Article

Joint Node Selection and Resource Allocation for Task Offloading in Scalable Vehicle-Assisted Multi-Access Edge Computing

Xuan-Qui Pham [ID]**, Tien-Dung Nguyen, VanDung Nguyen** [ID] **and Eui-Nam Huh ***

Department of Computer Science and Engineering, Kyung Hee University, Yongin-si 17104, Korea; pxuanqui@khu.ac.kr (X.-Q.P.); ntiendung@khu.ac.kr (T.-D.N.); ngvandung85@khu.ac.kr (V.N.)

* Correspondence: johnhuh@khu.ac.kr; Tel.: +82-031-201-2454

check for updates

Abstract: The resource limitation of multi-access edge computing (MEC) is one of the major issues in order to provide low-latency high-reliability computing services for Internet of Things (IoT) devices. Moreover, with the steep rise of task requests from IoT devices, the requirement of computation tasks needs dynamic scalability while using the potential of offloading tasks to mobile volunteer nodes (MVNs). We, therefore, propose a scalable vehicle-assisted MEC (SVMEC) paradigm, which cannot only relieve the resource limitation of MEC but also enhance the scalability of computing services for IoT devices and reduce the cost of using computing resources. In the SVMEC paradigm, a MEC provider can execute its users' tasks by choosing one of three ways: (i) Do itself on local MEC, (ii) offload to the remote cloud, and (iii) offload to the MVNs. We formulate the problem of joint node selection and resource allocation as a Mixed Integer Nonlinear Programming (MINLP) problem, whose major objective is to minimize the total computation overhead in terms of the weighted-sum of task completion time and monetary cost for using computing resources. In order to solve it, we adopt alternative optimization techniques by decomposing the original problem into two sub-problems: Resource allocation sub-problem and node selection sub-problem. Simulation results demonstrate that our proposed scheme outperforms the existing schemes in terms of the total computation overhead.

Keywords: task offloading; resource allocation; mobile cloud computing; multi-access edge computing; vehicular cloud; Internet of Things

1. Introduction

During the last decade, we witnessed a striking rise in population of mobile Internet of Things (IoT) devices, such as smart phones, tablets, wearable devices, and sensors. In addition, according to Information Handling Services Markit company, the IoT market will grow from an installed base of 15.4 billion devices in 2015 to 30.7 billion devices in 2020 and 75.4 billion in 2025 [1]. Together with it, a new breed of applications and services for IoT devices is constantly emerging and attracting great attentions. For example, the Internet of Medical Things (IoMT) [2] is an application of the IoT for medical and health-care purposes, from remote monitoring to smart sensors and medical device integration. The IoMT cannot only provide better treatment and assistant services to patients and disabled people in daily routine, but also offer health-care providers with actual health data to identify issues before they become critical [3,4]. Besides health-care systems, the IoT can also assist in the integration of communications, control and information processing across various transportation systems, which helps improve the efficiency and convenience of modern day transportation [5]. In particular, the IoT is used for smart road and traffic management as a means of reducing traffic

congestions or redirecting traffic in response to an accident or other types of hazard by building sensors into highways and surface streets [6]. Moreover, the IoT has driven the autonomous vehicle revolution. Modern vehicles utilize a series of radar lasers, high-powered cameras, and smart sensors that map out the vehicles' surrounding. Accordingly, many applications for autonomous vehicles have emerged, and can be classified into safety applications (e.g., localization and navigation, obstacle detection, accident avoidance, remote-sensing, etc.) [7–10] and non-safety applications (e.g., media sharing, infotainment, file transfer, gaming, etc.) [11–13]. Not only that, the applications of the IoT can be found in other areas, such as home automation [14], industrial manufacturing [15], environmental monitoring [16], and so on. Therefore, along with the explosive growth of IoT devices, the potential of next-generation IoT applications is boundless and they affect every aspects of human life.

However, these next-generation IoT applications usually require intensive computation capacity and persistent data processing. Meanwhile, IoT devices typically have intrinsic limitations of battery life, processing power, storage capacity, and network resources. To address this challenge, mobile cloud computing (MCC) [17] has been introduced to assist IoT devices by enabling offloading compute-intensive, resource-consuming tasks up to a powerful computing platform in cloud. Nevertheless, because of the far distance between the cloud and IoT devices, traditional MCC has the drawbacks of high transmission delay, poor reliability, heavy burden on network performance, and degradation of quality of service (QoS).

Multi-access edge computing (MEC) [18] or formerly mobile edge computing, as a new architecture and key technology for the coming 5G networks, has been proposed to tackle the above issues. MEC promotes the idea of deploying cloud resources (e.g., MEC servers) at the edge of networks in close proximity to IoT devices (e.g., WiFi access points and base stations). By this way, MEC can provide low-latency high-reliability computing services for IoT devices. Similar concepts including fog computing [19] and cloudlet [20] have been perceived as what is known as edge computing paradigm. However, considering the economic and scalable deployment, the resources of MEC are often limited. The rapid increase of task requests from IoT devices will cause the resource bottleneck of MEC servers, and considerably affect the task execution delay.

To deal with the resource limitation of MEC, a solution is to integrate MEC and remote cloud to extend the total resource capacity [21–24]. The benefit of scalability, on-demand and pay-as-you-go services of the cloud can help the MEC fulfill the increasing demands of large-scale compute-intensive offloading tasks if the resources of MEC are not sufficient. Moreover, thanks to the widespread popularity of the passive optical networks (PONs) recently, the latency and reliability during the data exchange with the centralized cloud via the wide area network (WAN) have been well improved [21]. Hence, cloud computing and MEC are complementary technologies.

On the other hand, vehicular cloud (VC) [25], the merging of MCC and Vehicular Ad hoc Network (VANET), has also emerged as another extension of the MEC resources by leveraging the on-board resources in participating vehicles. In [26], vehicle as a resource (VaaR) vision is introduced and stimulated by the ubiquity of vehicles and the vehicular resources that are readily available. To better perceive the VaaR potentialities, in 2015, Gartner estimated a quarter billion connected vehicles on the road globally by 2020, making connected cars a key enabler for the revolution of the IoT [27]. Thanks to the advances in automotive industry, vehicles are augmented with a variety of resources (e.g., wireless communication, on-board storage, and computing units), and can be regarded as "computers-on-wheels". However, currently these resources are not exploited efficiently. Their on-board resources are usually under-utilized for most of their time, during which can be used for other task. Also, urban vehicular networks are recognized as a significant component of the future intelligent transportation, supporting three types of communication: Between vehicles (V2V), between vehicles and infrastructures (V2I and I2V), between vehicles and any neighboring object (V2X) communications [28]. These features help vehicles become potential computing nodes in a cloud computing network. For example, in [29,30], vehicles are employed to share the computation tasks offloaded by MEC providers. By integrating various usable resources of volunteer vehicles,

we can further relieve the resource limitation of MEC, enhance the scalability of computing services for IoT devices and reduce the cost of using cloud resources. However, we note that most recent task offloading studies to address the resource limitation problem of MEC have only taken the MEC and remote cloud resources into consideration while ignoring the abundant resources that can be offered by volunteer vehicles in the VC.

Therefore, in this paper, we propose to combine MEC with fixed remote cloud and vehicular cloud to expand the currently available resources of MEC for task requests from IoT devices. In our scenario, a MEC provideroffers computing services for IoT devices. After receiving the task requests from IoT devices, the MEC provider makes a strategy to allocate these computation tasks to a computing platform originated from the infrastructure of MEC, remote cloud, and vehicular cloud. The remote cloud can help execute excessive offloading tasks of MEC. However, it is accompanied by monetary cost charged to the MEC provider for using the cloud resources. Meanwhile, in the vehicular cloud, we use buses in the VC as candidate mobile volunteer nodes (MVNs) that share their idle computing resources for tasks offloaded from the MEC provider. Incentives are offered to the owners of MVNs to encourage sharing their resources. Therefore, besides task completion time, the monetary cost for using computing resource should be in consideration of the MEC provider.

The main contributions of this paper are summarized as follows.

- We investigate an innovative framework of task offloading in a scalable vehicle-assisted MEC (SVMEC), where the MEC capacity is extended by renting resources from a remote cloud and vehicular cloud. We stand on the perspective of a MEC provider, whose objective is to minimize the total computation overhead in terms of the weighted-sum of task completion time and monetary cost for using computing resources.
- We formulate the problem of joint node selection and resource allocation as a Mixed Integer Nonlinear Programming (MINLP) problem that jointly optimizes the task offloading decisions and computing resource allocation to the offloaded tasks, so as to minimize the total computation overhead of the MEC provider.
- We solve the problem by decomposing the original problem into two-subproblem (i) Resource allocation (RA) problem with fixed task offloading decision and (ii) Node selection (NS) problem that optimizes the optimal-value function based on the solution of RA problem.
- We also justify the efficiency of our proposed scheme by extensive simulations. We compare the performance in terms of total computation overhead between our proposed scheme and three other strategies. The comparison is conducted under different situations, such as different number of tasks and task's profiles (i.e., compute-intensive and data-intensive tasks). In each situation, we also analyze the trend of task distribution on MEC, remote cloud, and MVNs in order to explain the achieved result of our proposal.
- Based on the experimental results, we can conclude that compared with other strategies, our proposed scheme provides the MEC provider a better solution to optimize the total computation overhead.

The remainder of this paper is organized as follows. In Section 2, we discuss some related works. The system model and optimization problem are described in Section 3. We present our proposed solution in Section 4. Simulation results are presented and discussed in Section 5, followed by our conclusions in Section 6.

2. Related Work

In recent year, there have been various studies that attempt to solve task offloading problem in MEC. The cooperation of multi-level central/edge clouds introduces new challenges and attracts relevant investigations on proper node selection and resource allocation for task offloading. For example, Zhao et al. [24] proposed a cooperative scheduling scheme of MEC (so-called local cloud) and central cloud to maximize the amount of the tasks served by the MEC while satisfying the delay requirements of the offloaded tasks. The task offloading decision depends on the task priorities in

terms of delay requirements, and the availability of MEC resources. Specifically, the offloaded tasks are firstly delivered to the local scheduler within the MEC. The scheduler applies a threshold-based policy, which defines several buffer thresholds for each priority level. When the computation load of the MEC server exceeds the buffer threshold, the scheduler delegates the task to the central cloud. Meanwhile, in [31], the minimization of the task execution delay is done by allocating computing resources of a cluster of nearby MECs, while avoiding using the central cloud. The MEC cluster is formed by means of a cooperative game approach such that the available computation resources are maximally exploited while participating MECs receive incentives in a fair manner. Not only the execution delay, the selection of computing nodes in [32] also considers power consumption of the computation nodes. This paper proposes three different MEC clustering strategies to handle a single user's request in order to minimize execution delay, minimize overall power consumption of cluster and minimize the power consumption of each MEC in the cluster, respectively. This work is extended to a multi-user scenario in [33] where each user has assigned different cluster size depending on the applications and the user's requirements. Note that in most of these works, the objective is to minimize the execution delay of the offloaded tasks and/or the power consumption of computing nodes while the cost for using the resources is not considered. Moreover, the resources are fixed servers at the edge or central cloud, while other mobile volunteer resources (e.g., vehicular cloud) are also not considered.

On the other hand, there are also existing researches on task offloading problem in vehicular cloud. The vehicular cloud consists of both vehicles in movement and statically parked ones. The authors in [34,35] propose to consider parked vehicles for task offloading and cooperative sensing. The parked vehicles can be treated as static cloud nodes, which are stable resources and the corresponding task offloading policy has been well examined [18]. Many other researches focus on utilizing vehicles in movement. In [36,37], the centralized task offloading and resource allocation problem in vehicular cloud are formulated based on a semi-Markov decision process (SDMP) approach in order to maximize the total long-term expected reward of VC in terms of both the income and cost of VC as well as the variability feature of available resources. However, centralized task scheduling requires large signaling overheads for updating vehicular states, and the SMDP-based approach suffers from high complexity, and thus cannot be applied in the scenarios of high density of vehicles. Alternatively, task offloading can be in distributed manner, i.e., each users makes task offloading decisions individually [38]. However, in this case, it still requires exchanges of vehicular states or requires a learning-based technique to obtain the performance of other vehicles [39]. To deal with the uncertainty of vehicle movements and improve service reliability, task replication policies are also proposed, which allows one task to be executed by several vehicles at the same time [40,41]. However, the drawback of the task replication approach is the high resource cost. Moreover, it is noteworthy that the above works focus on task offloading among vehicles in the VC by coordinating the resources of all participants in the VC while the utilization of remote cloud resources is not considered explicitly. In fact, if there are no sufficient available resources in the VC, the tasks will be simply offloaded to remote cloud.

Different from the mentioned papers, our paper focuses on the scenario that a MEC provider can execute its computing tasks locally or offload them to remote cloud or mobile volunteer nodes in the VC. In the literatures, few works consider offloading tasks from MEC to VC. For example, in [29], the authors consider a roadside cloudlet system, in which public service vehicles, such as buses, are leveraged as fog nodes to accomplish the tasks offloaded by the cloudlet. The authors propose an offloading strategy based on genetic algorithm, which enables the roadside cloudlet to spend the least incentive cost to motivate the bus and satisfy user experience. However, the paper does not consider resources of MEC servers. In [30], a combination of vehicular cloud, fixed central cloud, and cloudlet is proposed to expand the currently available resources. When infrastructure-based cloud does not satisfy the offloading requirements, a reliable worker node in VC will be selected to accomplish the offloading task. However, the authors consider offloading only one task and the cost for using the cloud resources is neglected. Hence, in this paper, we will provide an efficient task offloading strategy considering both the completion time of tasks and monetary cost for using the computing resources.

3. System Model and Problem Formulation

3.1. Scenario Description

We consider a scalable vehicle-assisted multi-access edge computing (SVMEC) paradigm as shown in Figure 1. SVMEC includes three parts: MEC provider, remote cloud, and vehicular cloud (VC). The MEC provider includes two main components: MEC server and vehicular cloud controller (VCC), which are deployed near the roadside base station. The MEC server virtualizes physical resources and acts as a cloud computing node. The VCC stores the status information (e.g., real-time location, available resources) of vehicles within the communication coverage of the MEC. IoT devices have compute-intensive tasks that cannot be accomplished themselves. So they access the MEC provider via access point and use the computing service of the MEC provider. Nevertheless, the computation resources of MEC are limited. When many IoT devices have requests, the resources of MEC may be insufficient.
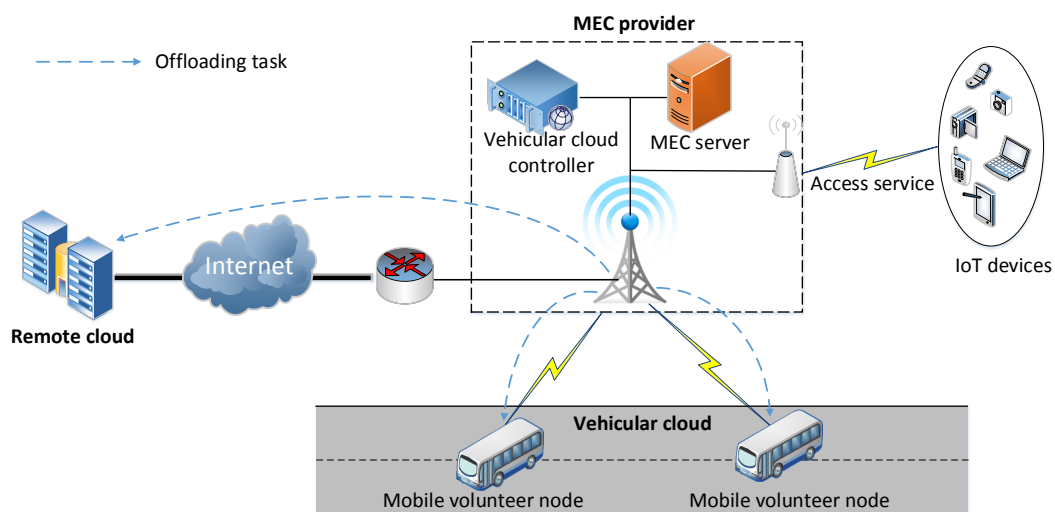

Figure 1. Illustration of scalable vehicle-assisted multi-access edge computing (SVMEC).

The resource capacity of MEC is extended by computing resources from a remote cloud. The remote cloud has rich computation resources, but it takes time to transfer data to the cloud node for task execution. Moreover, there is monetary cost charged to the MEC provider for the use of cloud resources.

On the other hand, the SVMEC transforms the conventional VC model and aims at exploiting the under-utilized resources of the vehicles in the VC. By this way, SVMEC can enhance its computing scalability by enabling different number of volunteering vehicles. We use the word "mobile volunteer node" (MVN) to indicate the vehicle that share its excess resource for the workload of the MEC. Specifically, we consider buses as potential candidates. We distinguish buses and other vehicles since buses have fixed mobility trajectories and strong periodicity, which are desirable for the MEC provider to offload the computing tasks. In addition, buses are public facilities and hence the issues of privacy and security are alleviated. The MEC provider offers some incentive to the bus owners to encourage sharing their excess resources. It is also noteworthy that compared to the rural environment, the SVMEC paradigm is better suited to the urban environment because of the following reasons. First, the vehicular network infrastructure is more developed and the mobility of vehicles is reasonably low in urban areas, so it enables a guaranteed and prolonged duration of the connection between vehicles and the infrastructure. In addition, the greater traffic density in urban areas provides more MVNs for task offloading.

The SVMEC works as a cyclical mechanism. In every cycle, the MEC provider has a number of tasks to be accomplished and the information of each computing node (e.g, available resources,

duration and data rate of connection with MEC). The MEC provider makes a strategy of node selection and resource allocation including two parts: (i) In which node should each task be placed, i.e., either MEC node, cloud node, or mobile volunteer node; and (ii) how much computing resource of the node should be allocated to each task? The main objective of the MEC provider is to minimize the total computation overhead in terms of the weighted-sum of task completion time and monetary cost for using cloud resources.

3.2. Computing Node and Task Model

Let $\mathcal{M} = \{1, 2, 3, ..., M\}$ denote the set of mobile volunteer nodes (MVNs) within the communication coverage of the MEC in one cycle. Let $\mathcal{J} = \{-1, 0, 1, 2, 3, ..., M\}$ denote the set of all computing nodes in the SVMEC. A computing node is indexed by $j \in \mathcal{J}$, in which $j = -1$ indicates the MEC server, $j = 0$ indicates the remote cloud server, and $j \in \{1, 2, ..., M\}$ indicates the jth MVN. Here, $\mathcal{M} \subset \mathcal{J}$.

Let $\mathcal{N} = \{1, 2, 3, ..., N\}$ denote the set of tasks of the MEC provider. Each computation task can be described by a tuple of two parameters as $T_i = (d_i, w_i), \forall i \in \mathcal{N}$, where d_i is the size of input data for the computation (in bits), w_i is the computation workload or intensity, i.e., the amount of computation to accomplish the task (in CPU cycles). We assume that task T_i is atomic and cannot be divided. Each computation task is executed on only one computing node, i.e., either MEC server, remote cloud server, or a MVN.

3.3. Local Computing on MEC

When task T_i is executed locally on MEC (i.e., $j = -1$), we denote $f_{i,-1}$ as the computational resource (in CPU cycles per second) allocated to task T_i by the MEC server. The completion time of task T_i on MEC can be calculated as

$$t_{i,-1} = \frac{w_i}{f_{i,-1}} \tag{1}$$

In this case, since the task is executed on the computing node owned by the MEC provider, there is no monetary cost for using the resource.

3.4. Offloading to Remote Cloud

In the case that the MEC provider decides to offload its task T_i to the remote cloud (i.e., $j = 0$), the completion time of the task is composed of the transmission time of the input data to the remote cloud, and the execution time of the task on the remote cloud. Since the size of the computation outcome is much smaller than that of the input data, we omit the time cost for sending the computation result from the remote cloud to the MEC [21,22,29]. Let $f_{i,0}$ and R_0 be the computational resource allocated to task T_i by the remote cloud and the uplink data rate of the wired link from the MEC to the remote cloud, respectively. The completion time of task T_i by offloading to the remote cloud can be calculated as

$$t_{i,0} = \frac{d_i}{R_0} + \frac{w_i}{f_{i,0}} \tag{2}$$

The MEC provider has to pay for cloud computing resource to accomplish its workload. Let δ_0^{comp} be the unit cost of computation for using remote cloud resource. The monetary cost of task T_i on the remote cloud node is calculated as

$$c_{i,0} = \delta_0^{comp} w_i \tag{3}$$

3.5. Offloading to Mobile Volunteer Node

In this case, the selection of a MVN for task offloading depends on the node's duration within the communication range of MEC.

We consider a two-dimensional space. Let (x_0, y_0) and D_r be the coordinate and the maximum communication radius of the base station, respectively. Since the MVNs (i.e., buses in the VC) have fixed mobility trajectories, the VCC can easily estimate the duration of each MVN $m \in \mathcal{M}$ within the communication coverage of MEC, denoted by Δt_m. Let $l_m = \{(x_1, y_1), (x_2, y_2), ..., (x_k, y_k)\}$ be the two-dimensional coordinate trajectory of the MVN $m \in \mathcal{M}$ such that the distance between the base station and the moving trajectory l_m satisfies

$$d_{m_i} = \sqrt{(x_i - x_0)^2 + (y_i - y_0)^2} \leq D_r, i = 1, 2, 3..., k \tag{4}$$

At each coordinate (x_i, y_i), the transmission rate can be obtained by

$$R_{m_i} = W_m \log_2 \left(1 + \frac{P_r d_{m_i}^{-\alpha}}{\sigma^2 + I}\right) \tag{5}$$

where W_m is the channel bandwidth, P_r is transmission power of base station, α is the path loss exponent, while σ^2 and I denote the addictive Gaussian noise and inter-cell interference, respectively.

The average transmission rate of the MEC to MVN m is as follows.

$$R_m = \frac{\sum\limits_{i=1}^{k} R_{m_i}}{k} \tag{6}$$

Since the limited computing resource and transient nature of MVNs, we assume that each MVN can receive at most one computation task at a given time. Let f_m be the computational capacity of the MVN m. The completion time of task T_i on MVN m can be calculated as

$$t_{i,m} = \frac{d_i}{R_m} + \frac{w_i}{f_m}, \qquad \forall m \in \mathcal{M} \tag{7}$$

In this case, the completion time of the task T_i must not exceed the duration of MVN m within the communication range of MEC. Hence, $t_{i,m} \leq \Delta t_m, \forall m \in \mathcal{M}$.

If the MVN m accomplishes the offloaded task, it will receive some incentives offered by the MEC provider to encourage the willingness of MVNs to participate in sharing their resources. Let δ_m^{comp} be the incentive unit cost of computation for using resources of the MVN m. We assume that $\delta_m^{comp} < \delta_0^{comp}$ since it motivates the MEC provider to utilize MVN's resources in order to reduce the cost of cloud resources. The incentive cost of task T_i on the MVN m can be obtained by

$$c_{i,m} = \delta_m^{comp} w_i, \qquad \forall m \in \mathcal{M} \tag{8}$$

3.6. Problem Formulation

The focus of our work is on the perspective of the MEC provider, who aims to minimize the total computation overhead in terms of the weighted-sum of task completion time and monetary cost for using cloud resources. We define the node selection profile as $X = \{x_{i,j} | i \in \mathcal{N}, j \in \mathcal{J}\}$, in which $x_{i,j} = 1$ means that task i is placed on node j, and $x_{i,j} = 0$ otherwise. We denote the resource allocation profile as $F = \{f_{i,j} | i \in \mathcal{N}, j \in \mathcal{J}\}$. The objective function is defined as follows.

$$G(X, F) = \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{J}} x_{i,j} \left(\lambda_i^t t_{i,j} + \lambda_i^c c_{i,j}\right), \tag{9}$$

where $\lambda_i^t, \lambda_i^c \in \{0, 1\}$, and $\lambda_i^t + \lambda_i^c = 1$ are the weighted parameters of the completion time and resource cost of task T_i, respectively. They represent the MEC provider's preference on task completion time and resource cost, respectively.

For a given node selection profile X, and resource allocation profile F, the optimization problem of joint node selection and resource allocation (JNSRA) can be expressed as follows.

$$\min_{X,F} \quad G(X,F) \tag{10}$$

s.t.

$$x_{i,j} \in \{0,1\}, \quad \forall i \in \mathcal{N}, j \in \mathcal{J}, \tag{10a}$$

$$\sum_{j \in \mathcal{J}} x_{i,j} = 1, \quad \forall i \in \mathcal{N}, \tag{10b}$$

$$\sum_{i \in \mathcal{N}} x_{i,j} \leq 1, \quad \forall j \in \mathcal{M}, \tag{10c}$$

$$\sum_{i \in \mathcal{N}} x_{i,j} t_{i,j} \leq \Delta t_j, \quad \forall j \in \mathcal{M}, \tag{10d}$$

$$f_{i,j} > 0, \quad \forall i \in \mathcal{N}_j, j \in \mathcal{J}, \tag{10e}$$

$$\sum_{i \in \mathcal{N}_j} f_{i,j} \leq f_j^{max}, \quad \forall j \in \mathcal{J}. \tag{10f}$$

The constraint in (10a) denotes X is a binary vector. The constraint in (10b) ensures that each computation task is executed on only one computing node, i.e., either MEC server, remote cloud server, or a MVN. The constraint in (10c) and (10d) state that each MVN can receive at most one task at a given time, and the selected MVN accomplishes its assigned task within its duration of connection with MEC. The constraint in (10e) states that each computing node must allocate a positive computing resource to each task assigned to it, where $\mathcal{N}_j = \{i \in \mathcal{N} | x_{i,j} = 1\}$ denotes the set of tasks assigned to the computing node. The constraint in (10f) makes sure that the total computing resources of a computing node allocated to its assigned tasks does not exceed its maximum capacity, denoted by f_j^{max}.

4. Joint Node Selection and Resource Allocation Solution

The JNSRA is considered as a Mixed Integer Nonlinear Programming (MINLP) problem since the node selection profile X is a binary vector and the resource allocation profile F is a continuous vector. Hence, the JNSRA problem is NP-hard [42]. In this section, we give the solution to the above problem, which is related with both aspects of node selection and resource allocation. Here, we adopt alternative optimization techniques and consider two sub-problems as follows.

(i) *Computing resource allocation problem:* When the strategy of node selection is given, i.e., $X = X^0$, the original problem in (10) is a convex problem with respect to F. Then we can obtain the optimal solution F^* by using the Karush-Kuhn-Tucker (KKT) conditions.

(ii) *Node selection problem:* Based on the solution F^*, the sub-problem $G(X, F^*)$ is transferred to 0–1 integer programming problem with respect to X. By adopting branch-and-bound algorithm, the optimal solution X^* can be obtained.

4.1. Computing Resource Allocation Problem

Theorem 1. *Given* $X = X^0$, *the original optimization problem in (10) with respect to F is a convex optimization problem.*

Proof of Theorem 1. Given $X = X^0$, the objective function becomes function of F. After omitting the terms of the objective function that are independent with F, we can rewrite the JNSRA problem in (10) as follows.

$$\min_{F} \quad \sum_{j \in \mathcal{J}} \sum_{i \in \mathcal{N}_j} \frac{\lambda_i^t w_i}{f_{i,j}} \tag{11}$$

$$\text{s.t.} (10e), (10f).$$

Notice that the feasible solution set of the above problem is convex. The remaining task is to show the convexity of the objective function. Denote the objective function in (11) as $\Psi(F)$, we calculate the second-order derivatives or Hessian matrix $\nabla^2 \Psi(F)$, whose elements are as follows.

$$\frac{\partial^2 \Psi(F)}{\partial f_{i,j}^2} = \frac{2\lambda_i^t w_i}{f_{i,j}^3}, \forall i \in \mathcal{N}_j, j \in \mathcal{J} \tag{12}$$

$$\frac{\partial^2 \Psi(F)}{\partial f_{i,j} \partial f_{m,n}} = 0, \forall (i,j) \neq (m,n) \tag{13}$$

It is easy to check that $\nabla^2 \Psi(F) \succeq 0$. Hence, the Hessian matrix is a positive semidefinite matrix. We conclude that this problem is a convex optimization problem. $\square$

Since the problem in (11) is a convex problem, the optimal solution can be obtained by using KKT conditions. Let $\nu = \{\nu_j\}_{j \in \mathcal{J}}$ be the Lagrange multiplier vector associated with the second constraint. The Lagrange function is as follows.

$$L(F, \nu) = \sum_{j \in \mathcal{J}} \sum_{i \in \mathcal{N}_j} \frac{\lambda_i^t w_i}{f_{i,j}} + \sum_{j \in \mathcal{J}} \nu_j \left(\sum_{i \in \mathcal{N}_j} f_{i,j} - f_j^{max} \right) \tag{14}$$

From the KKT conditions, we have

$$\nabla L(F, \nu) = 0 \tag{15}$$

$$\sum_{i \in \mathcal{N}_j} f_{i,j} - f_j^{max} = 0 \tag{16}$$

By setting the first-order derivative of the Lagrange function with respect to $f_{i,j}$ equal to zero as in (15), we can obtain the optimal $f_{i,j}^*$ as follows.

$$f_{i,j}^* = \sqrt{\frac{\lambda_i^t w_i}{\nu_j}} \tag{17}$$

Substituting (17) into (16), we can obtain the optimal ν_j^* as follows.

$$\nu_j^* = \left(\frac{\sum\limits_{i \in \mathcal{N}_j} \sqrt{\lambda_i^t w_i}}{f_j^{max}} \right)^2 \tag{18}$$

Finally, substituting (18) back to (17), the optimal computing resource allocation $f_{i,j}^*$ is given by

$$f_{i,j}^* = \frac{f_j^{max} \sqrt{\lambda_i^t w_i}}{\sum\limits_{i \in \mathcal{N}_j} \sqrt{\lambda_i^t w_i}} \tag{19}$$

4.2. Node Selection Problem

According to the above discussion, we have obtained the optimal resource allocation F^*. Based on this, the original problem in (10) is rewritten as follows.

$$\min_X \quad \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{J}} x_{i,j} \left(\lambda_i^t t_{i,j} + \lambda_i^c c_{i,j} \right) \tag{20}$$

$$\text{s.t.}(10a), (10b), (10c), (10d).$$

where $t_{i,j}$ and $c_{i,j}$ can be given as

$$t_{i,j} = \begin{cases} \frac{w_i}{f_{i,-1}^*}, & \text{if } j = -1 \\[2mm] \frac{d_i}{R_0} + \frac{w_i}{f_{i,0}^*}, & \text{if } j = 0 \\[2mm] \frac{d_i}{R_j} + \frac{w_i}{f_m}, & \text{if } j = 1, 2, ...M \end{cases} \tag{21}$$

$$c_{i,j} = \begin{cases} 0, & \text{if } j = -1 \\[2mm] \delta_0^{comp} w_i, & \text{if } j = 0 \\[2mm] \delta_j^{comp} w_i, & \text{if } j = 1, 2, ...M \end{cases} \tag{22}$$

The above problem is a 0–1 integer programming problem with respect to X. Hence, the problem can be solved by using the branch-and-bound algorithm. The optimal result represents node selection and resource allocation strategy of the MEC provider.

5. Performance Evaluation

5.1. Simulation Settings

In this simulation, we evaluate the performance of our proposed scheme, which includes MEC, remote cloud, and MVNs to collaboratively execute compute-intensive or data-intensive tasks.

For MVNs, we use the Seattle bus trace [43], which was also adopted in some other recent researches [29,44,45]. The Seattle bus trace includes the actual movement of approximately 1200 city buses on their normal routes in Seattle, Washington metropolitan area, USA for several weeks. For our experiment, we choose a sampled map from 23:30 p.m. to 23:40 p.m. of date 30 October in the dataset. Each record includes the bus route ID and the location in terms of the x-coordinate and y-coordinate along with time. The MEC base station is assumed to be located at the center of the map of the sampled dataset and the communication coverage is assumed to be 2 km. Due to the fixed trajectories of the buses, the number and duration of the buses within the communication range of the MEC in the cycle time can be estimated. The average data rate of each bus can also be calculated as in Section 3.5. Among these buses, the MEC provider can choose a number of MVNs for task offloading.

We use the following simulation settings for all experiments. For the computation task T_i, we adopt the face recognition application in [46,47], where the input data size is $d_i = 2000$ KB, and the required number of CPU cycles is $w_i = 1000$ Megacycles. For the computing nodes, the computational capacity of the MEC server and remote cloud server are set as 4 GHz and 10 GHz, respectively. The computational capacity f_m of the MVN $m \in \mathcal{M}$ is randomly assigned from the set {0.5, 0.8, 1.0} GHz. The uplink data rate from MEC to remote cloud is $R_0 = 100$ Mbps. For wireless communication from MEC to MVNs, we consider the system using orthogonal frequency-division multiple access (OFDMA) scheme [48]. We assume that there are 20 equal sub-bands. Each MVN is assigned to one sub-band. Thus, there are at most 20 MVNs connected to MEC at the same time. The transmission

power of the base station is $P_r = 46$ dBm and the channel bandwidth is $W = 10$ MHz. In addition, there is no interference, we assume the path loss exponent is $\alpha = 4$, and the background noise power is $\sigma^2 = -100$ dBm [21]. From the dataset, the duration of connection with MEC of each MVN ranges from 20 s to 300 s, and the achieved average data rate from MEC to each MVN ranges from 5 Mbps to 12 Mbps. The unit cost of computation for using cloud resources and MVNs' resources are set as 0.9 \$/gigacycles and 0.5 \$/gigacycles, respectively. The weighted parameters of the task completion time and monetary cost for using computing resources are both 0.5, i.e., $\lambda_i^t = \lambda_i^c = 0.5, \forall i \in \mathcal{N}$. Unless otherwise stated, the default number of tasks and MVNs are set as $N = 40$ and $M = 10$, respectively.

5.2. Simulation Results

In the following experiments, we evaluate the offloading performance in terms of total computation overhead, which is the weighted sum of task completion time and monetary cost for using computing resources. The proposed scheme is compared with other strategies as follows.

- MEC only scheme: The system includes only MEC server and all computation tasks are executed locally on MEC server. In this case, there is no monetary cost for using computing resources. The total computation overhead considers only the completion time of tasks. The resource allocation strategy in Section 4.1 is applied.
- MEC + Cloud scheme: The system combines MEC server and remote cloud server. The proposed joint node selection and resource allocation strategy is applied to allocate each computation task to the MEC server or the remote cloud server in order to achieve optimal total computation overhead.
- Random offloading in MEC + Cloud + MVNs scheme (RO_ECM): The system includes MEC server, remote cloud server, and MVNs. Each computation task is randomly assigned to only one computing node, i.e., either the MEC server, the remote cloud server or a MVN with equal probability such that the resource constraint and duration constraint of the selected computing node are satisfied. The resource allocation is given by the strategy in Section 4.1.

In Figure 2, we vary the number of tasks from 10 to 100 and evaluate the offloading performance. Figure 2a compares the performances of our proposed approach and three other schemes. It is shown that when the number of tasks increases, the total computation overhead of all schemes increases, especially in MEC only scheme. Due to the resource bottleneck of the MEC server, the completion time of tasks of MEC only scheme significantly increases and its effect dominates the total computation overhead. Hence, the performance of MEC only scheme is the worst in most cases (except in case the number of tasks is small enough for MEC to handle, e.g., $N = 10$). In MEC + Cloud scheme, although there is monetary cost for using the cloud resources, the remote cloud can help the overloaded MEC to execute the increasing number of tasks so that the completion time of tasks can be significantly reduced. Hence, the performance of MEC + Cloud scheme becomes much better than that of MEC only scheme when the number of tasks gets larger. In RO_ECM scheme, besides MEC and remote cloud, the MEC provider can also utilize MVNs. However, in this case, randomly assigning tasks to computing nodes with equal probability will not provide a stable and optimal solution. From Figure 2a, when the number of tasks is $N = 10$, the performance of RO_ECM scheme is even worse than that of MEC only scheme. Meanwhile, our proposed scheme can always achieve the best total computation overhead because it comprehensively considers both the completion time and monetary cost for using computing resources on MEC, remote cloud and MVNs. Figure 2b shows the task distribution on MEC, remote cloud and MVNs of the proposed scheme under different number of tasks. It can be observed that when the number of task is small ($N = 10$), the percentage of tasks allocated to MVNs is equal to 0. It means that the tasks can be executed optimally by using MEC and remote cloud and without using MVNs. It is the reason why the performance of our proposed scheme is the same as that of MEC + Cloud scheme in case $N = 10$ as shown in Figure 2a. When the number of tasks increases from 10 to 40 tasks, the percentage of tasks allocated to MVNs increases until the maximum number

of MVNs ($M = 10$) is utilized (at $N = 40$ tasks). Meanwhile, the percentages of tasks allocated to MEC and remote cloud decrease. When the number of tasks is greater than 40 tasks, the percentage of tasks allocated to MVNs decreases while the percentages of tasks allocated to MEC and remote cloud increase. It is because there are no more MVNs that can receive the offloaded tasks and the tasks will be executed on MEC or remote cloud. Moreover, it can also be seen that the percentage of tasks allocated to remote cloud is greater than the percentage of tasks allocated to MEC when the number of tasks increases.
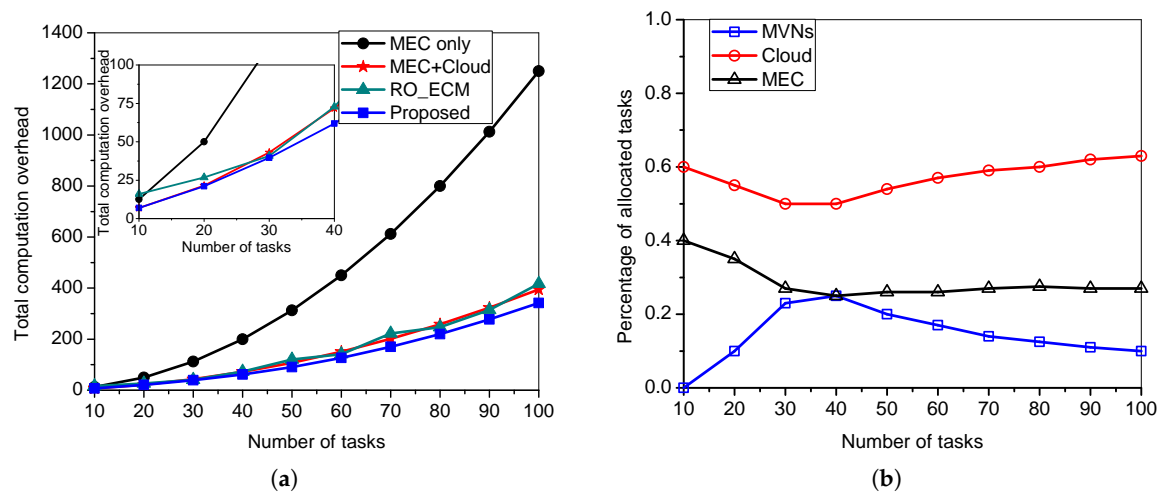
Figure 2. Performance evaluation under different number of tasks. (**a**) Comparison of our proposed scheme and three other schemes; (**b**) Task distribution of the proposed scheme.

In Figures 3 and 4, we evaluate the offloading performance with respect to the computation task's profiles in terms of input data size d_i's and computation intensity w_i's, respectively. Figure 3a shows that when the input data size d_i increases, the total computation overhead of MEC only scheme remains unchanged since there is no offloading. However, the performance of MEC only scheme is the worst due to the high completion time of tasks on MEC. RO_ECM scheme performs better than MEC + Cloud scheme when the input data size d_i is small (e.g., in case $d_i = 1$ MB and $d_i = 2$ MB), however, the performance of RO_ECM scheme significantly decreases and becomes worse than that of MEC + Cloud scheme when the input data size increases. Meanwhile, our proposed scheme performs the best when the input data size is small. When the input data size d_i is large enough (9 MB in this case), the total computation overhead of our proposed scheme can reach that of MEC + Cloud scheme. It is caused by the gradual decrease to 0 of the percentage of tasks allocated to MVNs when the input data size increases as seen in Figure 3b. The reason for this is that by increasing the input data size, the transmission time of input data to MVNs significantly increases. Due to the unstable connection with MEC of MVNs, a computation task with small data size is more preferable to be offloaded to MVNs than one with high data size. Instead of being offloaded to MVNs, the tasks with high data size can be executed by using MEC and remote cloud. From Figure 4a, we can observe that the total computation overhead of all schemes increases with the number of CPU cycles w_i required to accomplish the tasks. Similarly, the performance of MEC only scheme is the worst due to the resource bottleneck of the MEC server. In general, RO_ECM performs better than MEC + Cloud scheme since the MVNs can help alleviate the workload of MEC and remote cloud as the computation intensity of tasks increases. Meanwhile, our proposed scheme can always achieve lower total computation overhead than both MEC + cloud scheme and RO_ECM scheme by performing optimal task distribution on MEC, remote cloud and MVNs as can be seen in Figure 4b. It shows that the proposed scheme utilizes the maximal number of MVNs (i.e., $M = 10$) when the computation intensity of task increases.
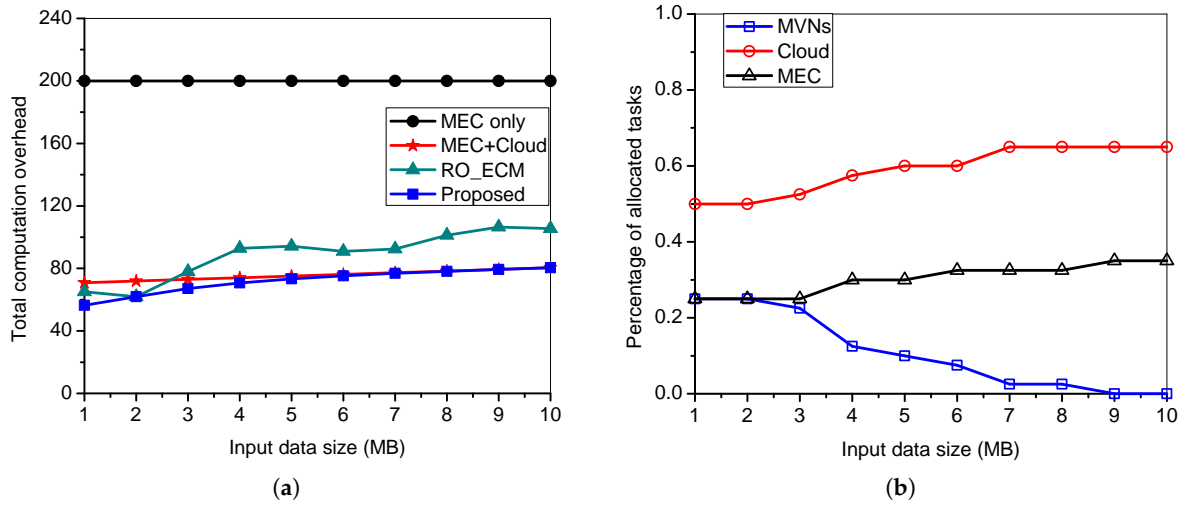
Figure 3. Performance evaluation under different tasks' input data sizes. (**a**) Comparison of our proposed scheme and three other schemes; (**b**) Task distribution of the proposed scheme.
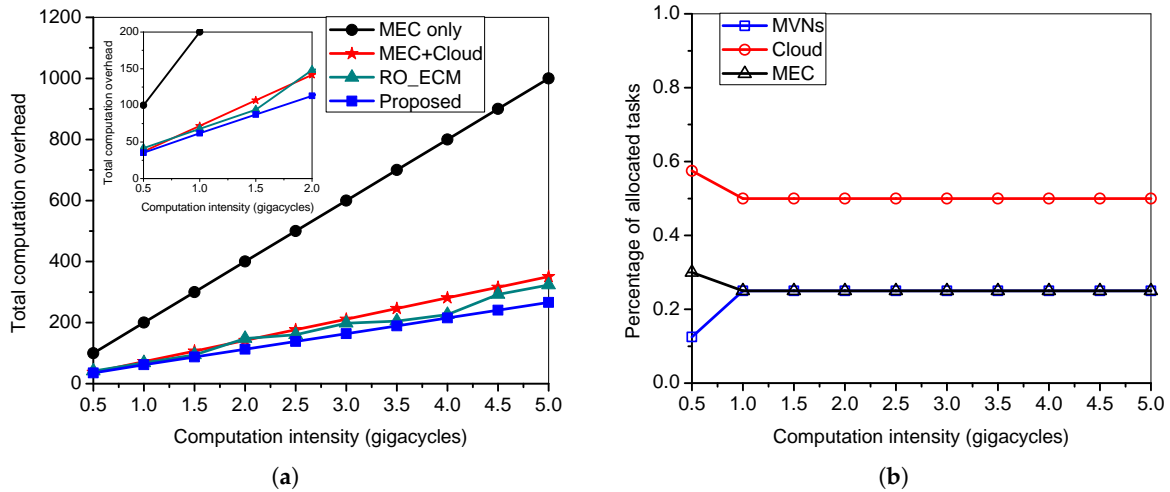

Figure 4. Performance evaluation under different tasks' computation intensities. (**a**) Comparison of our proposed scheme and three other schemes; (**b**) Task distribution of the proposed scheme.

We now conduct an experiment to examine the effect of the number of MVNs M on the total computation overhead of our proposed scheme. Figure 5 shows that the total computation overhead declines with the increase of number of MVNs. It is because the increasing number of MVNs brings more options for the MEC provider to offload the computation tasks to MVNs. When the number of MVNs M increases from 2 to 10, the total computation overhead decreases dramatically. After that, it slightly reduces to a stable value.

Finally, by varying the weighted parameter of the completion time λ_i^t from 0.1 to 0.9 with a step deviation of 0.1 and setting the weighted parameter of resource cost as $\lambda_i^c = 1 - \lambda_i^t$, we can also analyze the effect of the weighted parameters on the completion time of all tasks and monetary cost for using computing resources. It can be seen from Figure 6 that the changing trends in terms of the completion time and resource cost are the same in both cases where the number of tasks is $N = 40$ and $N = 50$. It is that the completion time of tasks decreases with the increase of monetary cost for using computing resources when λ_i^t increases. In addition, when $N = 50$, the completion time of tasks and monetary cost is higher than in the case when $N = 40$. It is also highly noticeable that the completion time of tasks and the monetary cost significantly decreases and increases, respectively, when the value of λ_i^t

changes from 0.1 to 0.2. Therefore, selection of the weighted parameters plays an important role in the achieved completion time of tasks and monetary cost for using computing resources.
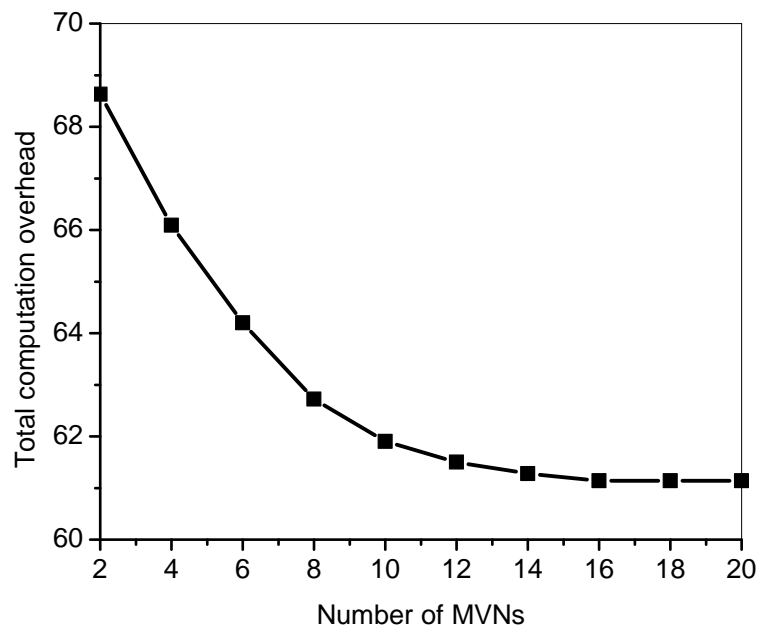

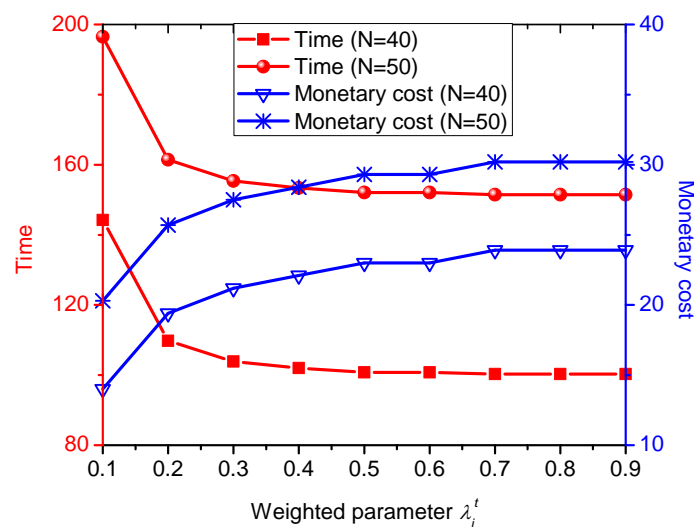**Figure 5.** Effect of the number of mobile volunteer nodes (MVNs) on the total computation overhead.

Figure 6. Effect of the weighted parameters on the completion time of tasks and monetary cost for using computing resources.

6. Conclusions

In this paper, we have proposed an SVMEC paradigm, in which the MEC capacity is expanded by renting resources from a remote cloud and vehicular cloud (VC) so as to efficiently handle the steep rise of task requests from IoT devices. In our scenario, we use buses in the VC as candidate mobile volunteer nodes that share their under-utilized resources for tasks offloaded from the MEC provider. The MEC provider makes a strategy to select proper computing nodes and amount of computing resources for the task requests of IoT devices. We present the joint node selection and resource allocation problem with the aim of minimizing the total computation overhead in terms of the weighted-sum of task completion time and monetary cost for using computing resources. We formulate

the problem as a MINLP and give the solution by alternative optimization techniques. Finally, a lot of simulations have been conducted and their results prove that our proposed scheme can achieve better performances than the existing schemes. In future work, we will extend our model considering the various QoS requirements of task requests. Moreover, it is noteworthy that in addition to buses, the MVNs can also be extended to other types of vehicle (e.g., cars, trucks, etc.), however, further investigation should be conducted in order to deal with various mobility trajectories of vehicles.

Author Contributions: Methodology, X.-Q.P.; supervision, E.-N.H.; validation, X.-Q.P.; writing—original draft, X.-Q.P.; writing—review and editing, X.-Q.P., T.-D.N., and V.N.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

IoT	Internet of Things
MCC	Mobile cloud computing
MEC	Multi-access edge computing
SVMEC	Scalable vehicle-assisted multi-access edge computing
VC	Vehicular cloud
VCC	vehicular cloud controller
MVN	Mobile volunteer node
QoS	Quality of service
JNSRA	Joint node selection and resource allocation

References

1. Lucero, S. IoT Platforms: Enabling the Internet of Things. 2016. Available online: https://cdn.ihs.com/www/pdf/enabling-IOT.pdf (accessed on 13 October 2018).
2. Gatouillat, A.; Badr, Y.; Massot, B.; Sejdić, E. Internet of Medical Things: A Review of Recent Contributions Dealing With Cyber-Physical Systems in Medicine. *IEEE Internet Things J.* **2018**, *5*, 3810–3822. [CrossRef]
3. Połap, D.; Winnicka, A.; Serwata, K.; Kęsik, K.; Woźniak, M. An Intelligent System for Monitoring Skin Diseases. *Sensors* **2018**, *18*, 2552. [CrossRef]
4. Malūkas, U.; Maskeliūnas, R.; Damaševičius, R.; Woźniak, M. Real Time Path Finding for Assisted Living Using Deep Learning. *J. Univ. Comput. Sci.* **2018**, *24*, 475–487.
5. Gurvey, S. IoT and Intelligent Transportation. 2015. Available online: https://newsroom.cisco.com/feature-content?articleId=1601746 (accessed on 18 December 2018).
6. Salman, M.A.; Ozdemir, S.; Celebi, F.V. Fuzzy Traffic Control with Vehicle-to-Everything Communication. *Sensors* **2018**, *18*, 368. [CrossRef]
7. Ito, S.; Hiratsuka, S.; Ohta, M.; Matsubara, H.; Ogawa, M. Small Imaging Depth LIDAR and DCNN-Based Localization for Automated Guided Vehicle. *Sensors* **2018**, *18*, 177. [CrossRef]
8. Połap, D.; Kęsik, K.; Książek, K.; Woźniak, M. Obstacle Detection as a Safety Alert in Augmented Reality Models by the Use of Deep Learning Techniques. *Sensors* **2017**, *17*, 2803. [CrossRef]
9. Qi, B.; Shi, H.; Zhuang, Y.; Chen, H.; Chen, L. On-Board, Real-Time Preprocessing System for Optical Remote-Sensing Imagery. *Sensors* **2018**, *18*, 1328. [CrossRef]
10. Zhu, Q.; Xiao, C.; Hu, H.; Liu, Y.; Wu, J. Multi-Sensor Based Online Attitude Estimation and Stability Measurement of Articulated Heavy Vehicles. *Sensors* **2018**, *18*, 212. [CrossRef]
11. Garrido Abenza, P.P.; Malumbres, M.P.; Piñol, P.; López-Granado, O. Source Coding Options to Improve HEVC Video Streaming in Vehicular Networks. *Sensors* **2018**, *18*, 3107. [CrossRef]

12. Nguyen, T.D.T.; Nguyen, T.D.; Nguyen, V.D.; Pham, X.Q.; Huh, E.N. Cost-Effective Resource Sharing in an Internet of Vehicles-Employed Mobile Edge Computing Environment. *Symmetry* **2018**, *10*, 594. [CrossRef]

13. Zhu, W.; Li, D.; Saad, W. Multiple Vehicles Collaborative Data Download Protocol via Network Coding. *IEEE Trans. Veh. Technol.* **2015**, *64*, 1607–1619. [CrossRef]

14. Froiz-Míguez, I.; Fernández-Caramés, T.M.; Fraga-Lamas, P.; Castedo, L. Design, Implementation and Practical Evaluation of an IoT Home Automation System for Fog Computing Applications Based on MQTT and ZigBee-WiFi Sensor Nodes. *Sensors* **2018**, *18*, 2660. [CrossRef] [PubMed]

15. Yang, C.; Shen, W.; Wang, X. The Internet of Things in Manufacturing: Key Issues and Potential Applications. *IEEE Syst. Man Cybern. Mag.* **2018**, *4*, 6–15. [CrossRef]

16. Fingas, M.; Brown, C.E. A Review of Oil Spill Remote Sensing. *Sensors* **2018**, *18*, 91. [CrossRef] [PubMed]

17. Dinh, H.T.; Lee, C.; Niyato, D.; Wang, P. A survey of mobile cloud computing: Architecture, applications, and approaches. *Wirel. Commun. Mob. Comput.* **2013**, *13*, 1587–1611. [CrossRef]

18. Mach, P.; Becvar, Z. Mobile Edge Computing: A Survey on Architecture and Computation Offloading. *IEEE Commun. Surv. Tutor.* **2017**, *19*, 1628–1656. [CrossRef]

19. Bonomi, F.; Milito, R.; Zhu, J.; Addepalli, S. Fog Computing and Its Role in the Internet of Things. In Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing, MCC'12, Helsinki, Finland, 17 August 2012; ACM: New York, NY, USA, 2012; pp. 13–16.

20. Satyanarayanan, M.; Bahl, P.; Caceres, R.; Davies, N. The Case for VM-Based Cloudlets in Mobile Computing. *IEEE Pervasive Comput.* **2009**, *8*, 14–23. [CrossRef]

21. Guo, H.; Liu, J. Collaborative Computation Offloading for Multiaccess Edge Computing Over Fiber–Wireless Networks. *IEEE Trans. Veh. Technol.* **2018**, *67*, 4514–4526. [CrossRef]

22. Guo, H.; Liu, J.; Qin, H. Collaborative Mobile Edge Computation Offloading for IoT over Fiber-Wireless Networks. *IEEE Netw.* **2018**, *32*, 66–71. [CrossRef]

23. Pham, X.Q.; Man, N.D.; Tri, N.D.T.; Thai, N.Q.; Huh, E.N. A cost- and performance-effective approach for task scheduling based on collaboration between cloud and fog computing. *Int. J. Distrib. Sens. Netw.* **2017**, *13*, 1550147717742073. [CrossRef]

24. Zhao, T.; Zhou, S.; Guo, X.; Zhao, Y.; Niu, Z. A Cooperative Scheduling Scheme of Local Cloud and Internet Cloud for Delay-Aware Mobile Cloud Computing. In Proceedings of the 2015 IEEE Globecom Workshops (GC Wkshps), San Diego, CA, USA, 6–10 December 2015; pp. 1–6.

25. Whaiduzzaman, M.; Sookhak, M.; Gani, A.; Buyya, R. A survey on vehicular cloud computing. *J. Netw. Comput. Appl.* **2014**, *40*, 325–344. [CrossRef]

26. Abdelhamid, S.; Hassanein, H.S.; Takahara, G. Vehicle as a resource (VaaR). *IEEE Netw.* **2015**, *29*, 12–17. [CrossRef]

27. Rob van der Meulen, J.R. Gartner Says By 2020, a Quarter Billion Connected Vehicles Will Enable New In-Vehicle Services and Automated Driving Capabilities. 2015. Available online: https://www.gartner.com/newsroom/id/2970017 (accessed on 22 December 2018).

28. Hou, X.; Li, Y.; Chen, M.; Wu, D.; Jin, D.; Chen, S. Vehicular Fog Computing: A Viewpoint of Vehicles as the Infrastructures. *IEEE Trans. Veh. Technol.* **2016**, *65*, 3860–3873. [CrossRef]

29. Ye, D.; Wu, M.; Tang, S.; Yu, R. Scalable Fog Computing with Service Offloading in Bus Networks. In Proceedings of the 2016 IEEE 3rd International Conference on Cyber Security and Cloud Computing (CSCloud), Beijing, China, 25–27 June 2016; pp. 247–251.

30. Zhang, H.; Zhang, Q.; Du, X. Toward Vehicle-Assisted Cloud Computing for Smartphones. *IEEE Trans. Veh. Technol.* **2015**, *64*, 5610–5618. [CrossRef]

31. Tanzil, S.M.S.; Gharehshiran, O.N.; Krishnamurthy, V. Femto-Cloud Formation: A Coalitional Game-Theoretic Approach. In Proceedings of the 2015 IEEE Global Communications Conference (GLOBECOM), San Diego, CA, USA, 6–10 December 2015; pp. 1–6.

32. Oueis, J.; Strinati, E.C.; Barbarossa, S. Small cell clustering for efficient distributed cloud computing. In Proceedings of the 2014 IEEE 25th Annual International Symposium on Personal, Indoor, and Mobile Radio Communication (PIMRC), Washington, DC, USA, 2–5 September 2014; pp. 1474–1479.

33. Oueis, J.; Strinati, E.C.; Sardellitti, S.; Barbarossa, S. Small Cell Clustering for Efficient Distributed Fog Computing: A Multi-User Case. In Proceedings of the 2015 IEEE 82nd Vehicular Technology Conference (VTC2015-Fall), Boston, MA, USA, 6–9 September 2015; pp. 1–5.

34. Liu, N.; Liu, M.; Lou, W.; Chen, G.; Cao, J. PVA in VANETs: Stopped cars are not silent. In Proceedings of the 2011 IEEE INFOCOM, Shanghai, China, 10–15 April 2011; pp. 431–435.

35. Eckhoff, D.; Sommer, C.; German, R.; Dressler, F. Cooperative Awareness at Low Vehicle Densities: How Parked Cars Can Help See through Buildings. In Proceedings of the 2011 IEEE Global Telecommunications Conference—GLOBECOM 2011, Kathmandu, Nepal, 5–9 December 2011; pp. 1–6.

36. Zheng, K.; Meng, H.; Chatzimisios, P.; Lei, L.; Shen, X. An SMDP-Based Resource Allocation in Vehicular Cloud Computing Systems. *IEEE Trans. Ind. Electron.* **2015**, *62*, 7920–7928. [CrossRef]

37. Wang, Z.; Zhong, Z.; Ni, M. Application-Aware Offloading Policy Using SMDP in Vehicular Fog Computing Systems. In Proceedings of the 2018 IEEE International Conference on Communications Workshops (ICC Workshops), Kansas City, MO, USA, 20–24 May 2018; pp. 1–6.

38. Feng, J.; Liu, Z.; Wu, C.; Ji, Y. AVE: Autonomous Vehicular Edge Computing Framework with ACO-Based Scheduling. *IEEE Trans. Veh. Technol.* **2017**, *66*, 10660–10675. [CrossRef]

39. Sun, Y.; Guo, X.; Zhou, S.; Jiang, Z.; Liu, X.; Niu, Z. Learning-Based Task Offloading for Vehicular Cloud Computing Systems. *arXiv* **2018**, arXiv:1804.00785.

40. Jiang, Z.; Zhou, S.; Guo, X.; Niu, Z. Task Replication for Deadline-Constrained Vehicular Cloud Computing: Optimal Policy, Performance Analysis, and Implications on Road Traffic. *IEEE Internet Things J.* **2018**, *5*, 93–107. [CrossRef]

41. Sun, Y.; Song, J.; Zhou, S.; Guo, X.; Niu, Z. Task Replication for Vehicular Edge Computing: A Combinatorial Multi-Armed Bandit based Approach. *arXiv* **2018**, arXiv:1807.05718.

42. Pochet, Y.; Wolsey, L.A. *Production Planning by Mixed Integer Programming*; Springer Series in Operations Research and Financial Engineering; Springer: Berlin/Heidelberg, Germany, 2006.

43. Jetcheva, J.G.; Hu, Y.C.; PalChaudhuri, S.; Saha, A.K.; Johnson, D.B. Design and evaluation of a metropolitan area multitier wireless ad hoc network architecture. In Proceedings of the 2003 Fifth IEEE Workshop on Mobile Computing Systems and Applications, Monterey, CA, USA, 9–10 October 2003; pp. 32–43.

44. Dias, D.S.; Costa, L.H.M.; de Amorim, M.D. Data offloading capacity in a megalopolis using taxis and buses as data carriers. *Veh. Commun.* **2018**, *14*, 80–96. [CrossRef]

45. Zheng, H.; Chang, W.; Wu, J. Traffic flow monitoring systems in smart cities: Coverage and distinguishability among vehicles. *J. Parallel Distrib. Comput.* **2018**, in press. [CrossRef]

46. Soyata, T.; Muraleedharan, R.; Funai, C.; Kwon, M.; Heinzelman, W. Cloud-Vision: Real-time face recognition using a mobile-cloudlet-cloud acceleration architecture. In Proceedings of the 2012 IEEE Symposium on Computers and Communications (ISCC), Cappadocia, Turkey, 1–4 July 2012; pp. 000059–000066.

47. Chen, X.; Jiao, L.; Li, W.; Fu, X. Efficient Multi-User Computation Offloading for Mobile-Edge Cloud Computing. *IEEE/ACM Trans. Netw.* **2016**, *24*, 2795–2808. [CrossRef]

48. Bazzi, A.; Zanella, A.; Masini, B.M. An OFDMA-Based MAC Protocol for Next-Generation VANETs. *IEEE Trans. Veh. Technol.* **2015**, *64*, 4088–4100. [CrossRef]