{l\in\mathcal{B}_1} r_n^l$. Accordingly, the uplink transmission delay and energy consumption for offloading $\lambda_n D_n$-bits of task T_n are obtained by $t_n^{up} = \frac{\lambda_n D_n}{R_1^n}$ and $E_n^{up} = \sum_{l\in\mathcal{B}_1} p_n^l t_n^{up}$, respectively.

3.3.2 Task Computed by RSU

After $\lambda_n D_n$-bits of task T_n is transmitted from vehicle n to RSU, it will be computed by the deployed RES. We use f_n^e $(0 \le f_n^e \le F_e)$ to denote the assigned processing capability (in CPU cycles/s) from RES, where F_e denotes the maximum processing capability of RES. The power consumption of RES is then calculated as $p_n^e = k_e(f_n^e)^3$, where k_e is a coefficient related to power in RES. For the $\lambda_n D_n$-bits of task T_n, it's processing time is expressed as $t_n^e = \frac{c_n\lambda_n D_n}{f_n^e}$. And the energy consumption of RES for computing the offloaded part of task T_n is expressed as $E_n^e = k_e c_n \lambda_n D_n (f_n^e)^2$.

3.3.3 Result Return

We denote β_n^m $(m \in \mathcal{B}_2)$ as the downlink binary RB allocation indicator, where $\beta_n^m = 1$ means RB m is allocated to vehicle n, and $\beta_n^m = 0$ otherwise. We denote p_n^m as the downlink transmission power of RSU sending back the computation result to vehicle n over RB m, h_n^m as the power gain between vehicle n and RSU over RB m. Accordingly, the downlink data rate of vehicle n for result return from RSU over RB m can be formulated as $r_n^m = \frac{W_2}{M}\log_2(1+\frac{\beta_n^m p_n^m h_n^m}{\zeta_n^m+\sigma^m(d_n)^\vartheta})$, where σ^m denotes the White Gaussian noise power over RB m, $\zeta_n^m = \sum_{j\in\mathcal{N}, j\neq n, h_n^m<h_j^m}\alpha_j^m p_j^m h_j^m$ denotes the interference signal power from other vehicles over RB m. The downlink data rate from RSU is then formulated as $R_2^n = \sum_{m\in\mathcal{B}_2} r_n^m$. Accordingly, the downlink transmission delay and energy consumption for result return can be obtained by $t_n^d = \frac{\gamma_n\lambda_n D_n}{R_2^n}$ and $E_n^d = \sum_{m\in\mathcal{B}_2} p_n^m t_n^d$, respectively.

Therefore, the total latency and energy consumption for the offloaded part of task T_n are expressed as

$$t_n^{off} = t_n^{up} + t_n^e + t_n^d = \frac{\lambda_n D_n}{R_1^n} + \frac{c_n\lambda_n D_n}{f_n^e} + \frac{\gamma_n\lambda_n D_n}{R_2^n}, \tag{1}$$

and

$$\begin{aligned} E_n^{off} &= E_n^{up} + E_n^e + E_n^d \\ &= \sum_{l=1}^{L} p_n^l t_n^{up} + k_e c_n \lambda_n D_n\left(f_n^e\right)^2 + \sum_{m=1}^{M} p_n^m t_n^d, \end{aligned} \tag{2}$$

respectively.

3.4 Problem Formulation

For a given task T_n, delay and cost would be produced to process it. For the delay aspect, it is determined by both the delays of processing the local part and the offloaded part. Accordingly, the total delay of processing T_n can be expressed as $t_n = \max\{t_n^l, t_n^{off}\}$. For the cost part, it is also determined by both the cost of processing the local part and the offloaded part. The former only includes the energy consumption for local computing while the latter includes three aspects: *a)* the energy consumption of transmitting and computing task T_n; *b)* the communication cost for using RBs; and *c)* the computing cost for RES computing the offloaded part of task T_n. Accordingly, the cost for processing task T_n can be expressed as

$$\begin{aligned} U_n &= U_n^l + U_n^{off} \\ &= \xi E_n^l + \xi\left(E_n^{up} + E_n^e + E_n^d\right) \\ &\quad + \mu\lambda_n D_n + \nu\gamma_n\lambda_n D_n + \rho c_n \lambda_n D_n, \end{aligned} \tag{3}$$

where ξ is a weighting coefficient indicating the energy consumption cost of one unit energy during task computing and transmission [23], μ and ν are coefficient s indicating the communication cost required to transmit one unit of task data by using uplink and downlink RBs, respectively. And ρ is a coefficient indicating the computing cost to execute one CPU cycle.

In this paper, we consider minimizing the delay and cost of all vehicles under the computation capability and communication resource limitations. To this end, the offloading decision variable, the local processing capability, the RES processing capability, the uplink binary RB allocation indicator, and the downlink binary RB allocation indicator need to be optimized. We denote $\boldsymbol{\lambda} = \{\lambda_1, \ldots, \lambda_N\}$, $\boldsymbol{f}^l = \{f_1^l, \ldots, f_N^l\}$, $\boldsymbol{f}^e = \{f_1^e, \ldots, f_N^e\}$, $\boldsymbol{\alpha} = \{\alpha_1^1, \ldots, \alpha_1^L, \ldots, \alpha_N^1, \ldots, \alpha_N^L\}$, $\boldsymbol{\beta} =$

3. Through superposition coding of signal at the vehicle side and retrieving it at the RSU side, the SIC method in NOMA supports simultaneous transmission of multiple signals on one RB, which can improve the spectral efficiency [29].

$\{\beta_1^1,\ldots,\beta_1^M,\ldots,\beta_N^1,\ldots,\beta_N^M\}$. Thus, the multi-objective optimization problem is formulated as

$$\begin{aligned}
&\min_{\{\boldsymbol{\lambda},\boldsymbol{f}^l,\boldsymbol{f}^e\boldsymbol{\alpha},\boldsymbol{\beta}\}} \quad t=\sum_{n=1}^{N} t_n \\
&\min_{\{\boldsymbol{\lambda},\boldsymbol{f}^l,\boldsymbol{f}^e\boldsymbol{\alpha},\boldsymbol{\beta}\}} \quad U=\sum_{n=1}^{N} U_n \\
\text{s.t.}\quad &\text{1-C1}: \quad 0\le \lambda_n \le 1, \forall n\in\mathcal{N}, \\
&\text{1-C2}: \quad 0\le f_n^l \le F_n, \forall n\in\mathcal{N}, \\
&\text{1-C3}: \quad 0\le f_n^e \le F_e, \forall n\in\mathcal{N}, \\
&\text{1-C4}: \quad \sum_{n=1}^{N} f_n^e \le F_e, \\
&\text{1-C5}: \quad \alpha_n^l, \beta_n^m \in \{0,1\}, \forall n\in\mathcal{N}, \\
&\qquad\qquad l\in\mathcal{B}_1, m\in\mathcal{B}_2, \\
&\text{1-C6}: \quad \sum_{n=1}^{N}\alpha_n^l\le 1, \sum_{n=1}^{N}\beta_n^m \le 1, \\
&\qquad\qquad \forall l\in\mathcal{B}_1, m\in\mathcal{B}_2,
\end{aligned} \tag{4}$$

where (1-C1) is the constrains on offloading decision variable; (1-C2), (1-C3) and (1-C4) are the processing capability constraints for vehicles and RES, where F_e denotes the maximum processing capability of RES at the RSU; (1-C5) are the binary constrains on uplink and downlink RB allocation indicators; (1-C6) ensures that each uplink RB and each downlink RB can be allocated to at most one vehicle.

Since the delay and energy consumption in downlink is much less than in uplink for most traditional computation-intensive tasks [18], [32]. Accordingly, for simplicity, we ignore the delay and energy consumption in downlink. Then Formulas (1) and (2) can be rewritten as

$$t_n^{off}=t_n^{up}+t_n^{e}=\tfrac{\lambda_n D_n}{R_1^n}+\tfrac{c_n\lambda_n D_n}{f_n^e}, \tag{5}$$

and

$$E_n^{off}=E_n^{up}+E_n^{e}=\sum_{l=1}^{L} p_n^l t_n^{up}+k_e c_n\lambda_n D_n\left(f_n^e\right)^2, \tag{6}$$

respectively. Accordingly, the problem in Formula (4) can be reformulated as

$$\begin{aligned}
&\min_{\{\boldsymbol{\lambda},\boldsymbol{\alpha},\boldsymbol{f}^l,\boldsymbol{f}^e\}} t=\sum_{n=1}^{N}\max\left\{\frac{c_n\left(1-\lambda_n\right)D_n}{f_n^l}, \lambda_n D_n\left(\frac{1}{R_1^n}+\frac{c_n}{f_n^e}\right)\right\} \\
&\min_{\{\boldsymbol{\lambda},\boldsymbol{\alpha},\boldsymbol{f}^l,\boldsymbol{f}^e\}} U=\sum_{n=1}^{N}\xi D_n\left\{k_n c_n\left(1-\lambda_n\right)\left(f_n^l\right)^2+\frac{\lambda_n\sum_{l=1}^{L}p_n^l}{R_1^n}\right. \\
&\qquad\qquad \left. + k_e c_n\lambda_n\left(f_n^e\right)^2\right\}+\lambda_n D_n\left\{\mu+\nu\gamma_n+\rho c_n\right\} \\
\text{s.t.}\quad &\text{2-C1}: 0\le\lambda_n\le 1, \forall n\in\mathcal{N}, \\
&\text{2-C2}: 0\le f_n^l\le F_n, \forall n\in\mathcal{N}, \\
&\text{2-C3}: 0\le f_n^e\le F_e, \forall n\in\mathcal{N}, \\
&\text{2-C4}: \sum_{n=1}^{N} f_n^e\le F_e, \\
&\text{2-C5}: \alpha_n^l\in\{0,1\}, \forall n\in\mathcal{N}, l\in\mathcal{B}_1, \\
&\text{2-C6}: \sum_{n=1}^{N}\alpha_n^l\le 1, \forall l\in\mathcal{B}_1,
\end{aligned} \tag{7}$$

where R_1^n denotes the uplink data rate from from vehicle n and is formulated as

$$R_1^n=\sum_{l=1}^{L}\frac{W_1}{L}\log_2\left(1+\frac{\alpha_n^l p_n^l h_n^l}{\sum_{i\in\mathcal{N},i\neq n h_n^l<h_i^l}\alpha_i^l p_i^l h_i^l+\sigma^l\left(d_n\right)^{\vartheta}}\right). \tag{8}$$

4 Multi-Objective Optimization and Computation Offloading Algorithm

4.1 Multi-Objective and Pareto Optimization

The purpose of Formula (7) is to minimize the cost and delay simultaneously, which is a multi-objective optimization problem. The improvement in the performance of one objective may cause a decrease in the performance of another objective during the optimization. Therefore, the two objectives cannot achieve their optimal values at the same time, a trade-off and compromise should be obtained between them. There is no optimal solution to Formula (7) but a set of optimal solutions, the elements of which are called Pareto-optimal solutions or non-dominated solutions [33].

Definition 1. *Consider an optimization of a multi-criteria decision-making problem aiming to minimize κ objective functions $f_q(\mathbf{x})$ $(q=1,2,\ldots,\kappa)$, where $\mathbf{x}=[x_1,x_2,\ldots,x_\iota]^T$ is a vector of ι decision variables. The set of all feasible solutions is denoted as Ω. Assume $S_1\in\Omega$ and $S_2\in\Omega$, it is said that solution S_1 dominates the other solution S_2 (denoted by $S_1\succ S_2$), if and only if $\forall q:\ f_q(\mathbf{x}_{S_1})\le f_q(\mathbf{x}_{S_2})$ and $\exists k:\ f_q(\mathbf{x}_{S_1})<f_q(\mathbf{x}_{S_2})$, where $f_q(\mathbf{x})$ are the objective functions and $\mathbf{x}_{S_1}$, $\mathbf{x}_{S_2}$ are vectors of ι variables representing S_1 and S_1, respectively [34].*

Definition 2. *All feasible solutions in search space Ω, which are not dominated (see Definition 1) by any other solution in the search space are called Pareto-optimal solutions. They form the so-called Pareto-optimal set (or Pareto-optimal front).*

Based on Definitions 1 and 2, this paper aims to find the Pareto-optimal solutions for the formulated delay and cost minimization problem in Formula (7). Since the problem formulated in Formula (7) is a MINLP problem, it is hard to find the Pareto-optimal solutions through traditional optimization methods [35]. Inspired by the swarm intelligence, we resort to the particle swarm optimization (PSO) [36] method and propose a PSO based computation offloading (PSOCO) algorithm. In the following, we will give a brief introduction about PSO and elaborate on the PSOCO algorithm.

4.2 Particle Swarm Optimization

As a computational intelligence method, PSO is inspired by the swarm behavior of birds to search for food, in which each bird changes its search pattern by learning from its own and others' experience [33]. PSO has been widely used in optimization problems in various fields due to its good ability to solve complex problems in multi-dimensional complex space [37]. Compared with other swarm intelligence algorithms, PSO has fewer parameters and is very suitable for multi-objective optimization problems [38]. More importantly, the PSO has a faster convergence speed than other population-based stochastic optimization methods (e.g., genetic algorithms) [39]. To describe the proposed PSOCO explicitly, we first introduce some terms about PSO.

1) *Particle:* The search space for solving optimization problems is compared to the flight space of birds during the foraging behavior, and each bird is abstracted into a particle without mass or volume. In this paper, a particle denotes a candidate solution of Formula (7).
2) *Particle swarm:* A particle swarm consists of several particles, and the number of particles denotes the size of the particle swarm.
3) *Solution encoding:* Solution encoding means how to represent the variables to be solved in Formula (7) by the particle. In this paper, we adopt the real encoding method.
4) *Fitness:* Fitness indicates the suitability of a particle for the solution and is presented by the objective function value. In this paper, fitness means the optimization objectives in Formula (7), i.e., the delay and cost.
5) *Fitness evaluation:* Fitness evaluation represents to calculate the fitness value (i.e., the value of the two objectives in Formula (7)) for each particle.
6) *Swarm updating:* Based on the best positions of individuals and swarm, particles update their speeds and positions. The swarm updating is completed after all particles are updated. Through swarm updating, the evolution of candidate solutions can be achieved.
7) *Boundary condition processing:* When the position and speed exceed the set value, the boundary condition processing can restrict particle's position to a feasible space, which avoids the expansion and divergence of the swam, and also avoids the blindly searching in a large range, thus improving the search efficiency.

4.3 Proposed PSOCO Algorithm

1) *Initialization:* Initialize iteration $g = 0$ and particle swarm $\mathcal{S}(g)$ randomly as $\mathcal{S}(g) = \{S_1(g), S_2(g), \dots, S_s(g)\}$, where s denotes the swarm size, $S_i(g)$ $(1 \le i \le s)$ denotes a particle. Specifically, $S_i(g)$ is represented by a set of $\boldsymbol{\lambda}$, $\boldsymbol{\alpha}$, $\boldsymbol{f}^l$ and $\boldsymbol{f}^e$, which can be defined as an array $S_i(g) = [\boldsymbol{\lambda}, \boldsymbol{\alpha}, \boldsymbol{f}^l, \boldsymbol{f}^e]_d$, where d is the size of the array and is defined as $d = 3 \times N + L$. In each $S_i(g)$, the size of $\boldsymbol{\lambda}$, $\boldsymbol{f}^l$ and $\boldsymbol{f}^e$ is N, the size of $\boldsymbol{\alpha}$ is L. And $\boldsymbol{\lambda}$ is binary coded, $\boldsymbol{\alpha}$, $\boldsymbol{f}^l$ and $\boldsymbol{f}^e$ are real coded. It is worth noting that the elements in $\boldsymbol{\alpha}$ are different from that in constraints (3-C5) and (3-C6). Here, the $\boldsymbol{\alpha}$ is defined as $\boldsymbol{\alpha} = \{\alpha^l | l \in \mathcal{B}_1, 1 \le \alpha^l \le N\}$, where α^l is an integer to indicate the vehicle which obtains the RB l.

   In addition, we define four memory units $\mathcal{P}^{indv}$, $\mathcal{O}^{indv}$, $\mathcal{P}^{glb}$ and $\mathcal{O}^{glb}$ to represent the best positions of individuals, the best objectives of individuals, the best positions of swarm, and the best objectives of swarm, respectively. Specifically, $\mathcal{P}^{indv}$ and $\mathcal{O}^{indv}$ are expressed by two cell arrays with the size of s, where each element in the two cell arrays represents the Pareto-optimal set and corresponding optimal objectives of a particle, respectively. $\mathcal{P}^{indv}$ and $\mathcal{O}^{indv}$ are formulated as

$$\mathcal{P}^{indv} = \left\{P_1^{indv}, P_2^{indv}, \dots, P_s^{indv}\right\}, \tag{9}$$

$$\mathcal{O}^{indv} = \left\{O_1^{indv}, O_2^{indv}, \dots, O_s^{indv}\right\}. \tag{10}$$

   $\mathcal{P}^{glb}$ and $\mathcal{O}^{glb}$ are two arrays, where each element in the two arrays represents one of the optimal Pareto-optimal set and corresponding optimal objectives of the swarm, respectively. For the sake of iteration, we assign the value of the initialized $\mathcal{S}(g)$ to $\mathcal{P}^{indv}$, and randomly select an element from $\mathcal{P}^{indv}$ and assign it to $\mathcal{P}^{glb}$. For the initialization of $\mathcal{O}^{indv}$ and $\mathcal{O}^{glb}$, we assign some values that are big enough (e.g., infinity) to them. We also initialize each particle's speed V_i as $V_i = \{v_{i,1}, v_{i,2}, \dots, v_{i,d}\}, \forall 1 \le i \le s$.
2) *Mapping of particle representation to optimization variables:* This step is also called solution encoding. Map each element $value_l$ $(1 \le l \le 3 \times N + L)$ of each $S_i(g)$ in the particle swarm to the variables to be solved in Formula (7). Specifically, since the elements in $\boldsymbol{\alpha}$, $\boldsymbol{f}^l$ and $\boldsymbol{f}^e$ are real coded, they can directly represent the variables of α_n, f_n^l and f_n^e $(n \in \mathcal{N})$. For the element α^l in $\boldsymbol{\alpha}$, the corresponding relationship with α_n^l in Formula (7) is expressed as

$$\alpha_n^l = \begin{cases} 1, & \alpha^l = n, \forall l \in \mathcal{B}_1, n \in \mathcal{N} \\ 0, & otherwise \end{cases}. \tag{11}$$

3) *Fitness evaluation:* Since we aim to minimize the cost and delay simultaneously, we regard both the cost and delay as the fitness functions. In order to find the Pareto-optimal solutions, we first calculate the fitness of each particle $S_i(g)$ $(1 \le i \le s)$ in $\mathcal{S}(g)$. Specifically, we use the variable values from the particle representation in step *2)* to calculate the objectives of delay and cost in Formula (7), and obtained the fitness values $O_{g,i}^{delay}$ $(1 \le i \le s)$ and $O_{g,i}^{cost}$, we use $O_{g,i}$ to represent the pair of objectives. It can be defined as $O_{g,i} = \{O_{g,i}^{delay}, O_{g,i}^{cost}\}$.
4) *Recording the best individuals and swarm:* After the two fitness values of each particle $S_i(g)$ are obtained by step *3)*, we then compare them with the optimal objectives O_i^{indv} and O^{glb}, respectively.

   For the recording of the best individuals, we first calculate the size of P_i^{indv} by s_i^{indv}, i.e., the number of optimal objectives of particle $S_i(g)$. Then, for any Pareto-optimal solution $P_{i,k}^{indv}$ $(1 \le k \le s_i^{indv})$ in P_i^{indv}

and its corresponding optimal objective $O_{i,k}^{indv}$ in O_i^{indv}, we judge whether $S_i(g)$ $(1 \le i \le s)$ is dominated by $P_{i,k}^{indv}$ $(1 \le k \le s_i^{indv})$ according to $O_{g,i}$ and $O_{i,k}^{indv}$. Based on Definition 1, if $\exists k : P_{i,k}^{indv} \succ S_i(g)$, then the current solution $S_i(g)$ is not the Pareto-optimal solution; otherwise (i.e., $S_i(g)$ is not dominated by any solution $P_{i,k}^{indv}$ $(1 \le k \le s_i^{indv})$ in P_i^{indv}), add $S_i(g)$ to the Pareto-optimal set P_i^{indv} and add $O_{g,i}$ to optimal objectives O_i^{indv}.

For the recording of the best swarm, we first calculate the size of $\mathcal{P}^{glb}$ by s^{glb}, then for any Pareto-optimal solution P_ς^{glb} $(1 \le \varsigma \le s^{glb})$ in $\mathcal{P}^{glb}$ and its corresponding optimal objective O_ς^{glb} in $\mathcal{O}^{glb}$, judge whether $S_i(g)$ $(1 \le i \le s)$ is dominated by P_ς^{glb} according to $O_{g,i}$ and O_ς^{glb}. Based on Definition 1, if $\exists \varsigma : P_\varsigma^{glb} \succ S_i(g)$, then the current solution $S_i(g)$ is not the Pareto-optimal solution; otherwise (i.e., $S_i(g)$ is not dominated by any solution P_ς^{glb} $(1 \le \varsigma \le s^{glb})$ in $\mathcal{P}^{glb}$), add $S_i(g)$ to the Pareto-optimal set $\mathcal{P}^{glb}$ and add $O_{g,i}$ to optimal objectives $\mathcal{O}^{glb}$.

5) *Judgement of termination condition:* If reach the termination iteration g_{max}, map the particle representation in Pareto-optimal set $\mathcal{P}^{glb}$ to the variables in Formula (7); Otherwise, go to step *6)*.

Algorithm 1. Particle Swarm Optimization Based Computation Offloading (PSOCO) Algorithm

Initialize $\mathcal{S}(g)$, $\mathcal{P}^{indv}$, $\mathcal{O}^{indv}$, $\mathcal{P}^{glb}$, $\mathcal{O}^{glb}$, V_i $(1 \le i \le s)$
for iteration $g = 0, 1, 2, \ldots, g_{max}$:
  for each particle:
    Map $S_i(g)$ to the variables to be solved in Formula (7)
    Calculate the fitness values $O_{g,i}^{delay}$ and $O_{g,i}^{cost}$ according to Formula (7), and define $O_{g,i} = \{O_{g,i}^{delay}, O_{g,i}^{cost}\}$
    Calculate the size of P_i^{indv} by s_i^{indv}
    Calculate the size of $\mathcal{P}^{glb}$ by s^{glb}
    for each Pareto-optimal solution $P_{i,k}^{indv}$ in P_i^{indv}:
      if $S_i(g)$ is not dominated by $P_{i,k}^{indv}$ **then**
        Add $S_i(g)$ to set P_i^{indv}
        Add $O_{g,i}$ to set O_i^{indv}
    end for
    for each Pareto-optimal solution P_ς^{glb} in $\mathcal{P}^{glb}$:
      if $S_i(g)$ is not dominated by P_ς^{glb} **then**
        Add $S_i(g)$ to set $\mathcal{P}^{glb}$
        Add $O_{g,i}$ to set $\mathcal{O}^{glb}$
    end for
    Randomly choose one Pareto-optimal solution from P_i^{indv} and $\mathcal{P}^{glb}$, denoted as $P_b^{indv}(g)$ and P_b^{glb}, respectively
    Each particle updates its speed and position according to Formulas (16) and (17)
    for each element $s_{i,\epsilon}(g)$ in $S_i(g)$:
      if $s_{i,\epsilon}(g)$ is beyond the position ranges **then**
        Re-initialize $s_{i,\epsilon}(g)$
  end for
end for

6) *Position and speed updating of individuals:* This step is also called swarm updating. Since there are multiple Pareto-optimal solutions of a particle in P_i^{indv} $(1 \le i \le s)$ and Pareto-optimal solutions of the swarm in $\mathcal{P}^{glb}$, we select randomly select one Pareto-optimal solution from each of P_i^{indv} and $\mathcal{P}^{glb}$, denoted by $P_b^{indv}(g)$ and $P_b^{glb}(g)$, respectively. Now, for each particle, we have the current position $S_i(g)$, the current speed $V_i(g)$, the randomly selected Pareto-optimal solutions $P_b^{indv}(g)$ and $P_b^{glb}(g)$, formulated as

$$S_i(g) = \Big\{s_{i,1}(g), s_{i,2}(g), \ldots, s_{i,d}(g)\Big\}, \tag{12}$$

$$V_i(g) = \Big\{v_{i,1}(g), v_{i,2}(g), \ldots, v_{i,d}(g)\Big\}, \tag{13}$$

$$P_b^{indv}(g) = \Big\{p_{b,1}^{indv}(g), p_{b,2}^{indv}(g), \ldots, p_{b,d}^{indv}(g)\Big\}, \tag{14}$$

$$P_b^{glb}(g) = \Big\{p_{b,1}^{glb}(g), p_{b,2}^{glb}(g), \ldots, p_{b,d}^{glb}(g)\Big\}. \tag{15}$$

Then, each particle updates speed and position as

$$\begin{aligned} v_{i,\epsilon}(g+1) = {} & \omega v_{i,\epsilon}(g) + \delta_1\theta_1(p_{b,\epsilon}^{indv}(g) - s_{i,\epsilon}(g)) \\ & + \delta_2\theta_2(p_{b,\epsilon}^{glb}(g) - s_{i,\epsilon}(g)), \forall \epsilon \in \{1, \ldots, d\}, \end{aligned} \tag{16}$$

$$s_{i,\epsilon}(g+1) = s_{i,\epsilon}(g) + v_{i,\epsilon}(g+1), \forall \epsilon \in \{1, \ldots, d\}, \tag{17}$$

where ω denotes an inertia weight factor between 0.8 and 1.2, and can be dynamically adjusted according to the linear decrement strategy, shown as

$$\omega = \omega_{max} - \frac{(\omega_{max} - \omega_{min}) \times g}{g_{max}}. \tag{18}$$

And δ_1 and δ_2 denote learning factor, also named acceleration constant. θ_1 and θ_1 are two random numbers uniformly distributed between 0 and 1. After step *6)*, the swarm is denoted as

$$\mathcal{S}'(g) = \Big\{S_1'(g), S_2'(g), \ldots, S_s'(g)\Big\}. \tag{19}$$

7) *Boundary condition processing:* For each particle, judge whether $s_{i,\epsilon}(g+1)$ is beyond the position ranges defined by constrains (3-C1)$\sim$(3-C6) in Formula (7). If $s_{i,\epsilon}(g+1)$ is beyond the position ranges, re-initialize it according to step *1)*. After step *7)*, the swarm is denoted as

$$\mathcal{S}''(g) = \Big\{S_1''(g), S_2''(g), \ldots, S_s''(g)\Big\}. \tag{20}$$

Then update the iterative number $g := g + 1$ and return to step *2)*.

We summarize the PSOCO algorithm in Algorithm 1. It is worth noting that the PSOCO algorithm can be also applied to other distributed RUSs and edge servers. It is also worth noting that vehicles may enter and leave the coverage of the RSU during task offloading, leading to a failed result reception from RSU and a higher delay and cost. To address this issue, we can first adopt a duration prediction method to evaluate the link duration between vehicles and RSU based on the current position and speed of vehicles, and the position of RSU. Then, a threshold duration is set, guaranteeing the task processing result can be returned before the vehicle leaves the coverage of the RSU. Only the vehicles whose predicted link duration is longer than the threshold duration can participate in the task offloading process. Another way is utilizing the cooperation between adjacent RSUs. The remaining task data can be offloaded to the next RSU if the vehicle leaves the coverage of the current RSU during task offloading, or can be migrated from the current RSU to the next RSU if the vehicle leaves the coverage of the current RSU during task processing. Also, the vehicles that will enter the coverage of the RSU during the process of self-learning of a vehicle can also participate in the task offloading process in the current RSU. For simplicity, in this paper, we assume that all vehicles can receive the results before they leave the coverage of the RSU.

5 Simulation Results and Discussions

5.1 Simulation Setup

We consider a two-way two-lane road with a length of 1000 m. The width of each lane is 4 m. And one RSU is deployed in the middle of the roadside and its coverage radius is 500 m. Vehicles that are on two different lanes are keeping moving back and forth along their lanes. For the behavior of vehicles, we use part of the GAIA Open Dataset containing mobility traces of DiDi Express in Xi'an China [40]. We randomly choose $20 \sim 60$ traces in our simulation. The data size is randomly distributed between 0.1 and 1 MB. The processing density is selected from $\{10, 100, 1000\}$ for type 1, 2, and 3 tasks, respectively. We present the parameters setting in Table 2.

For the simulation environment, we use a GPU-based server, where the CPU is Intel Xeno(R) E5-2690v4 with 64 GB memory. The software environment is Python 3.7 on Ubuntu16.04.6 LTS.

TABLE 2
Simulation Parameters

Description	Parameter	Value
Coefficients related to power in vehicle and RSU	k_n, k_e	10^{-27}, 10^{-29}
Downlink bandwidth	W_2	10 MHz
Downlink cost coefficient	ν	0.5×10^{-10}
Cost for RES processing task data	ρ	3×10^{-10}
Energy consumption cost coefficient	ξ	2.44×10^{-4}
Indicator of task type	I_n	$\{1, 2, 3\}$
Inertia weight factor	ω	$0.4 \sim 0.8$
Input data size	D_n	$0.1 \sim 1$ MB
Learning factor	θ_1, θ_2	1.5
Maximal local and RES processing capability	F_n, F_e	$\{1.4, 3\} \times 10^9$
Number of vehicles	N	$20 \sim 60$
Number of uplink RBs	L	$10 \times N$
Number of downlink RBs	M	$10 \times N$
Processing density of task T_n	c_n	$\{10, 100, 1000\}$
Power gains	h_n^l, h_n^m	1
Path loss exponent	ϑ	4
Ratio of output to input data size	γ_n	$0.05 \sim 0.2$
Swarm size	s	200
Transmission power over RB l	p_n^l	1 W
Transmission power over RB m	p_n^m	2 W
Uplink bandwidth	W_1	10 MHz
Uplink cost coefficient	μ	1.16×10^{-10}
White Gaussian noise powers	σ^l, σ^m	-100 dBm

5.2 Simulation Results

We consider the following schemes as benchmarks to evaluate our proposed PSOCO: 1) Offload-Comp-Only (OCO), where all vehicles offload their computation tasks to RSU to be processed; 2) Local-Comp-Only (LCO), where all vehicles compute their computation tasks locally; 3) GACO, a genetic algorithm-based computation offloading scheme.

5.2.1 *Effectiveness*

We first conduct simulations to verify the effectiveness of the proposed PSOCO. As a comparison, the GACO scheme is evaluated under the same conditions. In this set of simulations, we set $N = 20$. As shown in Fig. 2, we give the results for delay and cost varying with iterations. It is worth noting that we optimize the two objectives separately. From the figure, we can see that both PSOCO and GACO can converge to optimal solutions for delay and cost. Specifically, for PSOCO, its delay converges at about the 300th iteration and its cost converges at the 180th iteration. However, for the GACO scheme, its delay converges at about the 350th iteration and its cost converges at about the 310th iteration. Both results verify the effectiveness of our proposed PSOCO and show that PSOCO has a faster convergence than the benchmark GACO.

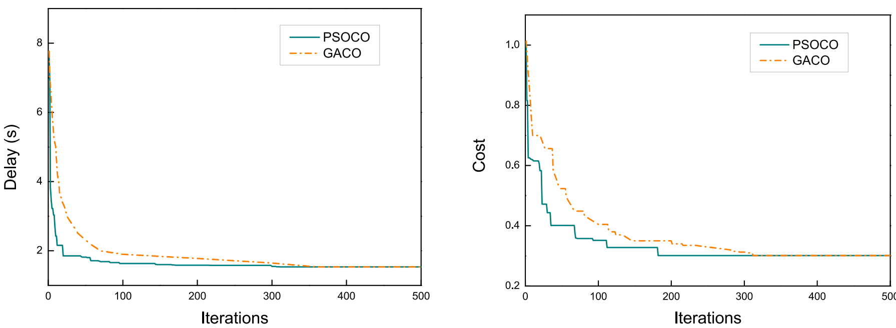


Fig. 2. Convergence of delay and cost.

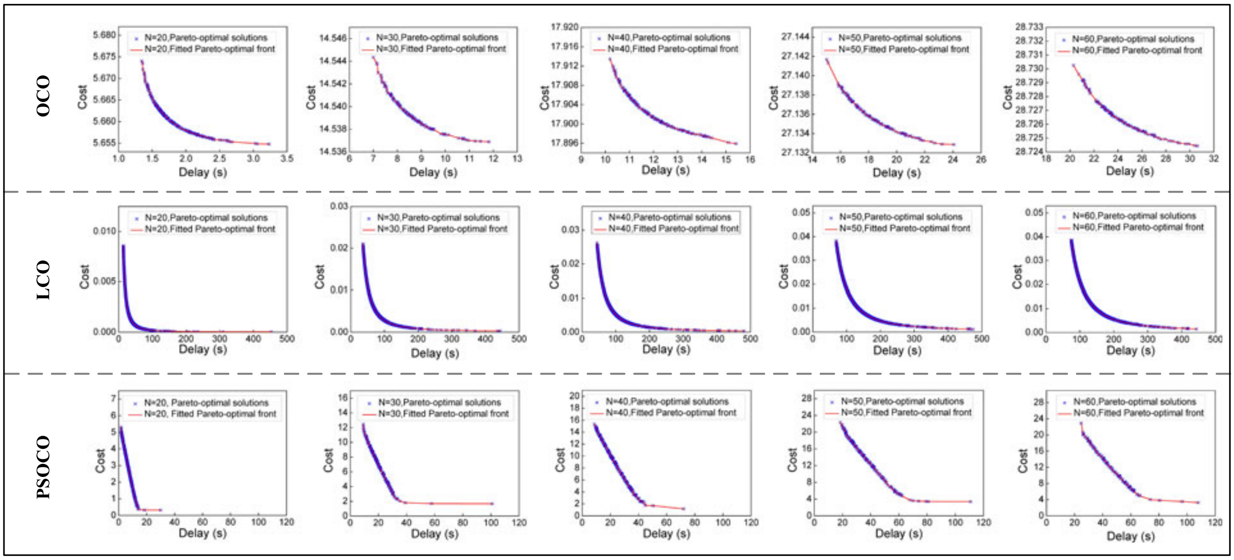


Fig. 3. Pareto-optimal solutions and Pareto-optimal front of OCO, LCO, and PSOCO under different number of vehicles.

5.2.2 Pareto-Optimal Solutions

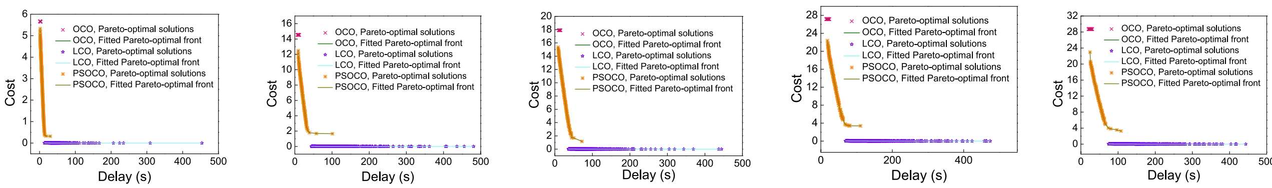
To find the Pareto-optimal solutions and the Pareto-optimal front, we implement the simulations with different number of vehicles in Figs. 3 and 5.

The simulation results of OCO are presented in the top of Figs. 3 and 5a. It can be easily observed from the top of Fig. 3 that the objectives for the delay and the cost run in the opposite direction, which means that the improvement of one objective may lead to the decline of the other. This is because that more communication and computation resources will be allocated to a certain task for transmitting and processing if a lower delay is preferred, which results in a higher cost for the utilization of communication and computation resources, and vice versa. When putting all five curves in the top of Fig. 3 together, we depict Fig. 5a. It is shown that both the delay and cost increase with the increase of N. This is because more vehicles lead to more tasks to be processed, which further results in higher delay and cost due to the limited communication and computation resources of RSU.

The simulation results of LCO are presented in the middle of Figs. 3 and 5b. It is shown from the middle of Fig. 3 that the objectives for the delay and the cost also run in the opposite direction. This is because more computing resources of a certain vehicle will be allocated to process its task if a lower delay is preferred, which will result in a higher cost for the utilization of computing resources, and vice versa. When putting all five curves in the middle of Fig. 3 together, we depict Fig. 5b. It is shown that both the delay and cost increase with the increase of N. The reason is that the delay and cost are the total delays and total costs of all vehicles. The increase of N will increase the total delays and total costs.

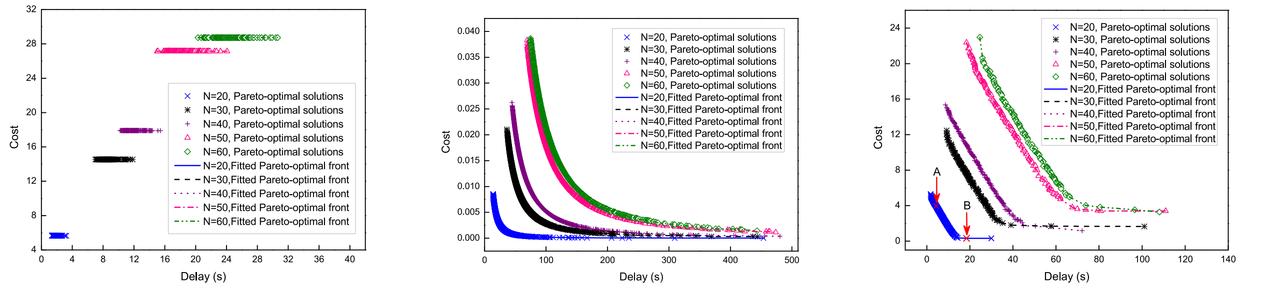
The simulation results of our proposed PSOCO are presented in the bottom of Figs. 3 and 5c. It can be easily observed that the objectives for the delay and the cost also run in the opposite direction. For example, in the second figure in the bottom of Fig. 3, among the Pareto-optimal-solutions, there is a solution with a delay of about 40 s and the cost of about 1.8. If we prefer a lower delay solution, we may find a solution along the Pareto-optimal front curve with a delay of about 20 s, however, the cost of the solution is increased to about 7. The reason is that more tasks will be offloaded to RSU for processing to reduce the processing time due to the higher processing capability of RSU, which may lead to a cost increase due to the cost of uplink communication resource and the cost for RSU to process tasks. When putting all five curves in the bottom of Fig. 3 together, we depict Fig. 5c. It is shown that both the delay and cost increase with the increase of N. The reason is that the communication resource and computing resource of RSU within one road segment are limited, each vehicle will be allocated less communication and computation resources from RSU with the increase of N. Therefore, a higher delay and hence a higher cost will be consumed to complete the tasks of vehicles.

What's more, we depict Fig. 4 to show the performance comparison among OCO, LCO, and PSOCO under different number of vehicles. It is shown that the Pareto-optimal solutions among OCO, LCO, and PSOCO vary greatly. For OCO, the delay is very low while the cost is very high. This



(a) N=20 (b) N=30 (c) N=40 (d) N=50 (e) N=60

Fig. 4. Comparison of Pareto-optimal solutions among OCO, LCO, and PSOCO under different number of vehicles.



(a) Pareto-optimal solutions and Pareto-optimal front of OCO.

(b) Pareto-optimal solutions and Pareto-optimal front of LCO.

(c) Pareto-optimal solutions and Pareto-optimal front of the proposed PSOCO.

Fig. 5. Pareto-optimal solutions and Pareto-optimal front under different schemes.

is because the cost of uplink communication resource and the cost for RSU to process tasks are high. For LCO, the cost is very low while the delay is very high. This is because the limited processing capability of vehicles will result in a severe delay for some compute-intensive tasks since all tasks are processed locally. Different from the OCO and LCO, the tasks can be partially offloaded to RSU in our proposed PSOCO, which jointly allocates the communication and computation resources of vehicles and RSU to obtain the optimal delay and cost. Moreover, the delay and cost of PSOCO are between that of OCO and LCO. A trade-off between delay and cost is also obtained, which can guide the allocation of communication and computation resources for different types of tasks.

5.2.3 Allocation of Communication and Computation

To elaborate on the communication and computation resource allocation, we select two Pareto-optimal solutions, i.e., A and B (hereinafter referred to as $Pareto^A$ and $Pareto^B$, respectively) from Fig. 5c. We first depict the task attribute distribution about c_n and γ_n in Fig. 6. It is shown that 4 tasks are with the processing density of 1000 cycles/bit, 7 tasks are with the processing density of 100 cycles/bit, 9 tasks are with the processing density of 10 cycles/bit. And all γ_n of tasks are randomly distributed between 0 and 0.2. Based on the 20 tasks, we present in the following how the communication and computation resources are allocated under Pareto-optimal solutions A and B.

1) *Allocation of offloading decision* λ_n: For the accuracy of the statistical results, we evaluate the average offloading decision, which is defined as the average λ_n of tasks with the same processing density. Fig. 7 shows that the average offloading decision increases with the increase of processing density. This is because a higher processing density needs more computing cycles, which results in that more task data bits should be offloaded to RSU for processing. More importantly, the average offloading decision of $Pareto^A$ is higher than that of $Pareto^B$, which reflects the preference for delay objective of $Pareto^A$. A lower delay is realized by offloading more task data bits to RSU since the higher computing capability of RSU.

2) *Allocation of local processing capability* f_n^l: For the accuracy of the statistical results, we evaluate the average local processing capability, which is defined as the average f_n^l of tasks with the same processing density. It can be shown from Fig. 8 that the average local processing capability increases with the increase of processing density. This is because more local processing capabilities should be allocated to process

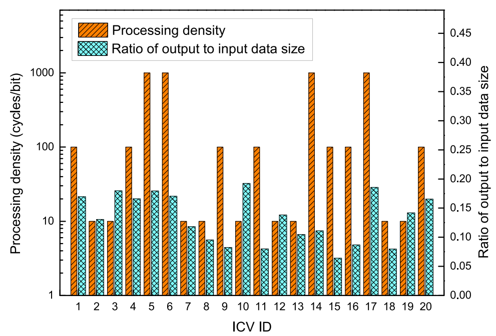


Fig. 6. Task attribute distribution about processing density and ratio of output to input data size.

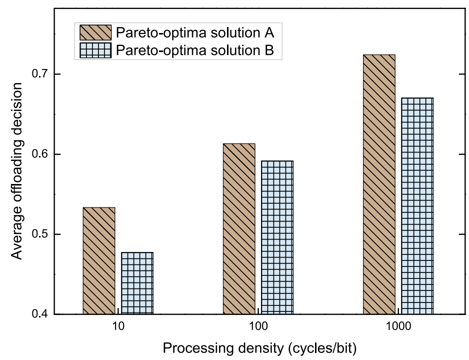


Fig. 7. Average offloading decision under different processing densities.

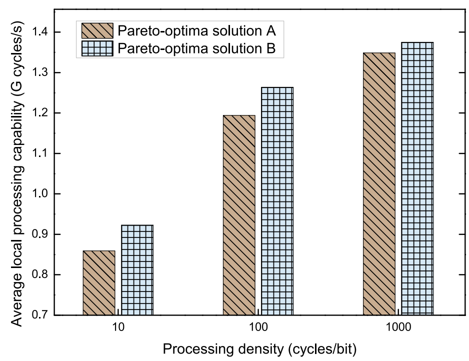


Fig. 8. Average local processing capability under different processing densities.

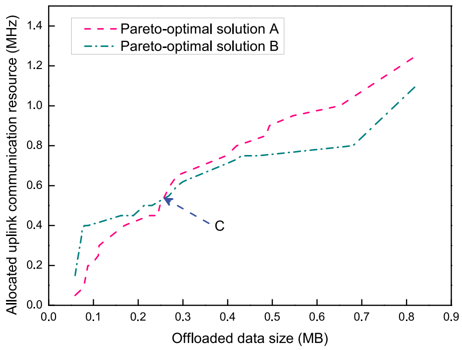


Fig. 9. Allocated uplink communication resource under different offloaded data size.

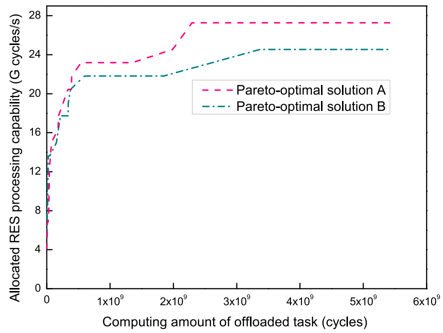


Fig. 10. Allocated RES processing capability under different computing amount of offloaded task.

more task data bits due to the increased processing density to obtain an optimal delay. And the average local processing capability of $Pareto^B$ is higher than that of $Pareto^A$. The reason is that more task data bits are processed locally for $Pareto^B$, with the result that more local processing capabilities are allocated to process the tasks.

3) *Allocation of communication resource*: We evaluate the allocation of communication resource by the allocated uplink communication resource. As shown in Fig. 9, the horizontal axis and vertical axis denote the offloaded data size and allocated uplink communication resource, respectively. The offloaded data size refers to the size of the data that will be offloaded to RSU for processing. It shows that the allocated uplink communication resource increases with the increase of offloaded data size. What's more, less communication resource is allocated to transmit offloading tasks for $Pareto^A$ when the offloaded data size is small. When the offloaded data size reaches a threshold, as point C in Fig. 9, the allocated communication resource of $Pareto^A$ is higher than that of $Pareto^B$. This is because the local computing capabilities of vehicles are enough to process the tasks when data size is small. Offloading tasks may cause extra delay, which is against the purpose of $Pareto^A$. With the increase of the offloaded data size, more communication resource is allocated to transmit the tasks to RSU. This is because the RES enabled RSU has powerful computing capability toward a lower processing delay, which is consistent with the purpose of $Pareto^A$.

4) *Allocation of RES computation resource*: Fig. 10 reflects the relationship between the allocated RES processing capability and the computing amount of offloaded tasks for Pareto-optimal solutions A and B. It is shown that the allocated RES processing capability increases with the increase of the computing amount of offloaded task. And the allocated RES processing capability of $Pareto^A$ is higher than that of $Pareto^B$. The reason is that the purpose of $Pareto^A$ is obtaining a lower delay in task processing, which needs more RES processing capabilities to achieve this goal. On the contrary, since the $Pareto^B$ prefers a low cost, less RES processing capabilities should be allocated due to the high cost of RES resource.

6 Conclusion

In this paper, we have investigated the computation offloading problem in a VEC network through jointly considering the allocation of communication and computation resources, aiming at providing a detailed analysis of the delay and cost of computation offloading for VEC and minimizing the delay and cost from the perspective of multi-objective optimization. To this end, we first established an offloading framework with communication and computation for VEC, where tasks with different requirements for computation capability are considered. To consider comprehensive performance, we then formulated a multi-objective optimization problem to minimize both the delay and cost during computation offloading. To solve the formulated optimization problem, which is also a MINLP problem, we introduced the concept of *Pareto optimality* and proposed a particle swarm optimization based computation offloading (PSOCO) algorithm to obtain the Pareto-optimal solutions. Finally, based on the real-world vehicular trace, we implemented extensive simulation to verify the performance of PSOCO. Moreover, we also presented a comprehensive analysis and discussion on the relationship between delay and cost among the Pareto-optimal solutions.

In the future, we will consider continuous tasks, the mobility of vehicles, and the handover between different RSUs for a more practical VEC scenario. And we will consider developing real prototypes to conduct field tests of the proposed algorithm.

Acknowledgments

This work was supported in part by the scholarship from China Scholarship Council, in part by the National Natural Science Foundation of China under Grant U1801266.

References

[1] Y. Dai, D. Xu, S. Maharjan, G. Qiao, and Y. Zhang, "Artificial intelligence empowered edge computing and caching for internet of vehicles," *IEEE Wireless Commun.*, vol. 26, no. 3, pp. 12–18, Jun. 2019.
[2] S. Sharma and B. Kaushik, "A survey on internet of vehicles: Applications, security issues & solutions," *Veh. Commun.*, vol. 20, 2019, Art. no. 100182.
[3] H. Zhou, W. Xu, J. Chen, and W. Wang, "Evolutionary V2X technologies toward the internet of vehicles: Challenges and opportunities," *Proc. IEEE*, vol. 108, no. 2, pp. 308–323, Feb. 2020.

[4] J. E. Siegel, D. C. Erb, and S. E. Sarma, "A survey of the connected vehicle landscape—architectures, enabling technologies, applications, and development areas," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 8, pp. 2391–2406, Aug. 2018.

[5] J. Du, F. R. Yu, X. Chu, J. Feng, and G. Lu, "Computation offloading and resource allocation in vehicular networks based on dual-side cost minimization," *IEEE Trans. Veh. Technol.*, vol. 68, no. 2, pp. 1079–1092, Feb. 2019.

[6] P. Nelson, "Just one autonomous car will use 4000 GB of data/day," 2016. [Online]. Available: https://www.networkworld.com/article/3147892/one-autonomous-car-will-use-4000-gb-of-dataday.html

[7] B. Johnstone, "How semiconductor IP is at the heart of generating autonomous vehicle value," 2019. [Online]. Available: https://y19.mwcshanghai.com/wp-content/uploads/2019/07/Bryce-Johnstone.pdf

[8] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, "Joint load balancing and offloading in vehicular edge computing and networks," *IEEE Internet of Things J.*, vol. 6, no. 3, pp. 4377–4387, Jun. 2019.

[9] L. Liu, C. Chen, Q. Pei, S. Maharjan, and Y. Zhang, "Vehicular edge computing and networking: A survey," *Mobile Netw. Appl.*, 2020. https://doi.org/10.1007/s11036-020-01624-1

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

[11] H. Zhou *et al.*, "Whitefi infostation: Engineering vehicular media streaming with geolocation database," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 8, pp. 2260–2274, Aug. 2016.

[12] Y. Liu, S. Wang, J. Huang, and F. Yang, "A computation offloading algorithm based on game theory for vehicular edge networks," in *Proc. IEEE Int. Conf. Commun.*, 2018, pp. 1–6.

[13] K. Zhang, Y. Mao, S. Leng, S. Maharjan, and Y. Zhang, "Optimal delay constrained offloading for vehicular edge computing networks," in *Proc. IEEE Int. Conf. Commun.*, 2017, pp. 1–6.

[14] 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.

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

[16] Y. Censor, "Pareto optimality in multiobjective problems," *Appl. Math. Optim.*, vol. 4, no. 1, pp. 41–59, 1977.

[17] K. Zhang *et al.*, "Energy-efficient offloading for mobile edge computing in 5G heterogeneous networks," *IEEE Access*, vol. 4, pp. 5896–5907, 2016.

[18] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Trans. Wireless Commun.*, vol. 16, no. 3, pp. 1397–1411, Mar. 2017.

[19] F. Wang, J. Xu, X. Wang, and S. Cui, "Joint offloading and computing optimization in wireless powered mobile-edge computing systems," *IEEE Trans. Wireless Commun.*, vol. 17, no. 3, pp. 1784–1797, Mar. 2018.

[20] T. Q. Dinh, Q. D. La, T. Q. Quek, and H. Shin, "Learning for computation offloading in mobile edge computing," *IEEE Trans. Commun.*, vol. 66, no. 12, pp. 6353–6367, Dec. 2018.

[21] M.-A. Messous, H. Sedjelmaci, N. Houari, and S.-M. Senouci, "Computation offloading game for an UAV network in mobile edge computing," in *Proc. IEEE Int. Conf. Commun.*, 2017, pp. 1–6.

[22] T. Q. Dinh, J. Tang, Q. D. La, and T. Q. Quek, "Offloading in mobile edge computing: Task allocation and computational frequency scaling," *IEEE Trans. Commun.*, vol. 65, no. 8, pp. 3571–3584, Aug. 2017.

[23] Y. Kim, J. Kwak, and S. Chong, "Dual-side optimization for cost-delay tradeoff in mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 67, no. 2, pp. 1765–1781, Feb. 2018.

[24] C. Wang, C. Liang, F. R. Yu, Q. Chen, and L. Tang, "Computation offloading and resource allocation in wireless cellular networks with mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 16, no. 8, pp. 4924–4938, Aug. 2017.

[25] L. Lin, X. Liao, H. Jin, and P. Li, "Computation offloading toward edge computing," *Proc. IEEE*, vol. 107, no. 8, pp. 1584–1607, Aug. 2019.

[26] Z. Ding, Z. Yang, P. Fan, and H. V. Poor, "On the performance of non-orthogonal multiple access in 5G systems with randomly deployed users," *IEEE Signal Process. Lett.*, vol. 21, no. 12, pp. 1501–1505, Dec. 2014.

[27] H. Peng, Q. Ye, and X. Shen, "Spectrum management for multi-access edge computing in autonomous vehicular networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 7, pp. 3001–3012, Jul. 2020.

[28] Y. Mao, J. Zhang, S. Song, and K. B. Letaief, "Power-delay tradeoff in multi-user mobile-edge computing systems," in *Proc. IEEE Global Commun. Conf.*, 2016, pp. 1–6.

[29] B. He, A. Liu, N. Yang, and V. K. Lau, "On the design of secure non-orthogonal multiple access systems," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 10, pp. 2196–2206, Oct. 2017.

[30] P. D. Diamantoulakis, K. N. Pappi, G. K. Karagiannidis, H. Xing, and A. Nallanathan, "Joint downlink/uplink design for wireless powered networks with interference," *IEEE Access*, vol. 5, pp. 1534–1547, 2017.

[31] L. Lei, D. Yuan, C. K. Ho, and S. Sun, "Joint optimization of power and channel allocation with non-orthogonal multiple access for 5G cellular systems," in *Proc. IEEE Global Commun. Conf.*, 2015, pp. 1–6.

[32] M.-H. Chen, M. Dong, and B. Liang, "Resource sharing of a computing access point for multi-user mobile cloud offloading with delay constraints," *IEEE Trans. Mobile Comput.*, vol. 17, no. 12, pp. 2868–2881, Dec. 2018.

[33] H. Jia, Y. Lin, Q. Luo, Y. Li, and H. Miao, "Multi-objective optimization of urban road intersection signal timing based on particle swarm optimization algorithm," *Advances Mech. Eng.*, vol. 11, no. 4, 2019, Art. no. 1687814019842498.

[34] F. Rudzinski, "Finding sets of non-dominated solutions with high spread and well-balanced distribution using generalized strength pareto evolutionary algorithm," in *Proc. Conf. Int. Fuzzy Syst. Assoc. Eur. Soc. Fuzzy Logic Technol.*, 2015, pp. 178–185.

[35] S.-Y. Pyun, W. Lee, and D.-H. Cho, "Resource allocation for vehicle-to-infrastructure communication using directional transmission," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 4, pp. 1183–1188, Apr. 2016.

[36] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," *IEEE Trans. Evol. Comput.*, vol. 6, no. 1, pp. 58–73, Feb. 2002.

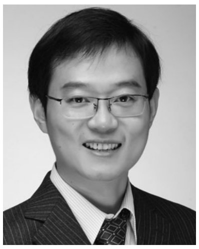
[37] R. Brits, A. P. Engelbrecht, and F. van den Bergh, "Locating multiple optima using particle swarm optimization," *Appl. Math. Comput.*, vol. 189, no. 2, pp. 1859–1883, 2007.

[38] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*. Hoboken, NJ, USA: Wiley, 2001.

[39] R. Hassan, B. Cohanim, O. De Weck, and G. Venter, "A comparison of particle swarm optimization and the genetic algorithm," in *Proc. 46th AIAA/ASME/ASCE/AHS/ASC Structures Structural Dyn. Materials Conf.*, 2005, Art. no. 1897.

[40] Didi, "Urban traffic time index and trajectory data (new)," 2018. [Online]. Available: https://gaia.didichuxing.com

Quyuan Luo received the PhD degree in communication and information system from Xidian University, Xi'an, China, in 2020. He had been a visiting scholar with computer science, Wayne State University, USA from 2019 to 2020. He is currently an assistant professor with the School of Information Science and Technology, Southwest Jiaotong University. His current research interests include intelligent transportation systems, content distribution, edge computing and resource allocation in vehicular networks.



Changle Li (Senior Member, IEEE) received the PhD degree in communication and information system from Xidian University, Xi'an, China, in 2005. He conducted his postdoctoral research with Canada and the National Institute of information and Communications Technology, Japan, respectively. He had been a visiting scholar with the University of Technology Sydney and is currently a professor with the State Key Laboratory of Integrated Services Networks, Xidian University. His research interests include intelligent transportation systems, vehicular networks, mobile ad hoc networks, and wireless sensor networks.

Tom H. Luan (Senior Member, IEEE) received the BE degree from Xi'an Jiaotong University, Xi'an, China, in 2004, the MPhil degree from the Hong Kong University of Science and Technology, Kowloon, Hong Kong, in 2007, and the PhD degree from the University of Waterloo, Waterloo, ON, Canada, in 2012. He has been a lecture with the School of Information Technology, Deakin University, Burwood, VIC., Australia from 2013 to 2017 and is currently a professor with the School of Cyber Engineering, Xidian University. His current research interests include the content distribution in vehicular networks, mobile cloud computing, and fog computing.

Weisong Shi (Fellow, IEEE) received the BS degree in computer engineering from Xidian University, Xi'an, China, in 1995, and the PhD degree in computer engineering from the Chinese Academy of Sciences, Beijing, China, in 2000. He is a Charles H. Gershenson distinguished faculty fellow and a professor of computer science with Wayne State University, Detroit, MI, USA. His current research interests include edge computing, computer systems, energy-efficiency, and wireless health. He was a recipient of the National Outstanding PhD Dissertation Award of China and the NSF CAREER Award. He is an ACM distinguished scientist.