IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 16, 2018, accepted January 12, 2019, date of publication January 18, 2019, date of current version February 8, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2893486

Task Scheduling for Smart City Applications Based on Multi-Server Mobile Edge Computing

YIQIN DENG[1], (Member, IEEE), ZHIGANG CHEN[1,2], (Member, IEEE), XIN YAO[2], (Member, IEEE), SHAHZAD HASSAN[1], AND JIA WU[2], (Member, IEEE)

[1]School of Information Science and Engineering, Central South University, Changsha 410083, China
[2]School of Software, Central South University, Changsha 410075, China

Corresponding author: Zhigang Chen (czg@csu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61672540, in part by the National Natural Science Foundation of China through the Major Program under Grant 71633006, and in part by the Mobile Health Ministry of Education-China Mobile Joint Laboratory.

ABSTRACT The smart city is increasingly gaining worldwide attention. It has the potential to improve the quality of life in convenience, at work, and in safety, among many others' utilizations. Nevertheless, some of the emerging applications in the smart city are computation-intensive and time-sensitive, such as real-time vision processing applications used for public safety and the virtual reality classroom application. Both of them are hard to handle due to the quick turnaround requirements of ultra-short time and large amounts of computation that are necessary. Fortunately, the abundant resource of the Internet of Vehicles (IoV) can help to address this issue and improve the development of the smart city. In this paper, we focus on the problem that how to schedule tasks for these computation-intensive and time-sensitive smart city applications with the assistance of IoV based on multi-server mobile edge computing. Task scheduling is a critical issue due to the limited computational power, storage, and energy of mobile devices. To handle tasks from the aforementioned applications in the shortest time, this paper introduces a cooperative strategy for IoV and formulates an optimization problem to minimize the completion time with a specified cost. Furthermore, we develop four evolving variants based on the alternating direction method of multipliers (ADMM) algorithm to solve the proposed problem: variable splitting ADMM, Gauss–Seidel ADMM, distributed Jacobi ADMM, and distributed improved Jacobi (DIJ)-ADMM algorithms. These algorithms incorporate an augmented Lagrangian function into the original objective function and divide the large problem into two sub-problems to iteratively solve each sub-problem. The theoretical analysis and simulation results show that the proposed algorithms have a better performance than the existing algorithms. In addition, the DIJ-ADMM algorithm demonstrates optimal performance, and it converges after approximately ten iterations and improves the task completion time and offloaded tasks by 89% and 40%, respectively.

INDEX TERMS Task scheduling, smart city, mobile edge computing, Internet of Vehicle, alternating direction method of multipliers (ADMM) algorithm.

I. INTRODUCTION

Smart city is an urban area that manages available resources more efficiently by collecting electronic data from citizens, devices, and assets. It is increasingly gaining worldwide attention. Statistics show that global spending on smart cities reached 14.85 billion U.S. dollars in 2015 and is forecasted to increase to 34.35 billion U.S. dollars in 2020 [1]. Smart city has the potential to improve the quality of life in convenience, at work, and in safety, among many others utilizations [2], [3]. However, some of the emerging applications in smart city are computation-intensive and time-sensitive, which means they require many computation resources within a very short completion time. Some of smart city applications even require a response in 2 milliseconds [4], [5]. For instance, the ''intelligent'' policing application in smart city requires real-time vision processing near the cameras to ensure public safety [7], and virtual reality classroom application [6].

Fortunately, the abundant resource of IoV can help to address the aforementioned issues and improve the development of the smart city. The Gartner forecasts that there will be 250+ million smart cars connected to high-tech networks

2169-3536 © 2019 IEEE. Translations and content mining are permitted for academic research only.
Personal use is also permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

by 2020 [8]. At the same time the computational capability of one Nvidia automatous vehicle Drive PX 2 is roughly the same as 150 MacBook Pros [9]. Academic and government environments are often interested in the technology to build smart city [10], [11]. Ding and Fang [10] design a data transportation scheme based on the vehicular temporary storage to support urban sensing services in smart cities. The U.S. Department of Transportation advocates leveraging the connected and autonomous vehicles to improve the integrated corridor management and operations in smart city [11]. Incorporating the IoV into the facilitation of a smart city can not only sharply reduce the cost of physical infrastructures, but also utilize reasonable idle resources of vehicles, which is also feasible from a future perspective.

To the best of our knowledge, this paper is the first work focusing on the problem that *how to schedule tasks for computation-intensive and time-sensitive smart city applications with the assistance of IoV based on multi-server mobile edge computing*. Task scheduling in this problem refers to selecting the right components in the IoV to complete tasks with the shortest completion time with the constrained cost, e.g., $min\ \mathcal{CT}(a_{n\bullet})$ $constrained\ \mathcal{E}_n^{total}(a_{n\bullet})$. Intuitively, to solve this problem, we need to seek a tradeoff between the completion time and the cost of tasks. These tasks refer to computation-intensive, time-sensitive and independent tasks, such as the real-time vision processing task and view rendering task in applications previously mentioned.

Task scheduling is a critical issue due to the limited computational power, storage and energy of mobile devices. It can improve computing efficiency, reduce task completion time, and utilize idle resources from other devices in the system [13], [14]. Task scheduling has been studied extensively in wireless networks [15]–[19]. In our recent work, we have developed a dynamic task scheduling algorithm for single-hop Fog-IoT architecture based on the Lyapunov optimization technique [15]. Chen and Hao [16] present an efficient task scheduling strategy to minimize the latency and extend the battery life of user's equipment for software defined ultra-dense network. However, all of these solutions consider a single edge server and are still not sufficient to handle the multi-server MEC environment with the support of IoV for smart city applications.

In our proposed situation, there are still some challenges to address task scheduling for smart city applications: 1) it is challenging to design cooperative algorithms to obtain resources from the IoV, including computation, storage and communication resources; 2) it is difficult to handle the problem of completing the task from computation-intensive and time-sensitive applications in an ultra-short time with a low time complexity algorithm.

In this paper, we design a system mainly consisting of three parts: cognitive radio router enabled vehicles (CRVs), cognitive radio capable roadside units (CRSUs), and the task publisher. To complete the task from the computation-intensive and time-sensitive applications utilizing the abundant resources of the IoV, we formulate an optimization problem to minimize the completion time with a specified cost. Additionally, we propose four evolving variants based on the ADMM algorithm to solve the proposed problem: VS-ADMM, GS-ADMM, DJ-ADMM and DIJ-ADMM algorithm. The ADMM algorithm is a mature and effective computational method for solving distributed optimization problems [20]. It introduces an augmented Lagrangian function and divides a large problem into two sub-problems to iteratively solve each sub-problem. Nevertheless, we cannot directly solve the proposed problem with more than two variables by using the ADMM algorithm. Thus, we introduce the VS-ADMM algorithm by applying the splitting variable technique into the ADMM algorithm. Then, in order to improve the effectiveness of the VS-ADMM algorithm when the number of nodes is large, we develop a GS-ADMM algorithm, but it does not update the variables parallelly. Therefore, a DJ-ADMM algorithm is put forward. The DIJ-ADMM algorithm further improves performance by injecting a proximal term and a damping parameter.

Furthermore, we analyze the performance of each algorithm thoroughly and prove that the DIJ-ADMM algorithm has the lowest time complexity and the best performance. Especially, the DIJ-ADMM algorithm has the capability to preserve privacy for users by solving the independent sub-problems. Simulation results show that the DIJ-ADMM algorithm can converge after approximately 10 iterations and improve task completion time and offloaded tasks by 89% and 40%, respectively. In short, the specific contributions of this paper are as follows:

- We first consider task scheduling for computation-intensive and time-sensitive smart city applications with the assistance of IoV based on multi-server mobile edge computing; and formulate an optimization problem to minimize the completion time with a specified cost.
- We design four evolving algorithms based on the existing alternating direction method of multipliers algorithm and analyze the advantages and disadvantages and scope of application of these four algorithms.
- Further, we theoretically analyze the optimality and complexity of the DIJ-ADMM algorithm and prove that it protects the privacy of the user in the IoV.
- Through implementing the proposed four algorithms, we find that they significantly reduce task completion time while increasing offloaded tasks and the DIJ-ADMM algorithm possesses the optimal performance with the best convergence rate $o(1/t)$, where t is the number of time slots in the network.

The rest of the paper is organized as follows: Section II describes related works. System model and problem formulation are presented in Section III. In Section IV, we propose four algorithms to solve the completion time minimization problem. The performance of the proposed algorithms is analyzed in Section V. Finally, extensive simulation results and conclusions are provided in Sections VI and VII, respectively.

II. RELATED WORK

Task scheduling is a classical method that involves transferring tasks to the external platform due to the limited computational power, storage and energy of the mobile device [21]. It can improve computing efficiency, reduce task completion time, and utilize resources efficiently from other devices in the system, and it has been extensively investigated in wireless networks [15]–[19], [22]–[24], [26]. In our recent work [15], we have developed a workload dynamic scheduling algorithm for single-hop Fog-IoT architecture based on Lyapunov optimization technique. It makes a trade-off between the optimal throughput utility and worst-case delay. Chen and Hao [16] present an efficient task scheduling strategy to minimize the latency and extend battery life of user's equipment for software defined ultra-dense network. Zhang *et al.* [24] propose a task scheduling algorithm to minimize the maximum tolerable delay in mobile edge computing. Chen *et al.* [26] introduce the Edge-CoCaCo, a task scheduling solution by jointly considering computation, caching, and communication on the edge cloud.

Mobile Edge Computing (MEC) is an emerging technology which brings computation and storage close to the client (e.g., user's personal computers, IoT devices and routers) as much as possible to minimize communication latency and bandwidth utilization. MEC is a supplement to cloud computing, not an alternative [27]. With the emergence of significant amounts of computation-intensive and time-sensitive applications, MEC is becoming increasingly significant in all fields. In some Virtual Reality (VR) scenarios, it makes much steeper requirements of views due to distance between user eyes and the screen of head-mounted device is very short. However, traditional approaches that uploading all the views to the cloud center for rendering and streaming will not only take a long time and occupy a large bandwidth, but also harms the user experience. Thus, a novel wireless VR strategy is implemented based on edge computing [6]. In the traditional video surveillance application, videos generated in cameras are sent to the cloud center for processing. Nevertheless, with the ubiquity of cameras and high usage of bandwidth, the amount of video that can be transmitted to the cloud center is limited. Zhang *et al.* [12] design and implement a real-time distributed wireless video surveillance system by leveraging MEC to preprocess videos, and thus greatly improve the coverage area and relevant objects. To address the big data in IoV, Zhang *et al.* [24] develop the regional cooperative fog-computing-based intelligent vehicular network architecture. By incorporating cognitive computing into edge computing, Chen *et al.* [28], [29] propose the edge cognitive computing (ECC) paradigm and a smart-healthcare system based on the ECC. To control the traffic intelligently, Chen *et al.* [30] design the LLTC algorithm based on label-less learning and edge computing.

Task scheduling is increasingly becoming the focus of MEC due to the development of computation-intensive and time-sensitive applications [31]. Mao *et al.* [17] develop a dynamic task scheduling method for a MEC system with an energy harvest mobile device and an edge server. Hao *et al.* [18] propose an energy efficient task caching and scheduling algorithm for a mobile device and an edge cloud MEC architecture based on the alternating iterative algorithm. Chen *et al.* [32] investigate multi-user task scheduling problem in MEC with multi-channel wireless interference and give a distributed solution based on a game theoretic approach. Nevertheless, to the best of our knowledge, almost all works only consider the scenario of a single edge server except studies in [16] and [33]. However, works in [16] and [33] do not consider to complete tasks in the shortest time via utilizing the resources of the IoV, which is critical to handle tasks in the computation-intensive and time-sensitive smart city applications.

Alternating direction method of multipliers (ADMM) algorithm is a mature and effective computational method for solving optimization problems, especially for distributed convex optimization problems [34]. It decomposes large problems into multiple small, easy-to-solve local sub-problems through the decomposition-coordination process, and obtains solutions to large problems by coordinating the solutions of sub-problems. Boyd *et al.* [20] review and prove that it is very suitable for large-scale distributed optimization problems. However, the limitations of ADMM algorithm start to bring out when it deals with complicated scenarios. For instance, ADMM can only be used to solve optimization problems without inequality restrictions and with two variables [20]. Additionally, it is a centralized optimization method that requires private information for each node [35].

Considering the limitations of traditional ADMM algorithm and existing works in task scheduling methods, this paper designs task scheduling for MEC-supported smart city applications and develops several novel task scheduling optimization algorithms based on the classical ADMM algorithm. In addition, we analyze strengths and weaknesses of each algorithm.

III. SYSTEM MODEL AND PROBLEM FORMULATION

In this section, we introduce the system model and mathematically define the new metrics of the network: completion time and cost. Then we give a definition of the proposed problem and provide a formulation for the problem.

A. OVERVIEW

As shown in Figure 1, we use the real-time vision processing application for public safety as the typical computation-intensive and time-sensitive application scenario in smart city. Follow the same line in [36], we assume that vehicles are equipped with powerful communication devices, called cognitive radio routers (CR routers), and one vehicle can discover other vehicles via user datagram protocol broadcasting messages. The task of real-time vision processing comes from smart cameras (task publisher in this scenario) in this application and then it is published on the platform consisting of the cognitive radio capable roadside units (CRSUs) and cognitive radio router enabled vehicles (CRVs).

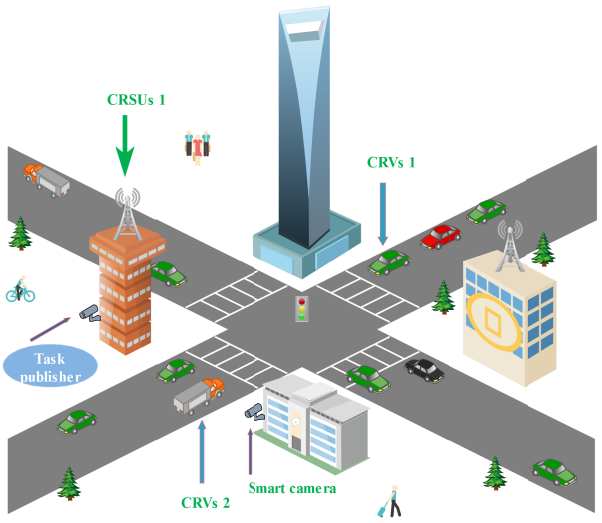

FIGURE 1. An illustration of system model.

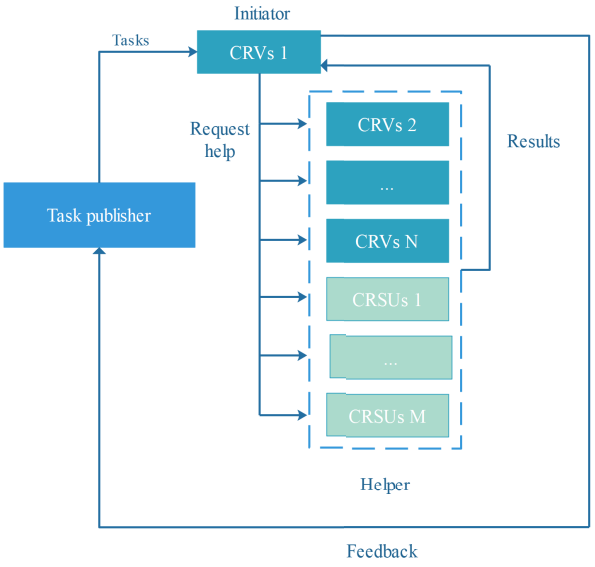

FIGURE 2. The comprehensive process of task.

TABLE 1. Main symbols and their meanings.

Symbol	Meaning
$\mathcal{N}$	number of CRVs
$\mathcal{M}$	number of CRSUs
$\mathcal{CT}$	completion time of task
ν_n	arrival rate of tasks in the vehicle n
μ_n	maximum processing capacity per unit time
s_{nm}	transmission time from the initiator to CRSUs
$\mathcal{E}_n^{total}$	total cost of completing the task

After identifying behaviors in the vision with the cooperation of the CRSUs and CRVs, the task publisher obtains the feedback from them and chooses to generate alerts or not in real-time. In short, this system mainly consists of the following three parts.

- CRVs: One part is composed of $n \in \{1, 2, \cdots, \mathcal{N}\}$ cognitive radio router enabled vehicles (CRVs), which are regarded as stationary at each time slot $t \in \{1, 2, \cdots, \mathcal{T}\}$. CRVs refer to vehicles equipped with CR routers as their communication devices [36].
- CRSUs: The second part includes $m \in \{1, 2, \cdots, \mathcal{M}\}$ cognitive radio capable roadside units (CRSUs). The CRVs and CRSUs are connected through cognitive radio, thus we can ignore the bandwidth resources of the connections between the CRSUs and CRVs since there are a wide range of under-utilized spectrum resources. Since that the scenario we consider is the urban area, the installation cost of CRSUs is high and the number of CRSUs is much smaller than the number of CRVs, e.g., $\mathcal{M} \ll \mathcal{N}$. In addition, both the CRSUs and CRVs have the capability of communication and computation. The CRSUs and CRVs are both equivalent to edge servers, but CRSUs' power of communication and computing is much greater than that of the CRVs.
- Task publisher: The third part is the task publisher, which refers to the smart cameras in this scenario. It produces these tasks by monitoring behaviors of citizens in urban areas and then publishes these tasks on the platform, which consists of CRSUs and CRVs. After identifying behaviors in the vision with the cooperation of the CRSUs and CRVs, the task publisher obtains the feedback from them and chooses to generate alerts or not in real-time. In addition, task publisher pays a reward to the initiator CRVs.

Figure 2 shows the procedure of processing the task in this scenario. First, an initiator CRVs accepts the task published by the task publisher and then divides these tasks into subtasks. Then, it will request help from the surrounding CRSUs and CRVs. After completing this task cooperatively, the initiator CRVs aggregates results from helper CRSUs and CRVs and pays reward (described as cost in section III.C) for them. Finally, the initiator CRVs send feedback to the task publisher and obtain the money. Specifically, in our system, each CRVs can act as the initiator and helper for the task. The meanings of main symbols in this paper are summarized as Table 1.

In the following section, we propose the model of completion time and cost of the task, and develop the problem formulation to complete the task in the shortest time with the specified cost constraint.

B. COMPLETION TIME

To meet the ultra-short time requirement of the tasks, we define a new metric called the completion time. It refers to the total time of completing the whole task.

In this system, initiator CRVs can divide the task into three parts: α_{nn}, α_{nl} and α_{nm}, which are processed in its own computing system, other helper CRVs and CRSUs, respectively, e.g., $\alpha_{n\bullet} = \{\alpha_{nn}, \alpha_{nl}, \alpha_{nm}\}$. Recall that the CRSUs is an edge server with much more computing power than the CRVs, thus, we ignore the time spent by the CRSUs processing tasks as shown in many previous works. Let ν_n, s_{nm}, and μ_n be the

arrival rate of tasks in the CRVs n, the transmission time of the computation task from the initiator CRVs n to the CRSUs m, and the maximum processing capacity per unit time of the CRVs n, respectively. Therefore, the task completion time ($\mathcal{CT}$) of the initiator CRVs n is mainly made up of the following three parts: the time initiator CRVs handles by itself, the time that both the helper CRVs process and the CRSUs processes. $\mathcal{CT}$ is denoted as

$$\mathcal{CT}(\alpha_{n\bullet}) = \frac{1}{\sum_{n\in\mathcal{N}} \nu_n}[\alpha_{nn}(\frac{1}{\mu_n - \alpha_{nn}}) + \sum_{n\in\mathcal{N}}\sum_{\iota\in\mathcal{N}\setminus\{n\}} \alpha_{n\iota} \times(s_{n\iota} + \frac{1}{\mu_\iota - \sum\alpha_{n\iota}})] + \sum_{n\in\mathcal{N}}\sum_{m\in\mathcal{N}\setminus\{n\}} \alpha_{nm}s_{nm}. \tag{1}$$

where the following restrictions should hold:

$$\alpha_{nn} + \sum_{\iota\in\mathcal{N}\setminus\{n\}} \alpha_{n\iota} + \sum_{m\in\mathcal{N}\setminus\{n\}} \alpha_{nm} = \nu_n. \tag{2}$$

C. COST

To obtain the computation, storage, and communication resources of the CRVs and CRSUs, the initiator CRVs pays a reward to the helper CRVs and CRSUs. This section defines a new metric which is the cost.

Cost is defined as the total money expenditure for completing the task, including the cost incurred by the CRSUs (e.g., $\mathcal{E}_{nm}$) and CRVs involved in completing the task (e.g., $\mathcal{E}_{n\iota}$). Thus, the cost of initiator CRVs n for completing the task is

$$\mathcal{E}_n^{total}(\alpha_{n\bullet}) = \sum_{n\in\mathcal{N}}\sum_{\iota\in\mathcal{N}\setminus\{n\}} \alpha_{n\iota}\mathcal{E}_{n\iota} + \sum_{n\in\mathcal{N}}\sum_{m\in\mathcal{N}\setminus\{n\}} \alpha_{nm}\mathcal{E}_{nm}. \tag{3}$$

Let Φ_n be the upper limit of the cost that initiator CRVs n permits, which is related to the reward ς that the initiator CRVs n can obtain from the task publisher, e.g., $\Phi_n = \epsilon\varsigma$. Here, ϵ is a coefficient and $\epsilon \in [0, 1]$, which depends on the pure benefit expected by the initiator CRVs n. Furthermore, the cost cannot exceed the upper limit of the cost that the initiator CRVs can accept. That is,

$$\mathcal{E}_n^{total}(\alpha_{n\bullet}) \le \Phi_n, \quad \forall n \in \mathcal{N}. \tag{4}$$

Recall that the installing cost of CRSUs is high, thus the cost required to utilize the CRSUs' resources is also much greater than that of the CRV resources, i.g., $\mathcal{E}_{nm} \gg \mathcal{E}_{n\iota}$. Considering both section III-B and section III-C, we know that if we want to exploit the resources of the CRVs and CRSUs to complete the computation task in the shortest time, we tend to transfer as many tasks as possible to the CRSUs due to its powerful computation capability. Nevertheless, the cost will increase greatly, which may exceed the permission of the initiator CRV. In the next section, we will define and formulate this problem to make a tradeoff between the completion time and the cost.

D. PROBLEM FORMULATION

According to the previous definition of completion time and cost, we define the ST-SC problem as follows.

Definition 1 (ST-SC Problem): In this paper, we focus on the problem of how to schedule tasks for computation-intensive and time-sensitive smart city applications (ST-SC problem) with the assistance of IoV based on multi-server mobile edge computing. These tasks are independent, such as the real-time vision processing task and view rendering task in applications as we mentioned previously. ST-SC problem refers to selecting the optimal policy $\mathcal{P} = \{\alpha_{n\bullet}\}, n \in \mathcal{N}$, which means choosing the right helper CRVs $\alpha_{n\iota}$ and CRSUs α_{nm} to complete the task in the shortest completion time with a constrained cost.

ST-SC problem can be mathematically formulated as follows:

$$\begin{aligned} &\min_{\alpha_{1\bullet},\cdots,\alpha_{n\bullet}} \sum_{n=1}^{\mathcal{N}} \mathcal{CT}(\alpha_{n\bullet}) \\ &s.t.\ \mathcal{E}_n^{total}(\alpha_{n\bullet}) \le \Phi_n, \quad \forall n \in \mathcal{N}, \\ &\alpha_{nn} + \sum_{\iota\in\mathcal{N}\setminus\{n\}} \alpha_{n\iota} + \sum_{m\in\mathcal{N}\setminus\{n\}} \alpha_{nm} = \nu_n. \end{aligned} \tag{5}$$

IV. FOUR DISTRIBUTED TASK SCHEDULING ALGORITHMS FOR ST-SC PROBLEM

In this section, we will give an analysis for the ST-SC problem and then propose four evolving task scheduling algorithms for it.

A. PROBLEM ANALYSIS

As we know from the previous discussion, it is critical to decide the optimal policy for the ST-SC problem, but it seems like a complex problem due to a large $\mathcal{N}$ and the requirement of processing parallelly in the context of this paper.

Intuitively, we can use the alternating direction method of multipliers (ADMM) algorithm to solve the ST-SC problem. Then, let us briefly review the ADMM algorithm. For the following general convex optimization problem,

$$\begin{aligned} &\min_{x_1,x_2,\cdots,x_N} \sum_{i=1}^{\mathcal{N}} f_i(x_i) \\ &s.t. \sum_{i=1}^{\mathcal{N}} \mathcal{A}_i x_i = c. \end{aligned} \tag{6}$$

The ADMM algorithm gives the augmented Lagrangian form of the problem (6) as follows:

$$\mathcal{L}_\rho(x_1 \cdots, x_\mathcal{N}, \varphi) = \sum_{i=1}^{\mathcal{N}} f_i(x_i) - \varphi^{\mathrm{T}}\left(\sum_{i=1}^{\mathcal{N}} \mathcal{A}_i x_i - c\right) + \frac{\rho}{2}\left\|\sum_{i=1}^{\mathcal{N}} \mathcal{A}_i x_i - c\right\|_2^2. \tag{7}$$

where ρ is a parameter of the augmented Lagrange method and $\rho > 0$. The variables are updated as follows:

$$x_1^{t+1} = arg\ min_{x_1}\ \mathcal{L}(x_1, x_2^t, \varphi^t). \tag{8}$$

$$x_2^{t+1} = arg\ min_{x_2}\mathcal{L}(x_1^{t+1}, x_2, \varphi^t). \tag{9}$$

$$\varphi^{t+1} = \varphi^t - \rho(\mathcal{A}_1 x_1^{t+1} + \mathcal{A}_2 x_2^{t+1} - c). \tag{10}$$

The ADMM algorithm combines the advantages of both the method of multipliers and the dual ascent method: convergence of weak conditions and the decomposable solvability. However, as we can observe from the ST-SC problem and Eq. (8),(9),(10), there are three main reasons that the ADMM algorithm fails to directly solve the ST-SC problem. First, the ADMM algorithm can only solve the problem of two blocks of variables. Second, the ADMM algorithm cannot solve the problem with inequality restrictions. Finally, the ADMM algorithm also cannot solve problems with some inseparable variables. Hence, we explore some better solutions to the ST-SC problem in the next section.

B. A DISTRIBUTED VARIABLE SPLITTING ADMM ALGORITHM

In this section, we combine the variable splitting technique and the construct indicator function technique with the ADMM algorithm to solve the ST-SC problem.

It is common to construct the indicator function for handling complicated constraints in optimization problems. This paper attempts to follow this idea and introduce two indicator functions to deal with the inequality and inseparable variables constraint. Let $k_n = \{\alpha_{\bullet n} : \sum_{n\in\mathcal{N}} \mathcal{E}_n^{total}(\alpha_{n\bullet}) \le \Phi_n,\ \forall n \in \mathcal{N}\}$ be the polyhedra of each constraint of CRVs and CRSUs i in the ST-SC problem where $\alpha_{\bullet\iota} = \langle\alpha_{n\iota}\rangle_{n\in\mathcal{N}}$ is the vector of amount of task to be completed by CRVs and CRSUs i. Then, follow the same line on variable splitting in [41], we can obtain the following equivalent problem of the ST-SC problem:

$$\min_{\{\alpha_{\bullet n}\},\{x\}} \sum_{n=1}^{\mathcal{N}} \left(\mathcal{CT}(\alpha_{\bullet n}) + \mathcal{I}_{k_n}(\alpha_{\bullet n})\right) + \mathcal{I}_{k_c}(x)$$
$$s.t.\ \alpha_{\bullet n} - x_n = 0, \quad n \in \mathcal{N}. \tag{11}$$

where $\mathcal{I}_{k_n}$ and $\mathcal{I}_{\mathcal{Z}}$ are the indicator function defined as follows:

$$\mathcal{I}_{k_n}(\alpha_{\bullet n}) = \begin{cases} 0, & \text{if } \alpha_{\bullet n} \in k_n \\ +\infty, & \text{else.} \end{cases} \tag{12}$$

$$\mathcal{I}_{k_c}(x) = \begin{cases} 0, & \text{if } x \in k_c \\ +\infty, & \text{else.} \end{cases} \tag{13}$$

and k_c is defined as follows:

$$k_c = \left\{ x : \sum_{n=1}^{\mathcal{N}} \mathcal{I}_{\mathcal{N}} x_n \le 1 \right\}, \tag{14}$$

where $\mathcal{I}_{\mathcal{N}}$ is an identity matrix with size N.

Problem (11), mentioned above, meets the requirements of the ADMM algorithm and the augmented Lagrangian form of it is as follows:

$$\mathcal{L}_\rho(\alpha, x, \varphi) = \sum_{n=1}^{\mathcal{N}} \{\mathcal{CT}(\alpha_{\bullet n}) + \mathcal{I}_{k_n}(\alpha_{\bullet n}) - \varphi(\alpha_{\bullet n} - x_n) + \frac{\rho}{2}\|\alpha_{\bullet n} - x_n\|_2^2\}. \tag{15}$$

Next, we decompose the above problem (15) into $\mathcal{N}$ independent sub-problems. Initializing x^0, φ^0 and updating α as follows:

$$\alpha^{t+1} = arg\ min \sum_{n=1}^{\mathcal{N}} \{\mathcal{CT}(\alpha_{\bullet n}) + \mathcal{I}_{k_n}(\alpha_{\bullet n}) - \varphi_n^t(\alpha_{\bullet n} - x_n^t) + \frac{\rho}{2}\|\alpha_{\bullet n} - x_n^t\|_2^2\}. \tag{16}$$

Then, update x and φ:

$$x^{t+1} = arg\ min\ \frac{\rho}{2}\left\|\alpha_{\bullet n}^{t+1} - x_n^t + \frac{1}{\rho}\varphi^t\right\|_2^2 + \mathcal{I}_{k_c}(x). \tag{17}$$

$$\varphi^{t+1} = \varphi^t - \rho(\alpha^{t+1} - x^{t+1}). \tag{18}$$

It is observed that the distributed variable splitting ADMM (VS-ADMM) algorithm is not efficient when $\mathcal{N}$ is large since this algorithm requires a large number of matrix multiplications and 2-Norm operations. Therefore, in the following section, we try to incorporate the Gauss-Seidel method to solve the above TS-SC problem in a better way.

C. A DISTRIBUTED GAUSS-SEIDEL ADMM ALGORITHM

To obtain a better solution than VS-ADMM algorithm, this section introduces the Gauss-Seidel method into solving the ST-SC problem. Initializing α^0, φ^0 and updating α, φ as follows:

$$\alpha^{t+1} = arg\ min \sum_{n=1}^{\mathcal{N}} \{\mathcal{CT}(\alpha_{\bullet n}) + \mathcal{I}_{k_n}(\alpha_{\bullet n}) - \varphi_n^t(\alpha_{\bullet n})\} + \frac{\rho}{2}\left\|\sum_{n=1}^{\mathcal{N}} \alpha_{\bullet n} + \sum_{j\le n} \mathcal{I}_{k_j}^{t+1}(\alpha_{\bullet j}) + \sum_{j>n} \mathcal{I}_{k_j}^{t}(\alpha_{\bullet j})\right\|_2^2. \tag{19}$$

$$\varphi^{t+1} = \varphi^t - \rho\left(\sum_{n=1}^{\mathcal{N}} \mathcal{I}_{k_n}^{t+1}(\alpha_{\bullet n})\right). \tag{20}$$

where j is a CRVs or CRSUs, e.g., $j \in \mathcal{N}$.

From Eq. (19) and (20), we find that the distributed Gauss-Seidel ADMM (GS-ADMM) algorithm is much more efficient than the distributed VA-ADMM algorithm because GS-ADMM only requires one time 2-Norm operation. However, it has two drawbacks. First, it may not converge when $\mathcal{N} \ge 3$ [38]. Second, there is a term $\mathcal{I}_{k_j}^{t+1}(\alpha_{\bullet j})$ in the Eq. (19), which means that all values of α must be updated sequentially. It does not meet the requirements of processing tasks in parallel among the CRVs and CRSUs in the proposed scenario.

To overcome the shortcomings of non-parallelization for the distributed GS-ADMM algorithm, we consider to incorporate a novel method, known as Jacobi, into solving the ST-SC problem in the following section.

D. A DISTRIBUTED JACOBI ADMM ALGORITHM

This section introduces a more efficient algorithm: the distributed Jacobi ADMM (DJ-ADMM) algorithm, which utilizes the advantages of parallel computing of Jacobi method. DJ-ADMM algorithm updates α in parallel as follows:

$$\alpha^{t+1} = arg\ min \sum_{n=1}^{\mathcal{N}} \{\mathcal{CT}(\alpha_{\bullet n}) + \mathcal{I}_{k_n}(\alpha_{\bullet n}) - \varphi_n^t(\alpha_{\bullet n})\} + \frac{\rho}{2}\left\| \sum_{n=1}^{\mathcal{N}} \alpha_{\bullet n} + \sum_{j\neq i} \mathcal{I}_{k_j}^t(\alpha_{\bullet j}) \right\|_2^2. \quad (21)$$

Then, update φ as follows:

$$\varphi^{t+1} = \varphi^t - \rho \left(\sum_{n=1}^{\mathcal{N}} \mathcal{I}_{k_n}^{t+1}(\alpha_{\bullet n}) \right). \quad (22)$$

By comparing Eq. (19) and (21), we observe that the value of $\mathcal{I}_{k_j}^{t+1}(\alpha_{\bullet j})$ is no longer needed when updating α; thus, we can process the task in parallel. However, DJ-ADMM algorithm is harder to converge than the GS-ADMM algorithm even for two block variables [39].

E. A DISTRIBUTED IMPROVED JACOBI ADMM ALGORITHM

Motivated by the proximal scheme [44], we propose a distributed and improved Jacobi ADMM (DIJ-ADMM) algorithm. The DIJ-ADMM algorithm adds a term $\frac{1}{2}\|\alpha_{\bullet n} - \alpha_{\bullet n}^t\|_{\Theta_n}^2$ and a weight $\tau \geq 0$ for the update of α and φ, respectively. Θ_n is a matrix and $\|\alpha_{\bullet n}\|_{\Theta_n}^2 := \alpha_{\bullet n}^{\mathrm{T}} \Theta_n \alpha_{\bullet n}$. First, update α as follows:

$$\alpha^{t+1} = arg\ min \sum_{n=1}^{\mathcal{N}} \{\mathcal{CT}(\alpha_{\bullet n}) + \mathcal{I}_{k_n}(\alpha_{\bullet n}) - \varphi_n^t(\alpha_{\bullet n})\} + \frac{\rho}{2}\left\| \sum_{n=1}^{\mathcal{N}} \alpha_{\bullet n} + \sum_{j\neq n} \mathcal{I}_{k_j}^t(\alpha_{\bullet j}) \right\|_2^2 + \frac{1}{2}\|\alpha_{\bullet n} - \alpha_{\bullet n}^t\|_{\Theta_n}^2. \quad (23)$$

Then, update φ as follows:

$$\varphi^{t+1} = \varphi^t - \tau \cdot \rho \left(\sum_{n=1}^{\mathcal{N}} \mathcal{I}_{k_n}^{t+1}(\alpha_{\bullet n}) \right). \quad (24)$$

The proposed DIJ-ADMM algorithm has several advantages over the previous ones. For example, as we will show in Section V-A, it possesses the global convergence with an high rate $o(1/t)$. In addition, DIJ-ADMM algorithm can make a subproblem strictly or strongly convex. That is, it can help to give an unique solution. The process of DIJ-ADMM algorithm shows in Algorithm 1.

Algorithm 1 DIJ-ADMM algorithm

Input: α^0, φ^0, ρ, $n \in \mathcal{N}$
Output: α^t, $t \in \mathcal{T}$
for $t = 0, 1, \cdots$ **do**
  Update α^t in parallel by:

$$\alpha^{t+1} = arg\ min \sum_{n=1}^{\mathcal{N}} \{\mathcal{CT}(\alpha_{\bullet n}) + \mathcal{I}_{k_n}(\alpha_{\bullet n}) - \varphi_n^t(\alpha_{\bullet n})\} + \frac{\rho}{2}\left\| \sum_{n=1}^{\mathcal{N}} \alpha_{\bullet n} + \sum_{j\neq n} \mathcal{I}_{k_j}^t(\alpha_{\bullet j}) \right\|_2^2 + \frac{1}{2}\|\alpha_{\bullet n} - \alpha_{\bullet n}^t\|_{\Theta_n}^2.$$

  Update

$$\varphi^{t+1} = \varphi^t - \tau \cdot \rho \left(\sum_{n=1}^{\mathcal{N}} \mathcal{I}_{k_n}^{t+1}(\alpha_{\bullet n}) \right).$$

F. A CASE STUDY

In this section, we present a case study to map the proposed DIJ-ADMM algorithm with the considered scenario.

The real-time vision processing application that we considered in this paper is different from the traditional surveillance application to find the "Person of Interest." To ensure public safety in the smart city, the real-time vision processing application identifies behaviors instead of identities of citizens through smart cameras. After the vision is processed with the cooperation of CRVs and CRSUs near the smart camera, it generates an alert in real-time accordingly.

Once accepting the task from smart cameras, the initiator CRVs can discover both helper CRVs and CRSUs via user datagram protocol broadcasting messages. Then, the initiator CRVs sets up initial parameters for the DIJ-ADMM algorithm, e.g., ρ, α^0, φ^0 and τ. It also determines the part of tasks completed by itself, helper CRVS, and CRSUs via the proposed DIJ-ADMM algorithm, e.g., α_{nn}, α_{nl} and α_{nm}. Then, the initiator CRVs transfers the corresponding tasks to the right CRVs and CRSUs through the CR routers deployed on them. After completing the task, the initiator CRVs aggregates results from the participated CRVs and CRSUs and transfers the feedback to smart cameras.

V. PERFORMANCE ANALYSIS

In this section, we analyze the optimality and complexity of the proposed DIJ-ADMM algorithm in detail and prove that all of the proposed four algorithms provide privacy preserving for participated CRVs and CRSUs.

A. OPTIMALITY AND COMPLEXITY ANALYSIS OF THE PROPOSED DIJ-ADMM ALGORITHM

Lemma 1: When $t \geq 1$, we have

$$\left\|\Re^t - \Re^*\right\|_{\mathcal{P}}^2 - \left\|\Re^{t+1} - \Re^*\right\|_{\mathcal{P}}^2 \geq \left\|\Re^t - \Re^{t+1}\right\|_{\mathcal{H}}^2,$$

where

$$\left\|\Re^t - \Re^{t+1}\right\|_{\mathcal{H}}^2 := \left\|\alpha^t - \alpha^{t+1}\right\|_{\mathcal{P}}^2 + \frac{2-\tau}{\rho\tau^2}\left\|\varphi^t - \varphi^{t+1}\right\|^2 - \frac{2}{\tau}(\varphi^t - \varphi^{t+1})^{\mathrm{T}}\mathcal{C}\mathcal{T}(\alpha^t - \alpha^{t+1}),$$

$\left\|\Re^t - \Re^*\right\|_{\mathcal{P}}^2$ is the bound of error.

Proof: For the process proofing, please refer to [44, Lemma 2.1]. □

Lemma 2: The proposed DIJ-ADMM algorithm possesses the convergence rate $o(1/t)$ of converging to the global optimal policy for the ST-SC problem.

Proof: First, we prove that the convergence of proposed DIJ-ADMM algorithm.

When $\mathcal{H}$ is a positive definite matrix, we have

$$\left\|\Re^t - \Re^{t+1}\right\|_{\mathcal{H}}^2 \geq \ell \cdot \left\|\Re^t - \Re^{t+1}\right\|^2 \geq 0. \tag{25}$$

where $\ell > 0$. Then, according to Lemma 1, the following inequality is established

$$\left\|\Re^t - \Re^*\right\|_{\mathcal{P}}^2 - \left\|\Re^{t+1} - \Re^*\right\|_{\mathcal{P}}^2 \geq \ell \cdot \left\|\Re^t - \Re^{t+1}\right\|^2. \tag{26}$$

$\left\|\Re^t - \Re^*\right\|_{\mathcal{P}}^2$ represents the gap between the proposed DIJ-ADMM algorithm and the optimal policy after t iterations. (26) indicates that this gap is non-increasing, as well as this algorithm is converging, e.g., $\left\|\Re^t - \Re^{t+1}\right\|^2 \to 0$, which proves the optimality of the proposed DIJ-ADMM algorithm.

Then, let us prove the convergence rate $o(1/t)$ of proposed DIJ-ADMM algorithm. We have the following inequality

$$\left\|\Re^t - \Re^*\right\|_{\mathcal{P}}^2 - \left\|\Re^{t+1} - \Re^*\right\|_{\mathcal{P}}^2 \geq \left\|\Re^t - \Re^{t+1}\right\|_{\mathcal{H}}^2 \geq \ell \cdot \left\|\Re^t - \Re^{t+1}\right\|_{\mathcal{G}}^2. \tag{27}$$

summing (27) over t gets $\sum_{t=1}^{\infty}\left\|\Re^t - \Re^{t+1}\right\|_{\mathcal{G}}^2 < \infty$. Furthermore, $\sum_{t=1}^{\infty}\left\|\Re^t - \Re^{t+1}\right\|_{\mathcal{G}}^2$ is monotonically non-increasing. When $t \to +\infty$, the following formula is established

$$t \cdot \Re^{2t} \leq \Re^{t+1} + \Re^{t+2} + \cdots + \Re^{2t} \to 0. \tag{28}$$

thus, $\left\|\Re^t - \Re^{t+1}\right\|_{\mathcal{G}}^2 = o(1/t)$ holds. That is, $\left\|\alpha^t - \alpha^{t+1}\right\|^2 = o(1/t)$ and $\left\|\varphi^t - \varphi^{t+1}\right\|^2 = o(1/t)$. It means that the convergence rate of proposed DIJ-ADMM algorithm is $o(1/t)$. □

The minimum time complexity of existing algorithms is $\mathcal{O}(1/t)$. However, the proposed DIJ-ADMM algorithm achieves a lower time complexity with the convergence rate $o(1/t)$. Note that time complexity is sharply decreased when the convergence rate changes from $\mathcal{O}(1/t)$ to $o(1/t)$.

In the next section, from the perspective of security, we analyze the ability of preserving the privacy of the proposed four algorithms. Preserving the privacy of users is very important in task scheduling, but almost all works do not consider this, such as surrogate vehicle selection strategies in [40] require multi private attributes (e.g., locations, speed, computation capability, etc.) of vehicles. Nevertheless, note that preserving privacy is not the main focus of this paper, further information about security can be obtained from our recent work in [42].

B. PRIVACY PRESERVING ANALYSIS OF THE PROPOSED FOUR ALGORITHMS

Lemma 3: The optimization problem in (5) can be divided into $\mathcal{N}$ independent sub-problems, which means that each CRUs and CRSUs can independently complete its own computing tasks, thus protecting the privacy of them.

Proof: We prove the privacy preserving of the proposed four algorithms from two aspects. First, we analyze the augmented Lagrangian form of the objective function in (5) and prove that it is separable. We can rewrite (5) as the following:

$$\mathcal{L}(\alpha_{\bullet 1}, \alpha_{\bullet 2}, \cdots, \alpha_{\bullet\mathcal{N}}, \varphi, \Lambda) = \sum_{i=1}^{\mathcal{N}} \mathcal{L}_{\mathcal{D}}(\alpha_{\bullet n}, \varphi_n, \Lambda_n), \tag{29}$$

where

$$\mathcal{L}_{\mathcal{D}}(\alpha_{\bullet n}) = \mathcal{D}(\alpha_{\bullet}) + \mathcal{I}_{\mathcal{Z}}(\alpha_{\bullet n}) + \Lambda_n^T\alpha_{\bullet n} - \varphi_n) + \frac{\rho}{2}\left\|\alpha_{\bullet n} - \varphi_n\right\|_2^2, \tag{30}$$

and $\mathcal{D}(\alpha_{\bullet n})$ is defined as

$$\mathcal{D}(\alpha_{\bullet n}) = \alpha_{nn}\left(\frac{1}{\mu_n - \alpha_{nn}}\right) + \sum_{\iota\in\mathcal{N}\setminus\{n\}}\alpha_{n\iota}\left(s_{n\iota} + \frac{1}{\mu_\iota - \sum_{\iota\in\mathcal{N}\setminus\{n\}}\alpha_{n\iota}}\right) + \sum_{m\in\mathcal{N}\setminus\{n\}}\alpha_{nm}s_{nm}. \tag{31}$$

Eq. (30) shows that variables in $\mathcal{L}_{\mathcal{D}}$ are independent from each other. It validates that the augmented Lagrangian form of the objective function in (5) can be divided into $\mathcal{N}$ sub-problems and each of them can be solved by the respective CRVs and CRSUs with their own private information. Shortly, the proposed algorithms for the objective function in (5) can preserve the privacy of CRVs and CRSUs in the system. □

Lemma 3 is consistent with the observation in variable update of four proposed algorithms.

VI. SIMULATION RESULTS

In this section, we provide performance analysis based on simulation concerning task scheduling for the real-time vision process applications with the assistance of IoV based on multi-server mobile edge computing.

A. SIMULATION SETUP

In the simulation, the task publisher (e.g., smart cameras) publishes a task, e.g., processing real-time vision to ensure public safety, which is completed with the cooperation of the CRVs and the CRSUs (as the distributed edge servers). The number of the CRVs and the CRSUs are 5 and 1, respectively, but the numbers are not fixed to validate the effect. The task publisher, CRVs and the CRSUs are connected via Wi-Fi

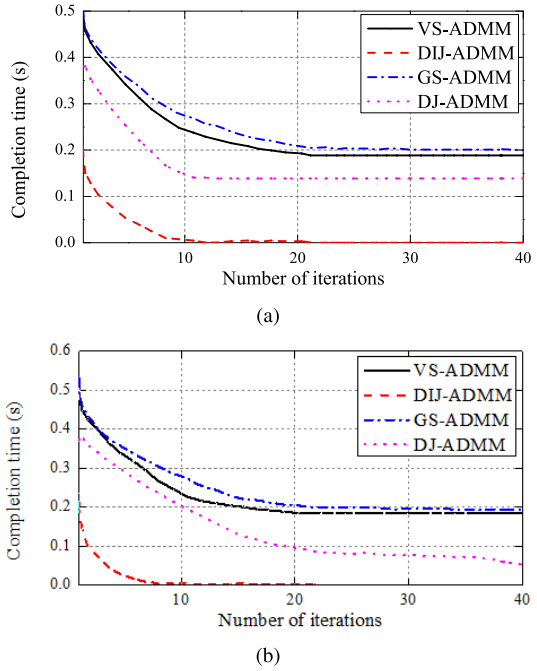

FIGURE 3. Completion time on number of iterations with different node numbers when the task arrival rate is fixed. (a) Completion time vs. Number of iterations ($\mathcal{N} = 6$). (b) Completion time vs. Number of iterations ($\mathcal{N} = 60$).

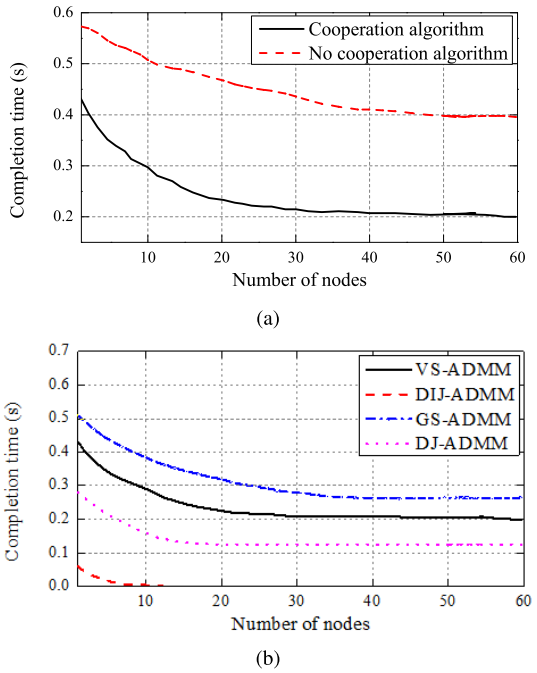

FIGURE 4. Completion time on different number of nodes. (a) Completion time vs. Number of nodes. (b) Completion time vs. Number of nodes.

networks, which is ubiquitous in future smart city, for example, new Wi-Fi standards (802.11ah) have been developed to operate below 1GHz that have a greater range [43]. The distance between the CRVs and CRSUs follows the uniformly randomly distribution with the mean of 90 meters. The considered network system is managed by a network operator (e.g., cloud center.).

In the following analysis of the simulation results, two comparative experiments are set to evaluate the performance of the proposed algorithms. 1) The comparison between the no cooperation algorithm and the cooperation algorithm. No cooperation algorithm refers to processing tasks independently by each individual node and the algorithm with cooperation is represented by the VS-ADMM algorithm proposed in this paper. 2) The comparison of the four task scheduling algorithms proposed in this paper.

B. CONVERGENCE ANALYSIS

First, we consider the convergence iperformance of each algorithm with respect to the number of iterations in Figure 3. From Figure 3(a), we find that all algorithms can converge to the global optimal value and the proposed DIJ-ADMM algorithm has the fastest convergence rate (roughly 10 iterations) and the shortest completion time when the number of nodes (nodes refers to CRVs and CRSUs) is 6. From Figure 3(b), we find that the VS-ADMM algorithm, GS-ADMM algorithm and DIJ-ADMM algorithm can converge to the global optimal value when the number of nodes is 60, but the GS-ADMM algorithm takes the longest time to complete the task. Because the parameters in GS-ADMM algorithm are updated one after another instead of parallelly, as described in section IV-C, which brings a long processing time. To sum up, the proposed DIJ-ADMM algorithm has the fastest convergence rate and the least completion time, which is consistent with the conclusions we made in the algorithm description and theoretical performance analysis.

C. PARAMETERS ANALYSIS

Further, we explore the relationship between the number of nodes and the convergence performance. From Figure 4(a), we observe that the completion time of the no cooperation algorithm is much larger than the algorithm with cooperation and the gap increases as the number of nodes increases. For that reason, the cooperation algorithm can make full use of the resource of the surrounding nodes to offload computation tasks. From Figure 4(b), we observe that the VS-ADMM algorithm and the DJ-ADMM algorithm converge slowly as the number of nodes increases. However, the proposed DIJ-ADMM algorithm still maintains good convergence performance and the shortest completion time when $\mathcal{N}$ is large.

Next, we explore the impact of task arrival rate on task completion time. Here, we set the task arrival rate of all nodes to a fixed value except for node n. From Figure 5(a), we find that the task completion time of both no cooperation algorithm and the proposed VS-ADMM algorithm increases as the task arrival rate increases. However, the task completion time of the DIJ-ADMM algorithm is much smaller than the no cooperation algorithm and it increases slower than the no cooperation algorithm. From Figure 5(b), we find that the four algorithms approximately increase linearly with the task arrival rate and the DIJ-ADMM algorithm has the smallest increase with the arrival rate of the task.

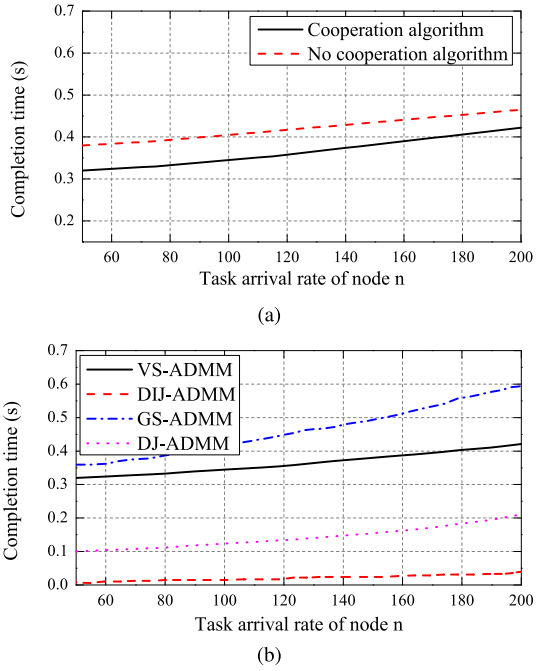

FIGURE 5. Completion time on different task arrival rate. (a) Completion time vs. Task arrival rate of node n ($\mathcal{N} = 6$). (b) Completion time vs. Task arrival rate of node n ($\mathcal{N} = 6$).

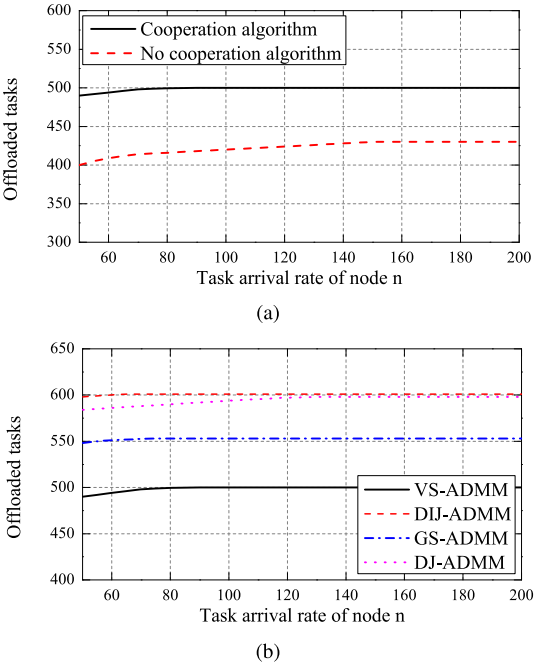

FIGURE 6. Offloaded task on different task arrival rate. (a) Offloaded task vs. Task arrival rate of node n. (b) Offloaded task vs. Task arrival rate of node n.

In Figure 6, we study the relationship between the amount of offloaded tasks and the task arrival rate. Figure 6(a) reflects that the amount of offloaded tasks of both no cooperation algorithm and cooperation algorithm have an approximate linear growth relationship with the task arrival rate when the task arrival rate is small (about 80 or less). After that, both algorithms converge to a fixed value individually and

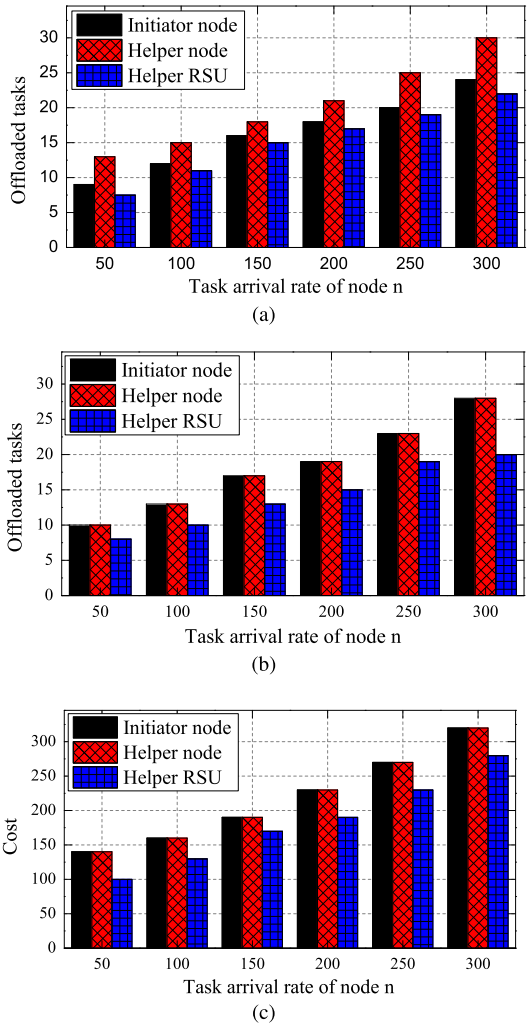

FIGURE 7. Performance comparison on different task arrival rate of node n. (a) Offloaded tasks vs. Task arrival rate of node n (No cooperation algorithm). (b) Offloaded tasks vs. Task arrival rate of node n (Cooperation algorithm). (c) Expense vs. Task arrival rate of node n (Cooperation algorithm).

the VS-ADMM algorithm converges to a value much larger than the no cooperation algorithm. According to Figure 6(b), we observe that the DIJ-ADMM algorithm is capable of scheduling the most tasks among four proposed distributed algorithms.

Next, in Figure 7, we discuss the impact of task arrival rate on the cost and the amount of offloaded tasks. From Figures 7(a), 7(b) and 7(c), the cooperation algorithm can balance the cost and the amount of offloaded tasks of different nodes. Because in the cooperation algorithm, the task initiator node n process the task by itself when the task arrives at a small rate. It offloads the tasks to the other nodes when the task arriving is beyond the capability of initiator node or a ultra low latency is required. The cooperation algorithm can offload to helper CRVs and CRSUs in our scenario.

D. TRADEOFF ANALYSIS

In Figure 8, we investigate the tradeoff between completion time and cost. Here, we merely change the cost of node

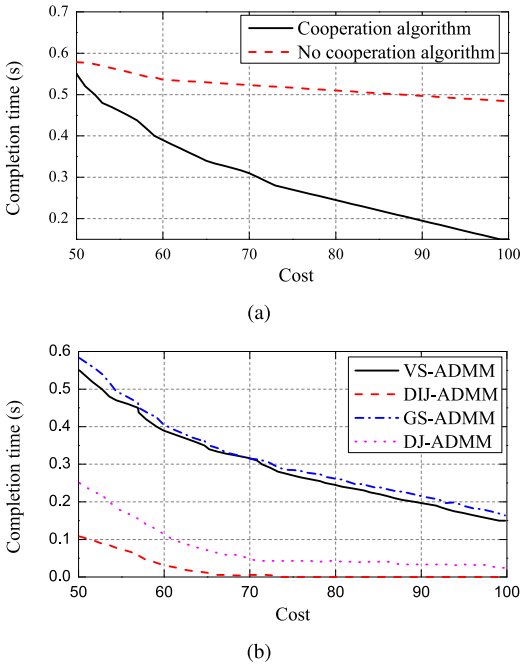

FIGURE 8. Tradeoff between completion time and cost. (a) Completion time vs. Cost. (b) Completion time vs. Cost.

n while the other nodes are constant. From Figure 8(a), we observe that the completion time for both of the cooperation algorithm and no cooperation algorithm decrease with the increase of the cost. The decrease speed is very large when the cost is small (approximately within 60). Moreover, as the number of nodes increases, the completion time decreases drastically. From Figure 8(b), we observe that the proposed four algorithms reduce the completion time with the increase of the cost, but the DIJ-ADMM algorithm has the fastest decline rate and the fastest tendency to obtain the stable value.

VII. CONCLUSIONS

In this paper, we utilize the resources of the IoV to handle the tasks produced by computation-intensive and time-sensitive applications in smart city. We consider the real-time vision process application as the scenario in this paper. To acquire the resources from the IoV for task processing, we formulate an optimization problem to minimize the completion time with a given cost of task scheduling. Furthermore, we develop four evolving task scheduling algorithms to solve the proposed problem based on the classic ADMM algorithm: VS-ADMM, GS-ADMM, DJ-ADMM, and DIJ-ADMM algorithm. By theoretical analysis, we observe that the DIJ-ADMM algorithm can achieve the optimality with a significantly fast convergence rate $o(1/t)$ and these four algorithm have better performance than the existing solutions. At the same time, these algorithms have the ability of privacy preserving. Comparison with the existing methods on task scheduling proves that our algorithms can greatly reduce task completion time and increase the number of offloaded tasks. Additionally, through the comparison of the proposed four methods, we find that the proposed DIJ-ADMM algorithm achieves optimal performance while significantly reduces complexity.

REFERENCES

[1] The Statistics Portal. (Jul. 2018). *Spending on Smart Cities Worldwide in 2015 and 2020 (in Billion U.S. Dollars)*. [Online]. Available: http://bit.ly/2rxQF6Z
[2] J. Woetzel *et al.*, "Smart cities: Digital solutions for a more livable future," McKinsey Global Inst., New York, NY, USA, Tech. Rep., Jun. 2018. [Online]. Available: https://mck.co/2JCWOdq
[3] X. Yao, Y. Lin, Q. Liu, and J. Zhang, "Privacy-preserving search over encrypted personal health record in multi-source cloud," *IEEE Access*, vol. 6, pp. 3809–3823, Jan. 2018.
[4] H. Ding, C. Zhang, Y. Cai, and Y. Fang, "Smart cities on wheels: A newly emerging vehicular cognitive capability harvesting network for data transportation," *IEEE Wireless Commun.*, vol. 25, no. 2, pp. 160–169, Oct. 2018.
[5] J. Feng, S. Yang, and Z. Feng, "Vehicle-assisted offloading on metropolitan streets: Enhancing geographical fluidity of wireless resources," *IEEE Wireless Commun. Lett.*, vol. 6, no. 5, pp. 622–625, Oct. 2017.
[6] X. Hou, Y. Y. Lu, and S. Dey, "Wireless VR/AR with Edge/cloud computing," in *Proc. ICCCN*, Vancouver, BC, Canada, Jul./Aug. 2017, pp. 1–8.
[7] National Science Foundation. (2018). *SCC: Building Safe and Secure Communities through Real-Time Edge Video Analytics*. [Online]. Available: http://bit.ly/2JTOFy2
[8] V. D. Rob. (Jan. 2015). *Predicts 2015: The Internet of Things*. [Online]. Available: http://www.gartner.com/document/2952822
[9] M. Weinberger. (Jan. 2016). *Processor Company Nvidia's New Car-Mounted Supercomputer is as Powerful as 150 MacBook Pros*. [Online]. Available: https://read.bi/2PjWbYZ
[10] H. Ding and Y. Fang, "Virtual infrastructure at traffic lights: Vehicular temporary storage assisted data transportation at signalized intersections," *IEEE Trans. Veh. Technol.*, to be published, doi: 10.1109/TVT.2018.2871414.
[11] T. McGuckin *et al.* (2017). *Leveraging the Promise of Connected and Autonomous Vehicles to Improve Integrated Corridor Management and Operations: A Primer*. [Online]. Available: https://ops.fhwa.dot.gov/publications/fhwahop17001/fhwahop17001.pdf
[12] T. Zhang, A. Chowdhery, P. V. Bahl, K. Jamieson, and S. Banerjee, "The design and implementation of a wireless video surveillance system," *ACM MobiCom*, Paris, France, Sep. 2015, pp. 426–438.
[13] Y. Wu *et al.*, "Secrecy-driven resource management for vehicular computation offloading networks," *IEEE Netw.*, vol. 32, no. 3, pp. 84–91, May/Jun. 2018.
[14] X. Yao, R. Zhang, Y. Zhang, and Y. Lin, "Verifiable social data outsourcing," in *Proc. IEEE INFOCOM*, Atlanta, GA, USA, May 2017, pp. 1–9.
[15] Y. Deng, Z. Chen, D. Zhang, and M. Zhao, "Workload scheduling toward worst-case delay and optimal utility for single-hop fog-IoT architecture," *IET Commun.*, vol. 12, no. 17, pp. 2164–2173, Oct. 2018.
[16] M. Chen and Y. Hao, "Task offloading for mobile edge computing in software defined ultra-dense network," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 587–597, Mar. 2018.
[17] Y. Mao, J. Zhang, Z. Chen, 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.
[18] Y. Hao, M. Chen, L. Hu, M. S. Hossain, and A. Ghoneim, "Energy efficient task caching and offloading for mobile edge computing," *IEEE Access*, vol. 6, pp. 11365–11373, Mar. 2018.
[19] L. Pu, X. Chen, J. Xu, and X. Fu, "D2D fogging: An energy-efficient and incentive-aware task offloading framework via network-assisted D2D collaboration," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3887–3901, Dec. 2016.
[20] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, Jan. 2011.
[21] A. ur R. Khan, M. Othman, A. N. Khan, J. Shuja, and S. Mustafa, "Computation offloading cost estimation in mobile cloud application models," *Wireless Pers. Commun.*, vol. 97, no. 3, pp. 4897–4920, Dec. 2017.
[22] L. Xiao, C. Xie, T. Chen, H. Dai, and H. V. Poor, "A mobile offloading game against smart attacks," *IEEE Access*, vol. 4, pp. 2281–2291, 2016.

[23] S. J. Lee and X. Lin, "Energy-aware paired sampling-based decision model for dynamic mobile-to-mobile service offloading," *IEEE Access*, vol. 5, pp. 5031–5045, Apr. 2017.

[24] W. Zhang, Z. Zhang, S. Zeadally, and H.-C. Chao, "Efficient task scheduling with stochastic delay cost in mobile edge computing," *IEEE Commun. Lett.*, vol. 23, no. 1, pp. 4–7, Jan. 2018, doi: 10.1109/LCOMM.2018.2879317.

[25] W. Zhang, Z. Zhang, and H.-C. Chao, "Cooperative fog computing for dealing with big data in the Internet of vehicles: Architecture and hierarchical resource management," *IEEE Commun. Mag.*, vol. 55, no. 12, pp. 60–67, Dec. 2017.

[26] M. Chen, Y. Hao, L. Hu, M. Hossain, and A. Ghoneim, "Edge-CoCaCo: Towards joint optimization of computation, caching and communication on edge cloud," *IEEE Wireless Commun.*, vol. 25, no. 3, pp. 21–27, Jun. 2018.

[27] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 450–465, Feb. 2018.

[28] M. Chen *et al.*, "A dynamic service-migration mechanism in edge cognitive computing," *ACM Trans. Internet Techn.*, to be published. [Online]. Available: https://arxiv.org/pdf/1808.071981

[29] M. Chen, W. Li, Y. Hao, Y. Qian, and I. Humar, "Edge cognitive computing based smart healthcare system," *Future Gener. Comput. Syst.*, vol. 86, pp. 403–411, Sep. 2018.

[30] M. Chen, Y. Hao, K. Lin, Z. Yuan, and L. Hu, "Label-less learning for traffic control in an edge network," *IEEE Netw.*, vol. 32, no. 6, pp. 8–14, Nov./Dec. 2018.

[31] P. P. Mach and Z. Becvar. (2017). "Mobile edge computing: A survey on architecture and computation offloading." [Online]. Available: https://arxiv.org/abs/1702.05309

[32] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016.

[33] T. X. Tran and D. Pompili. (2017). "Joint task offloading and resource allocation for multi-server mobile-edge computing networks." [Online]. Available: https://arxiv.org/abs/1705.00704

[34] M. Hong, Z.-Q. Luo, and M. Razaviyayn, "Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems," *SIAM J. Optim.*, vol. 26, no. 1, pp. 337–364, 2016.

[35] R. Zhang and J. Kwok, "Asynchronous distributed ADMM for consensus optimization," in *Proc. Int. Conf. Mach. Learn.*, Jan. 2014, pp. 1701–1709.

[36] H. Ding, C. Zhang, B. Lorenzo, and Y. Fang, "Access point recruitment in a vehicular cognitive capability harvesting network: How much data can be uploaded?" *IEEE Trans. Veh. Techn.*, vol. 67, no. 7, pp. 6438–6445, Jul. 2018.

[37] Y. Xiao and M. Krunz, "QoE and power efficiency tradeoff for fog computing networks with fog node cooperation," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, May 2017, pp. 1–9.

[38] C. Chen, B. He, Y. Ye, and X. Yuan, "The direct extension of ADMM for multi-block convex minimization problems is not necessarily convergent," *Math. Program.*, vol. 155, nos. 1–2, pp. 57–79, 2016.

[39] B. He, L. Hou, and X. Yuan, "On full Jacobian decomposition of the augmented Lagrangian method for separable convex programming," *SIAM J. Optim.*, vol. 25, no. 4, pp. 2274–2312, 2015.

[40] B. Li, Y. Pei, H. Wu, Z. Liu, and H. Liu, "Computation offloading management for vehicular ad hoc cloud," in *Proc. Int. Conf. Algorithms Architectures Parallel Process.*, 2014, pp. 728–739.

[41] D. Bertsekas and J. Tsitsiklis. (2003). *Parallel and Distributed Computation: Numerical Methods*. [Online]. Available: https://dspace.mit.edu/bitstream/handle/1721.1/3719/part1.pdf

[42] X. Yao, Y. Chen, R. Zhang, Y. Zhang, and Y. Lin, "Beware of what you share: Inferring user locations in Venmo," *IEEE Internet Things J.*, vol. 5, no. 6, pp. 5109–5118, Dec. 2018, doi: 10.1109/JIOT.2018.2844218.

[43] TechRadar Pro. (Jul. 2018). *Are WiFi Networks Ready for Smart Cities?* [Online]. Available: https://www.techradar.com/news/are-wifi-networks-ready-for-smart-cities

[44] W. Deng, M. J. Lai, Z. M. Peng, and W. T. Yin, "Parallel multi-block ADMM with o(1/k) convergence," *J. Sci. Comput.*, vol. 66, no. 3, pp. 889–916, Mar. 2016.

YIQIN DENG received the B.S. degree in project management from the Hunan Institute of Engineering, in 2014, and the M.S. degree in software engineering from the Central South University, China, in 2017, where she is currently pursuing the Ph.D. degree in computer science and technology. Her research interests include edge/fog computing, the Internet of Vehicles, smart city, and resource management. She is a member of the IEEE and the China Computer Federation.

ZHIGANG CHEN received the B.S., M.S., and Ph.D. degrees from Central South University, China, in 1984, 1987, and 1998, respectively. He is currently a Professor, a Ph.D. Supervisor, and the Dean of the School of Software, Central South University. His research interests include cluster computing, parallel and distributed systems, computer security, and wireless networks. He is also the Director and an Advanced Member of the China Computer Federation (CCF) and a member of the Pervasive Computing Committee of CCF.

XIN YAO received the B.S. degree in computer science from Xidian University, in 2011, and the M.S. degree in software engineering and the Ph.D. degree in computer science and technology from Hunan University, in 2013 and 2018, respectively. From 2015 to 2017, he was a Visiting Scholar with Arizona State University. He is currently an Assistant Professor with Central South University. His research interests include security and privacy issues in social networks, the Internet of Things, cloud computing, and big data. He is a member of the IEEE and the China Computer Federation.

SHAHZAD HASSAN received the B.S. degree in software engineering from Riphah International University, Pakistan, in 2013, and the M.Eng. degree in software engineering from Central South University, China, in 2016, where he is currently pursuing the Ph.D. degree in computer application and technology. His research interests include sensors, the Internet of Things, fog computing, and edge computing.

JIA WU received the Ph.D. degree in software engineering from Central South University, Changsha, Hunan, China, in 2016, where he currently holds a Postdoctoral position at the School of Information Science and Engineering. Since 2010, he has been an Algorithm Engineer with IBM, Seoul, South Korea, and Shanghai, China. His research interests include wireless communications and networking, wireless networks, big data research, and mobile health in network communication. He is a Senior Member of the China Computer Federation and a member of the IEEE and ACM.