Decentralized Configuration Protocols for Low-Cost Offloading from Multiple Edges to Multiple Vehicular Fogs

Li-Hsing Yen, Jui-Chung Hu, Ying-Dar Lin
Department of Computer Science, College of Computer Science
National Chiao Tung University, Hsinchu, Taiwan.
Email: lhyen@nctu.edu.tw, tony84822@gmail.com, ydlin@cs.nctu.edu.tw
Binayak Kar
Department of Computer Science and Information Engineering
National Taiwan University of Science and Technology, Taipei, Taiwan
Email: bkar@mail.ntust.edu.tw

***Abstract*—A vehicular-fog (VF) system as an emerging platform consists of electric vehicles with computing resources that are mostly under-utilized. This paper considers a two-tier federated Edge and Vehicular-Fog (EVF) system, where edge systems may partially offload user traffic to nearby VFs for potential cost reduction. Offloading configuration is to determine the ratios and targets of offloading traffic for maximal cost reduction, which is formulated as a mixed integer programming problem in this paper. We first present a decentralized offloading configuration protocol (DOCP) for an individual edge system to set up its own offloading configuration. We then propose a matching protocol among multiple edge systems to resolve resource contention when they simultaneously request resources from the same VF. Simulation results show that the proposed approach can leverage the heterogeneity of cost and capacity between edge systems and VFs. The proposed protocol outperforms greedy approaches by at most 40% and is comparable to a centralized off-line approach that is based on Particle Swarm Optimization.**

***Index Terms*—MEC, vehicular-fog, offloading, matching, network optimization**

I. Introduction

Computation offloading is to shift computation tasks from one platform to another. In the paradigm of cloud computing, resource-constrained battery-powered mobile devices may offload computation-intensive tasks to a cloud data center to speedup the computation, save battery energy, and circumvent limitations on computation capability and resource capacity of the devices [1]. However, cloud service provider may charge mobile devices for the cloud resource usage. Offloading itself may also incur communication overheads such as extra delay and energy consumption, which could be the primary concerns when offloading traffic goes through wireless channels.

Edge computing, or more specifically, multi-access edge computing (MEC), provides cloud service at the edge of wireless networks [2]. Compared with cloud computing, MEC significantly reduces the communication latencies between servers and end devices [3]. However, MEC server generally has less resource than cloud data center so a single MEC system may not have adequate capacity to serve all service requests. This calls for offloading among MEC servers (i.e., horizontal offloading) and between MEC servers and cloud data centers (i.e., vertical offloading).

In vertical offloading, MEC servers offload tasks to cloud only when necessary since task execution in cloud generally causes higher end-to-end latency. It is an optimization problem how to select offloading tasks to minimize overall cost while meeting latency constraint with limited MEC resource capacities.

Recent progress in electric automobile and information communication technology (ICT) enables Internet of Vehicles (IoV) [4]. In IoV, vehicles serve not only for transportation but also as a part of communication and computation infrastructure. Vehicular-fog computing (VFC) [5] is an emerging technology which further turns vehicles into fog nodes that act as small-scale cloud platforms for vehicles themselves and other connected devices as well. VFC is suitable for applications such as autonomous driving and navigation. However, when a vehicle is parked, the platform goes off which can be otherwise exploited by other cloud platforms. In this paper, a vehicular fog (VF) consists of a road side unit (RSU) and all vehicles associated with it (as VF nodes). RSU acting as VF manager aggregates and dispatches the resources of idle vehicles.

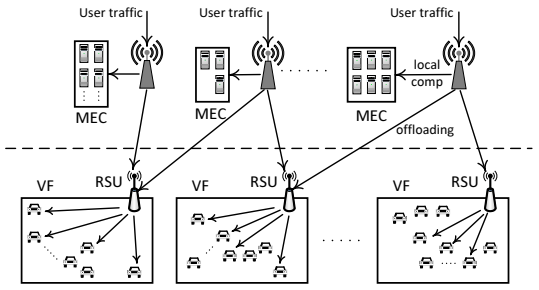


Fig. 1: Two-tier federated EVF architecture

We consider the federation of MECs with VFs to form a two-tier edge and vehicular-fog (EVF) architecture (Fig. 1). Most studies on computation offloading in an MEC-vehicle

coexisting scenario considered offloading computing tasks from vehicles to MEC servers [6], [7] for energy saving or better computation power. In fact, EVF also enables vertical offloading from MEC systems to VFs. Vehicles have been used for offloading to minimize system response time in traffic management system [8]. The main idea in our study is to aggregate otherwise-idle resources on vehicles to share the workload of dedicated edge servers, for which the potential benefit is twofold. On one hand, the offloading alleviates resource demand in MEC and thus reduces operating expense (OPEX) of MEC systems. On the other hand, the offloading offers vehicle owners extra payoffs by a better utilization of their idle resources. The key point is to leverage the heterogeneity of cost and capacity between edge systems and VFs for an MEC-to-VF computation offloading to be cost-effective. However, this potential has been little studied. To the best knowledge of the authors, the preliminary version of this paper [9] was the only work on offloading from edge to VFs. However, that work considered only one edge system. In this paper, we extend [9] to consider offloading from multiple edges to VFs.

When an MEC system takes user requests from outside, it has to decide an *offloading configuration*, the dispatch of user requests to execution platforms including its own MEC servers and selected VFs. The objective is to minimize overall cost subject to the latency constraint associated with user requests and also the processing capacities of the MEC and VFs. As VFs are heterogeneous in terms of cost and capacity, an optimal offloading configuration involves traffic offloaded to multiple VFs, each with a distinct amount of workload. With queuing model for the capacity and workload calculation, we formulate the optimization as a mixed integer programming problem. However, we do not intend to pursue an optimal solution to the problem because the problem is NP-hard and a globally-optimal solution is hardly attainable as independent MEC service providers may not have the incentive to conform to the optimal result. Instead, we propose a decentralized offloading configuration protocol (DOCP) for individual MEC system to collect capacity and cost information from each VF, based on which the system then determines its own offloading configuration.

DOCP works well in a single-MEC EVF environment. When multiple MEC systems execute DOCP concurrently to minimize their respect costs, however, their optimal configurations may be in conflict with one another. This calls for a conflict-resolution mechanism. We model a conflict-free set of configurations as a many-to-many matching between MEC systems and VFs, and propose a decentralized conflict-resolution protocol on top of DOCP which is patterned after the deferred acceptance (DA) matching algorithm [10], [11]. Our approach effectively decomposes the goal of overall cost minimization into local payoff-pursuing subgoals. To this end, we designate a payoff-related preference function for each MEC system and each VF that specifies its preference over potential matching partners. As every participant pursues its own goal, the overall cost can be generally reduced.

We conducted extensive simulations to study the performance of the proposed approach. The results confirm that the proposed approach can effectively reduce overall cost by leveraging the heterogeneity of cost and capacity between MEC systems and VFs. When compared with other alternatives, the proposed matching outperforms greedy approaches that prefer offloading to a VF that has either the most number of vehicles or the lowest vehicle cost. The performance of the proposed matching protocol is also comparable to that of Particle Swarm Optimization algorithm.

In short, the main contributions of this work are summarized as follows.

- The problem of minimizing overall cost of computation offloading from multiple MEC systems to multiple VFs in an EVF system has been formulated.
- A decentralized offloading configuration protocol named DOCP has been devised for low-cost offloading configuration in individual MEC system.
- A matching protocol on top of DOCP has been proposed for multiple MEC systems to resolve conflicts among their offloading configurations.
- The conflict-resolution matching protocol serves as a group negotiation mechanism for offloading requesters and providers such that no requester can be better off by dropping its matching outcome (a property named *individual rationality*).

The remainder of this paper is organized as follows. The next section reviews related work and presents the computation and communication models of EVF system. Sec. III formulates the cost minimization problem. In Sec. IV, we present the details of DOCP for individual MEC system to perform offloading configuration. The matching protocol for multiple MEC systems to contend VF resources is detailed in Sec. V. Sec. VI presents performance evaluation of the proposed approach and comparisons with other alternatives. Finally, Sec. VII concludes this paper.

II. Background

A. Related Work

Computation offloading is a user-centric use case in cloud computing. Some early studies have been devoted to user's decisions on computation offloading [12], [13]. For mobile devices in MEC, user's decision variables may also include computation speed [14], [15], channel or bandwidth access [16], [17], transmission power [12] and the ratio of offloading traffic [14]. Other issues of computation offloading are the allocation of computation resource [18] and the scheduling of user's computation [19]. Recently, researchers have attempted joint optimizations of offloading decisions and resource allocations [20], [21].

Like mobile devices, fog nodes also have limited computation capability and resource. Some researchers proposed offloading computation tasks from fog nodes to central cloud to reduce task execution time and energy consumption of fog nodes [22]. Huang *et al.* [23] presented a dynamic offloading algorithm based on Lyapunov optimization to save energy for mobile devices while meeting the execution time constraint of mobile applications. In [24], Liu *et al.*, modeled a multi-objective optimization problem to minimize energy consumption, execution delay, and payment cost. They proposed an

approach that finds the optimal offloading probability and transmit power for each mobile device.

The idea of utilizing storage resource of parking cars is not new [25]. Hou *et al.* [5] firstly proposed utilizing parked and moving vehicles as fog nodes. Wang *et al.* [8] considered offloading the workload of a real-time traffic management in an Internet of Vehicle (IoV) system. To this end, a city is partitioned into several regions. In each region, a cloudlet is set up, and nearby parked and moving vehicles are used as potentially fog nodes which help the offloading. In [26], time-sensitive and computation-intensive application running on a mobile device is offloaded to nearby smart vehicles. Before a vehicle executing the offloading becomes unavailable due to mobility, the vehicle needs to hand over the job to another vehicle or return it to the mobile device. The problem is to find out the right time and the right target for the handover to minimize energy consumption while meeting the delay constraint of the application.

Zhou *et al.* [6] considered a vehicular workload offloading scenario, where multiple vehicles are under the coverage of a single RSU. The computation workload of each vehicle is either processed locally or offloaded to the edge server co-located with the RSU. The offloading decisions usually give vehicle users higher quality of experience but also incur payments. For the edge computing service providers, offloading brings in revenue but also electricity cost. Zhang *et al.* [27] also considered offloading from multiple vehicles to multiple Vehicular Edge Computing (VEC) servers that are co-located with RSUs. They modeled it as a Stackelberg game and proposed a distributed algorithm to maximize the utilities of both the vehicles and the VEC servers

Lin *et al.* [9] proposed the first work on offloading from edge to VFs. They assumed the same system model as this paper but considered only one edge system with multiple VFs. In this paper, we extend [9] to consider offloading from multiple edges to multiple VFs.

Table I summarizes recent offloading studies in vehicular environment. These studies differ in the direction of offloading and the number of entities involved.

B. System Model

There are two types of VFs in VFC. In a *static VF*, vehicles are parked for hours or even days in a large parking lot (like those located on airports or train stations). Though vehicles may still dynamically join or leave a VF, the frequency is not high so the capacity of a VF does not change dramatically. Static VF can thus be exploited to host computation-intensive tasks such as Blockchain-based applications. In a *dynamic VF*, vehicles temporarily participate in a VF when they stop by some locations (e.g., highway rest areas). Because of dynamic-changing capacity, a dynamic VF may only host lightweight tasks that are easy to instantiate and terminate. The number of dynamic VFs can be high enough in some areas such that, if these VFs are exploited carefully and wisely, we can attain considerable resources from dynamic VFs.

We consider offloading configuration from multiple MEC systems to multiple VFs as shown in Fig. 2. In this two-tier EVF hierarchy, computation workload fed by outside users into an MEC system can be partially severed by the servers of the MEC system and partially offloaded to VFs. Meanwhile, a VF can serve offloading requests from multiple MEC systems.

We assume a set of MEC systems $E = \{e_i\}_{i=1}^{|E|}$. User requests coming into MEC system e_i form a Poisson process $\mathcal{R}_i$ with mean arrival rate λ_i^{in}. The workload comes with latency constraint $L_i^{\max}$. For each MEC system, there is a traffic splitter which splits user's traffic among the MEC and VF systems based on a configured workload splitting ratio.

Each MEC system $e_i \in E$ has $\hat{m}_i$ homogeneous MEC servers. Among them, $m_i \leq \hat{m}_i$ servers will be allocated to process user's workload. A single MEC server in e_i is modeled as an M/M/1 queueing system with service rate μ_i. Consequently, an allocation of c MEC servers as a whole to serve workload in a first-come, first-serve manner renders an M/M/c queue (as in [6]).

We assume a set of VFs $F = \{f_j\}_{j=1}^{|F|}$, each corresponding to an on-street or off-street parking lot. The dynamics of vehicles entering a parking lot have been modeled as an M/M/C [28], [29] or M/M/∞ queue [30]. We assume that vehicles join VF f_j following a Poisson process with rate a_j. The dwell times of vehicles in f_j are assumed i.i.d. exponential random variables with mean $1/d_j$. We take the M/M/∞ model so the number of vehicles in f_j is a Poisson distribution with mean a_j/d_j. To utilize a vehicle residing in a VF, we should also consider its residual energy and power consumption rate. Let $p_{j,k}^{\text{res}}$ and $p_{j,k}^{\text{rate}}$ be the amounts of residual energy and power consumption rate (both in percentage), respectively, of vehicle k in VF f_j. If the vehicle is exploitable only when the residual energy is not less than $p_{j,k}^{\text{th}}$ percentage, then the *maximal usage time* of the vehicle is

$$t_{j,k} = (p_{j,k}^{\text{res}} - p_{j,k}^{\text{th}})/p_{j,k}^{\text{rate}}. \tag{1}$$

Note that we assume a non-linear energy consumption model. Whether a vehicle k in VF f_j with maximal usage time $t_{j,k}$ should be used to share workload offloaded from MEC system e_i is application dependent. Some application only needs to process short stateless transactions, for which small $t_{j,k}$ should be enough. However, some other application may demand large $t_{j,k}$ to avoid overhead associated with stateful workload migration. Let D_i be the minimal service time demanded by the workload from e_i. We use V_j to denote the set of vehicles in VF f_j that are almost surely available with $t_{j,k} \geq D_i$. Only vehicles in V_j could be considered for processing offloaded requests to maintain a certain level of service quality.

When multiple MEC systems offload requests to a common VF, all requests from the same MEC are collectively processed by an exclusively allocated set of vehicles. Let $v_{i,j}^k$ be an variable indicating where a vehicle $k \in V_j$ is allocated to serve workload offloaded from e_i. Define $V_j^i = \{k \in V_j \mid v_{i,j}^k = 1\}$. We have $\bigcup_i V_j^i \subseteq V_j$ and $V_j^i \cap V_j^{i'} = \varnothing$ for any two MEC systems e_i and $e_{i'}$.

Similar to MEC system, each vehicle in a VF is also modeled as an M/M/1 queue with uniform service rate μ_{v}. Therefore, a bundle of c vehicles in the same VF allocated to process requests offloaded from the same MEC system can be modeled as an M/M/c queue.

TABLE I: Recent Offloading Studies in Vehicular Environment

	Offloading From	Offloading To	Our Classification
Ref. [8]	Cloud	Cloudlets & VFs	Cloud to Edge/VF
Ref. [26]	Mobile Device	Multiple Vehicles	Mobile to Vehicle
Ref. [27]	Vehicles	VEC Servers	Vehicle to Edge
Ref. [6]	Vehicles	Single Edge (RSU)	Vehicle to Edge
Ref. [7]	Vehicles	Multiple Edges	Vehicle to Edge
Ref. [9]	Single Edge	Multiple VFs	Edge to VF
This work	Multiple Edges	Multiple VFs	Edge to VF

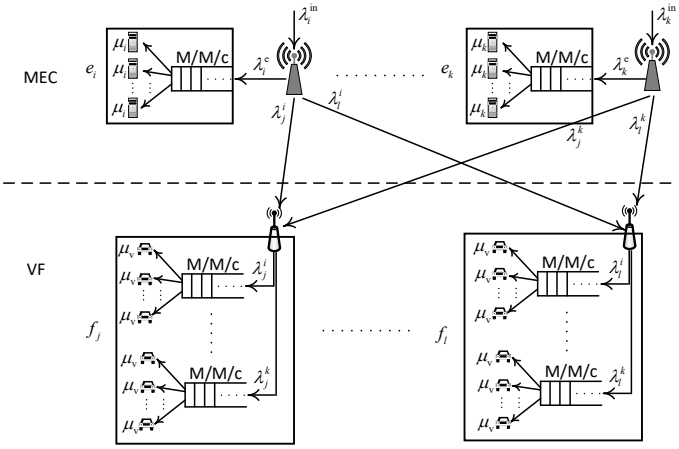


Fig. 2: Offloading in Multi-MEC EVF System

Different offloading configurations incur different costs. We assume that all MEC servers in the same MEC system e_i have identical computation cost c_i. So the total computation cost of e_i when it activates m_i out of $\hat{m}_i$ MEC servers is $m_i c_i$. On the other hand, we assume that vehicles have heterogeneous allocation costs. We use c_j^k to denote the cost of allocating vehicle $k \in V_j$.

Let λ_j^i be the rate of the requests offloaded from e_i to f_j. The mean rate of requests locally processed by e_i is

$$\lambda_i^{\mathrm{e}} = \lambda_i^{\mathrm{in}} - \sum_{j=1}^{|F|} \lambda_j^i. \tag{2}$$

In steady state, λ_i^{e} should be less than $m_i \mu_i$, where m_i is the number of allocated MEC servers in e_i. When the condition is met, the computation latency of requests served by e_i is a function of m_i defined as

$$l_i^{\mathrm{e}}(m_i) = \frac{\mathrm{C}(m_i, \lambda_i^{\mathrm{e}}/\mu_i)}{m_i \mu_i - \lambda_i^{\mathrm{e}}} + \frac{1}{\mu_i}, \tag{3}$$

where $\mathrm{C}(c, \lambda/\mu)$ is Erlang's C formula [31] defined as

$$\mathrm{C}(c, \lambda/\mu) = \frac{1}{1 + (1-\rho)\left(\frac{c!}{(c\rho)^c}\right)\sum_{k=0}^{c-1}\frac{(c\rho)^k}{k!}}, \tag{4}$$

where λ is the arrival traffic rate, μ is the service rate and $\rho = \lambda/(c\mu)$.

Let $n_j^i = |V_j^i|$ be the number of vehicles in V_j^i and $l_{j,i}^{\mathrm{f}}(n_j^i)$ be the computation latency associated with these n_j^i vehicles. Similar to the case of MEC, we have

$$l_{j,i}^{\mathrm{f}}(n_j^i) = \frac{\mathrm{C}(n_j^i, \lambda_j^i/\mu_{\mathrm{v}})}{n_j^i \cdot \mu_{\mathrm{v}} - \lambda_j^i} + \frac{1}{\mu_{\mathrm{v}}}. \tag{5}$$

We assume that every MEC system has a dedicated channel to each VF system. The communication delay in each channel consists of queuing delay, transmission time, and propagation delay. The queuing delay plus transmission time is modeled as an M/M/1 queue with mean service rate μ_{ef}. The propagation delay is equal to $d_{i,j}$, the distance between e_i and f_j, divided by signal speed s. Formally,

$$l_{i,j}^{\mathrm{ef}} = \frac{1}{\mu_{\mathrm{ef}} - \lambda_j^i} + \frac{d_{i,j}}{s}, \tag{6}$$

with $\lambda_j^i < \mu_{\mathrm{ef}}$. This implies that the offloaded traffic cannot exceed channel capacity.

After processing requests from e_i, f_j needs to send results back to e_i. The return traffic is supposed to be a portion of the request traffic. Let ϵ_i be the ratio of the return traffic rate (from f_j to e_i) to λ_j^i. Let μ_{fe} be mean service rate of the channel from a VF to an MEC. The communication delay of the return traffic, $l_{j,i}^{\mathrm{fe}}$, can be estimated by

$$l_{j,i}^{\mathrm{fe}} = \frac{1}{\mu_{\mathrm{fe}} - \lambda_j^i \cdot \epsilon_i} + \frac{d_{i,j}}{s}, \tag{7}$$

with $\lambda_j^i \cdot \epsilon_i < \mu_{\mathrm{fe}}$.

Table II summarizes key notations that will be used in our problem formulation.

III. Problem Formulation

Offloading configuration is to decide the number of allocated MEC servers in each MEC system, the set of allocated vehicles in each VF for each MEC system, and the offloading ratio from the MEC system to each VF to minimize overall cost while meeting the end-to-end latency constraint of each

TABLE II: Summary of Key Notations

(a) System Parameters/User Requirements (Input)

E	Set of MEC systems. $E = \{e_1, e_2, \dots\}$
$\hat{m}_i$	Total number of MEC servers in e_i
μ_i	Mean service rate of a single MEC server in e_i
c_i	Cost of allocating an MEC server in e_i
F	Set of VFs. $F = \{f_1, f_2, \dots\}$
V_j	Set of allocable vehicles in f_j
μ_{v}	Mean service rate of a vehicle
c_j^k	Cost of allocating vehicle k in V_j
$t_{j,k}$	Maximal usage time of vehicle k in V_j
$l_{i,j}^{\text{ef}}$	Communication latency from e_i to f_j
$l_{j,i}^{\text{fe}}$	Communication latency from f_j to e_i
D_i	Minimal service time demanded by e_i
$L_i^{\max}$	Latency constraint of user request at e_i
λ_i^{in}	User request arrival rate at e_i

(b) Decision Variables (Output)

m_i	Total number of servers in e_i serving λ_i^{e}
λ_j^i	Traffic offloaded from e_i to f_j
$v_{i,j}^k$	Variable indicating whether $k \in V_j^i$.

(c) Auxiliary Variables/Functions

V_j^i	Set of vehicles in V_j allocated for λ_j^i
$l_i^{\text{e}}(m)$	Computation latency of m servers in e_i processing $\lambda_i^{\text{e}} = \lambda_i^{\text{in}} - \sum_j \lambda_j^i$
$l_{j,i}^{\text{f}}(n)$	Computation latency of n vehicles in f_j processing λ_j^i

user request. Formally, the objective of offloading configuration is

$$\min_{\{m_i\}_i, \{v_{i,j}^k\}_{i,j}^k, \{\lambda_j^i\}_j^i} \sum_{i=1}^{|E|} \left(m_i c_i + \sum_{j=1}^{|F|} \sum_{k \in V_j} v_{i,j}^k c_j^k \right). \quad (8)$$

The objective is subject to the following constraints.

$$\lambda_i^{\text{in}} - \sum_{j=1}^{|F|} \lambda_j^i < m_i \cdot \mu_i, \quad \forall i \quad (9)$$

$$\lambda_j^i < |V_j^i| \mu_{\text{v}}, \quad \forall i, j \quad (10)$$

$$0 \le m_i \le \hat{m}_i, \quad \forall i \quad (11)$$

$$0 \le \sum_{i=1}^{|E|} \sum_{k \in V_j} v_{i,j}^k \le |V_j|, \quad \forall j \quad (12)$$

$$l_i^{\text{e}}(m_i) \le L_i^{\max}, \quad \forall i, m_i > 0 \quad (13)$$

$$l_{i,j}^{\text{ef}} + l_{j,i}^{\text{f}}(|V_j^i|) + l_{j,i}^{\text{fe}} \le L_i^{\max}, \quad \forall i, j, \lambda_j^i > 0 \quad (14)$$

$$v_{i,j}^k = 1 \rightarrow t_{j,k} \ge D_i, \quad \forall i, j, k \quad (15)$$

$$v_{i,j}^k \in \{0, 1\}, \quad \forall i, j, k \quad (16)$$

Eqs. (9) and (10) ensure that the requests toward e_i that are locally served by e_i and offloaded to f_j should have arrival rates lower than the allocated service rates of e_i and f_j, respectively. Eq. (11) implies that the number of allocated servers cannot exceed the number of available servers in e_i. Eq. (12) ensures that the total number of vehicles in f_j that are allocated to serve offloaded requests does not exceed the number of allocable vehicles. Eq. (13) indicates that the latency of requests toward e_i when being locally served by m_i MEC servers, $l_i^{\text{e}}(m_i)$, should not be higher than the associated latency constraint $L_i^{\max}$. For same requests offloaded to and severed by VF f_j, the total latency, including communication and computation delays, should not be higher than $L_i^{\max}$ as well, as indicated by (14). Eq. (15) demands that only vehicles with usage time not less than the minimal service duration could be allocated. Eq. (16) limits $v_{i,j}^k$ to be an indicator variable.

The cost minimization problem defined above is a mixed integer programming problem, which can be proven NP-hard [32]. Therefore, pursuing an optimal solution to the problem is computationally expensive. Furthermore, MEC systems are not under the control of a single personnel or authority. In fact, we assume independent MEC service providers who make decisions merely for their own business interests. Therefore, a globally-optimal solution is hardly attainable as MEC service providers may have no incentive to conform to the optimal result. We thus propose a decentralized approach for independent MEC systems.

IV. Decentralized Offloading Configuration Protocol (DOCP)

This section presents the decentralized offloading configuration protocol (DOCP) for each MEC system e_i to determine its offloading configuration and request vehicles from corresponding VF managers. For each individual MEC system e_i, its local goal is

$$\min_{m_i, \{\lambda_j^i\}_j, \{v_{i,j}^k\}_j^k} c_{\text{total}}^i = m_i \cdot c_i + \sum_{j=1}^{|F|} \sum_{k \in V_j} v_{i,j}^k c_j^k \quad (17)$$

subject to the same set of constraints (9) to (16). DOCP also serves as a basis for the conflict-resolution protocol presented in the next section.

A. The Protocol

The general behavior of DOCP is as follows. Before an MEC system determines its offloading configuration, it inquires about the workload processing capacity and the associated cost of each federated VF. After an MEC system determines its offloading configuration, it requests a certain number of vehicles from corresponding VF manager. If the manager grants the request, the manager dispatches subsequent traffic from the MEC to the set of vehicles allocated for the MEC.

When workload λ_i^{in} with latency constraint $L_i^{\max}$ and minimal service time D_i arrives at MEC e_i, the proposed EVF greedy approach presented in Algorithm 1 attempts to allocate

Algorithm 1 *EVF_alloc*$_i(F, \lambda_i^{\text{in}}, L_i^{\max}, D_i)$

Require: Set of VFs F; Workload λ_i^{in}; Maximum latency $L^{\max}$; Minimal service time D_i

$S \leftarrow \varnothing$; $U \leftarrow \{0\} \cup \{j | f_j \in F\}$
$\Lambda \leftarrow \lambda_i^{\text{in}}$; $c_{\text{total}}^i \leftarrow 0$; $\mathcal{V} \leftarrow \varnothing$
repeat
  if $0 \in U$ **then**
    $(\lambda_0, c_0) \leftarrow$ *edge_alloc*$_i(\Lambda, L_i^{\max})$
  end if
  for all $1 \leq j \leq |F|, j \in U$ **do**
    $(\lambda_j, V_j^i, c_j) \leftarrow$ *vfog_alloc*$_j(\Lambda, L_i^{\max}, D_i)$
  end for
  $u \leftarrow \arg\max_{j \in U}\{\lambda_j / c_j\}$
  if $\lambda_u \leq \Lambda$ **then**
    if $u = 0$ **then**
      $S \leftarrow S \cup \{e_i\}$
    else
      $S \leftarrow S \cup \{f_u\}$
      $\mathcal{V} \leftarrow \mathcal{V} \cup V_u^i$
    end if
    $c_{\text{total}}^i \leftarrow c_{\text{total}}^i + c_u$
    $\Lambda \leftarrow \Lambda - \lambda_u$
  end if
  $U \leftarrow U \setminus \{u\}$
until $\Lambda = 0$ or $U = \varnothing$
$(\lambda_i^{\text{e}}, c_i^{\text{e}}) \leftarrow$ *edge_alloc*$_i(\lambda_i, L_i^{\max})$
if $\lambda_i^{\text{e}} \geq \lambda_i^{\text{in}}$ and $c_i^{\text{e}} \leq c_{\text{total}}^i$ **then**
  return $(\{e_i\}, \varnothing, c_i^{\text{e}})$
else
  return $(S, \mathcal{V}, c_{\text{total}}^i)$
end if

λ_i^{in} to MEC e_i and/or several VFs to minimize overall cost. The algorithm first resets the solution set S and initializes candidate set U to contain MEC e_i and all VFs. The processing load to be allocated, Λ, is set to λ_i^{in} initially. For each element $j \in U$ that has not been included to S, the algorithm calculates its service capacity λ_j and associated cost c_j. This is done by calling either *edge_alloc*$_i(\Lambda, L_i^{\max})$ or *vfog_alloc*$_j(\Lambda, L_i^{\max}, D_i)$, depending on whether j is e_i or a VF. Among all candidates in U, the algorithm then selects $u \in U$ that has the highest service capacity per unit cost, includes u into S, and deducts its capacity λ_u from Λ. The selection process repeats if neither Λ nor U is empty. After the selection process completes, the algorithm then compares the result with that of *zero offloading*, i.e., $\lambda_j^i = 0$ for all j. Zero offloading will be chosen if it is feasible and has lower total cost.

Let $\lambda_i^{\text{e}} = \lambda_i^{\text{in}} - \sum_j \lambda_j^i$ be the rate of requests locally processed by e_i. Zero offloading is feasible if $l_i^{\text{e}}(m_i) \leq L_i^{max}$ for some $m_i \leq \hat{m}_i$ with $\lambda_i^{\text{e}} = \lambda_i^{\text{in}}$. As the MEC is modeled as an M/M/c queue, we can easily calculate $l_i^{\text{e}}(m_i)$ given λ_i^{e} and m_i. Fig. 3 shows the relationship between the latency (mean waiting time) and the number of servers in M/M/c. Given workload λ and latency constraint $L^{\max}$, Algorithm 2 attempts allocating a minimal number of servers in e_i for zero offloading, and calculates the associated cost. If zero offloading is not feasible, Algorithm 2 figures out the maximal workload (request arrival rate) that e_i can process while still meeting the latency constraint. This can be done using the Bisection method [33] as shown in Algorithm 3.

Algorithm 4 details the allocation of vehicles in a VF f_j

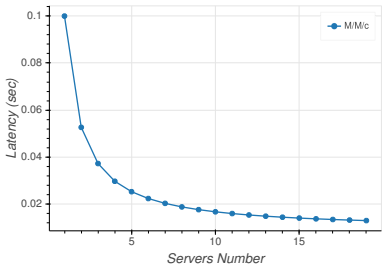


Fig. 3: Latency vs. the number of allocated servers

Algorithm 2 *edge_alloc*$_i(\lambda, L^{\max})$

Require: Number of MEC servers: $\hat{m}_i$; Workload λ; Latency constraint $L^{\max}$

$m_i \leftarrow 0$; $L' \leftarrow \infty$
while $m_i < \hat{m}_i$ and $L' > L^{\max}$ **do**
  $m_i \leftarrow m_i + 1$
  $L' \leftarrow l_i^{\text{e}}(m_i)$
end while
if $L' \leq L^{\max}$ **then**
  $\lambda^{\max} \leftarrow \lambda$
else
  $\lambda^{\max} \leftarrow$ *max_capacity*$_i(\lambda, L^{\max}, m_i)$
end if
return $(\lambda^{\max}, m_i c_i)$

for processing workload λ with latency constraint $L^{\max}$ and service duration D. The vehicle allocation is more complicated than edge server allocation in e_i due to heterogeneous vehicle cost and usage time in VF. Rather than just determining the number of allocated vehicles, the algorithm needs to determine whether to allocate each individual vehicle $k \in V_j$ with allocation cost c_j^k and usage time $t_{j,k}$ so as to meet the associated latency constraint $L^{\max}$ yet minimize the vehicle allocation cost. We construct V_j^i by a heuristic that considers both costs and usage times of vehicles. Initially, V_j^i is set to empty. All vehicles in V_j with usage time not less than D are qualified for allocation. The algorithm allocates qualified vehicles one by one in a non-decreasing order of either their usage-time-to-cost (U/C) ratios or simply their usage times. The allocation stops when the latency associated with V_j^i is already less than $L^{\max}$ or when no more vehicle can be allocated. In the latter case, which means vehicles in V_j^i cannot process λ to meet the latency constraint, we calculate the

Algorithm 3 *max_capacity*$_i(\lambda, L^{\max}, n)$

Require: Workload λ; Latency constraint $L^{\max}$; Number of servers/vehicles n;

$\lambda_L \leftarrow 0$; $\lambda_U \leftarrow \lambda$
repeat
  $\lambda_M \leftarrow (\lambda_L + \lambda_U)/2$
  $L' \leftarrow l_i^{\text{e}}(n)$ ▷ $L' \leftarrow l_{i,j}^{\text{ef}} + l_{j,i}^{\text{f}}(n) + l_{j,i}^{\text{fe}}$ in case of VF
  if $L' \leq L^{\max}$ **then**
    $\lambda_L \leftarrow \lambda_M$
  else
    $\lambda_U \leftarrow \lambda_M$
  end if
until $\lambda_U - \lambda_L \leq 2\gamma\lambda_L$
return (λ_L)

maximal workload $\lambda^{\max}$ that vehicles in V_j^i can collectively process with latency equal to or less than $L^{\max}$. This value can also be found using the bisection method.

Algorithm 4 *vfog_alloc*$_j(\lambda, L^{\max}, D)$

Require: Set of allocable vehicles V_j; Workload λ; Latency constraint $L^{\max}$; Minimal service time D
1: $U \leftarrow \{k \in V_j | t_{j,k} \geq D\}$; $V_j^i \leftarrow \varnothing$; $L' \leftarrow \infty$
2: **while** $U \neq \varnothing$ and $L' > L^{\max}$ **do**
3: $v \leftarrow \max_{k \in U}\{t_{j,k}/c_k\}$ or $v \leftarrow \max_{k \in U}\{t_{j,k}\}$
4: $U \leftarrow U \setminus \{v\}$
5: $V_j^i \leftarrow V_j^i \cup \{v\}$
6: $L' \leftarrow l_{i,j}^{\text{ef}} + l_{j,i}^{\text{f}}(|V_j^i|) + l_{j,i}^{\text{fe}}$
7: **end while**
8: **if** $L' \leq L^{\max}$ **then**
9: $\lambda^{\max} \leftarrow \lambda$
10: **else**
11: $\lambda^{\max} \leftarrow$ *max_capacity*$_j(\lambda, L^{\max}, |V_j^i|)$
12: **end if**
13: return $(\lambda^{\max}, V_j^i, \sum_{k \in V_j^i} c_j^k)$

The time complexity of Algorithm 3 is $O(\log(\lambda))$. Accordingly, the time complexities of Algorithms 2 and 4 are $O(m_i + \log(\lambda_i^{\text{in}}))$ and $O(|V_j| \log(|V_j|) + \log(\lambda_i^{\text{in}}))$, respectively. The main loop in Algorithm 1 comprises at most $|F|$ iterations. Line 5 in the loop will be executed at most once. For Line 8, the time complexity will be $O(|F| V_{\max} \log(V_{\max}) + |F| \log(\lambda_i^{\text{in}}))$ per iteration, where $V_{\max} = \max_j |V_j|$. It turns out that the time complexity of Algorithm 1 is $O(|F|^2 (V_{\max} \log(V_{\max}) + \log(\lambda_i^{\text{in}})) + m_i)$.

B. A Running Example

We present a simple running example of DOCP as follows. We assume an MEC system e_i and two VFs f_1 and f_2 with system parameters shown in Table III.

TABLE III: System Parameters of the Running Example

System	Service rate	Cost	Usage time
e_i ($\hat{m}_i = 1$)	$\mu_i = 200$	$c_i = 200$	∞
f_1 ($n_1 = 2$)	$\mu_{\text{v}} = 5$	$(c_1^1, c_1^2) = (10, 20)$	≥ 2
f_2 ($n_2 = 3$)	$\mu_{\text{v}} = 5$	$(c_2^1, c_2^2, c_2^3) = (5, 20, 50)$	≥ 1

Suppose that $\lambda_i^{\text{in}} = 20$ requests per second (rps) and $L_i^{\max} = 1$ sec. Because $l_i^{\text{e}}(\hat{m}_i) = 1/(\mu_i - \lambda_i^{\text{in}}) = 0.0056$ does not exceed $L_i^{\max}$, zero offloading is feasible with cost $c_i = 200$. For VF f_1, since all vehicles in V_1 have a usage time larger than $L_i^{\max}$, they can be exploited with aggregated service rate $2\mu_{\text{v}} = 10$ rps. With this service rate, the total latency $L' > L_i^{\max}$ if f_1 alone is to process λ_i^{in}. So Algorithm 4 calculates $\lambda^{\max} =$ *max_capacity*$_1(20, 1, 2) = 8.9$ which is the maximal workload f_1 can process without latency exceeding $L_i^{\max}$. The cost associated with this allocation is $\sum_k c_1^k = 30$. Similarly, $\lambda^{\max} = 13.9$ with cost 75 for f_2. Algorithm 1 then chooses among e_i, f_1, and f_2 a system with the highest capacity–to-cost ratio. As shown in Table IVa, f_1 will be chosen firstly to process a portion of λ_i^{in}, leaving a workload $\lambda = 20 - 8.9 = 11.1$ to e_i and f_2. In the second round, f_2 will be chosen due to its higher capacity–to-cost ratio. The allocation of all vehicles in f_2 leaves no more workload to process. The total cost turns out to be 105.

TABLE IV: Algorithm Execution

(a) $\lambda_i^{\text{in}} = 20$

Round		1	2
λ_j/c_j	e_i	$20/200 = 0.10$	$11.1/200 = 0.06$
	f_1	$8.9/30 = 0.30$	N/A
	f_2	$13.9/75 = 0.19$	$11.1/75 = 0.15$
Selection		f_1	f_2
Cost (Subtotal)		\$30	\$105

(b) $\lambda_i^{\text{in}} = 200$

Round		1	2
λ_j/c_j	e_i	$199/200 = 0.96$	N/A
	f_1	$8.9/30 = 0.30$	$6/30 = 0.20$
	f_2	$13.9/75 = 0.19$	$6/25 = 0.24$
Selection		e_i	f_2
Cost (Subtotal)		\$200	\$225

If we increase λ_i^{in} to 205 rps, e_i will has a higher capacity–to-cost ratio than either f_1 or f_2. So e_i with capacity 199 rps (after rounding) will be selected in the first round (Table IVb), leaving workload 6 rps for process. Note that f_1 has a higher capacity–to-cost ratio than f_2 in the first round. However, f_1's superiority over f_2 no longer holds in the second round because both VFs allocate two vehicles for processing the remaining workload but the cost associated with f_1 is higher than that with f_2.

V. Matching Protocol For Multiple MEC Systems

We next present a matching protocol based on request-response paradigm for multi-MEC multi-VF EVF architecture. In a general framework of this protocol, each MEC system independently selects a set of VFs to submit its offloading request. As a VF may receive offloading requests from multiple MEC systems at a time, it tentatively grants a set of requests while rejecting the others without coordination with other VFs. An MEC system with requests rejected may submit requests to other VFs subsequently. On the other hand, a VF may later reject a request that was tentatively granted previously as long as the VF can be better off by doing so. The negotiation process ends when every MEC system has no request remained to submit.

In the context of matching theory, the set of MEC systems E and the set of VFs F are two groups of agents. A matching relation maps an MEC system e_i to a VF e_j and reversely if $\lambda_j^i > 0$. The matching relation is many-to-many [34] because an MEC system can offload its workload to multiple VFs while a VF can serve offloading requests from multiple MEC systems.

In the proposed approach, each $e_i \in E$ decides the set of VFs to submit requests based on a preference relation $\succ_{e_i}$ defined on all possible sets of VFs. For any two sets of VFs $S, T \subseteq F$, relation $S \succ_{e_i} T$ indicates e_i prefers S to T. Likewise, each $f_j \in F$ decides the set of requests to grant based on a preference relation $\succ_{f_j}$ defined on all possible sets of MEC systems. For any two sets of MEC systems

$S, T \subseteq E$, relation $S \succ_{f_j} T$ means f_j prefers S to T. For each $e_i \in E$ and $F' \subseteq F$, we define $C(F', \succ_{e_i})$ to be e_i's most-preferred subset of F' according to e_i's preference relation $\succ_{e_i}$. Similarly, for each $f_j \in F$ and $E' \subseteq E$, we define $C(E', \succ_{f_j})$ to be f_j's most-preferred subset of E' according to f_j's preference relation $\succ_{f_j}$.

A. The Protocol

The matching protocol proceeds in rounds. In each round, each MEC system e_i independently uses Algorithm 1 to obtain an offloading configuration which specifies the possible set of vehicles ($V_j^i \subseteq \mathcal{V}$) to be requested from each VF $f_j \in S$. If e_i itself also processes user's workload (i.e., $e_i \in S$), its capacity λ_i^e is first calculated and deducted from the whole workload Λ. If there is any VF in the offloading configuration, e_i then selects the most-preferred subset of VFs in S (i.e., $C(S \setminus \{e_i\}, \succ_{e_i})$) to submit its resource request. If the request to some f_j is accepted, the amount of workload to be offloaded to f_j, λ_j^i, is calculated and deduced from Λ. Otherwise, if e_i receives a rejection that corresponds to a previously accepted request, the amount of workload corresponding to the request is added back to Λ. The process repeats until no more offloading request is needed or possible. The detailed procedure for each e_i is shown in Algorithm 5.

Algorithm 5 Procedure for each $e_i \in E$

1: $\Lambda \leftarrow \lambda_i^{\text{in}}$
2: $A_i \leftarrow \varnothing$ ▷ Set of VFs that accept e_i's requests
3: $(S, \mathcal{V}, c_{\text{total}}^i) \leftarrow$ *EVF_alloc*$_i(F, \Lambda, L_i^{\max}, D_i)$
4: **if** $e_i \in S$ **then**
5: $(\lambda_i^{\text{e}}, c_i^{\text{e}}) \leftarrow$ *edge_alloc*$_i(\Lambda, L_i^{\max})$
6: $\Lambda \leftarrow \Lambda - \lambda_i^{\text{e}}$
7: **end if**
8: **while** $\mathcal{V} \neq \varnothing$ **do**
9: $P_i \leftarrow C(S \setminus \{e_i\}, \succ_{e_i})$
10: send *req*$(|V_j^i|, mv_{i,j})$ to each $f_j \in P_i$
11: **for all** *decision* received from each f_j **do**
12: **if** *decision* = ACCEPT **then**
13: $A_i \leftarrow A_i \cup \{f_j\}$
14: $\lambda_j^i \leftarrow$ *max_capacity*$_j(\Lambda, L_i^{\max}, |V_j^i|)$
15: $\Lambda \leftarrow \Lambda - \lambda_j^i$
16: **else if** *decision* = REJECT and $f_j \in A_i$ **then**
17: $A_i \leftarrow A_i \setminus \{f_j\}$
18: $\Lambda \leftarrow \Lambda + \lambda_j^i$
19: **end if**
20: **end for**
21: $F \leftarrow F \setminus P_i$
22: $(S, \mathcal{V}, c_{\text{total}}^i) \leftarrow$ *EVF_alloc*$_i(F, \Lambda, L_i^{\max}, D_i)$
23: **end while**

A potential issue of this procedure is that the result may not be *individually rational* for MEC systems. That is, with the outcome, some MEC system can be better off by dropping its matching with some VF. For example, suppose that e_i sends a request to each VF in $C(S \setminus \{e_i\}, \succ_{e_i}) = \{f_1, f_2, f_3\}$ but only f_2 and f_3 accept e_i's request. If $\{f_2, f_3\} \subseteq C(S \setminus \{e_i, f_1\}, \succ_{e_i})$, then e_i may send further requests to all VFs in $C(S \setminus \{e_i, f_1\}, \succ_{e_i})$ for the most-preferred result when f_1 is excluded from consideration. However, if $\{f_2, f_3\} \not\subseteq C(S \setminus \{e_i, f_1\}, \succ_{e_i})$, e_i would have been more satisfied if VF f_2 or f_3 (or both, depending on whether the VF is in $C(S \setminus \{e_i, f_1\}, \succ_{e_i})$) denied its request. This happens because vehicles have heterogeneous allocation costs.

To ensure individual rationality, we define $C(F, \succ_{e_i})$ to be e_i's most-preferred VF in F (a singleton). Formally,

$$C(F, \succ_{e_i}) = \{f_j \in F \mid \{f_j\} \succeq_{e_i} \{f_j'\}, \forall f_{j'} \in F\}, \quad (18)$$

where $\{f_j\} \succeq_{e_i} \{f_j'\}$ if $f_j = f_j'$ or $\{f_j\} \succ_{e_i} \{f_j'\}$.

For MEC system's preference over set of VFs, we define MEC e_i's *preference value* over each VF $f_j \in F$ to be $P_i(f_j) = |V_j^i|$, the number of vehicles to be allocated from f_j. Accordingly, we have the following definition for each $F' \subseteq F$:

$$C(F', \succ_{e_i}) = \{f_j \in F' \mid |V_j^i| \geq |V_{j'}^i|, \forall f_{j'} \in F'\}. \quad (19)$$

We prove that all matching results are individually rational for all MEC systems in the Appendix.

On the other hand, a VF grants all received offloading requests as long as it has adequate resource. If the VF's resource is not enough for all the offloading requests, the VF selectively grants some requests while denying the others. The key to the selection rule here is to minimize overall cost as much as possible. In the proposed approach, VF f_j's selection is based on the reduction of each e_i's total cost when f_j serves e_i. Let $c_{\text{total}}^i(F)$ be the total cost of e_i returned by *EVF_alloc*$_i(F, \lambda_i^{\text{in}}, L_i^{\max}, D_i)$ (cf. Algorithm 1) when e_i and all VFs in F participate in the offloading configuration. The *marginal value* of f_j with respect to e_i is

$$mv_{i,j} = c_{\text{total}}^i(F \setminus \{f_j\}) - c_{\text{total}}^i(F). \quad (20)$$

The notion of marginal value is based on *Social Welfare Maximization* [35]. Note that f_j's marginal value with respect to e_i is calculated and informed by e_i.

VFs receive requests in rounds. Let $\mathcal{E}^{(k)}$ be the set of MEC systems from which requests have been received by VF f_j in round $k \geq 1$. Let $A_j^{(k)}$ be the set of all MEC systems whose requests were accepted in some round $k' < k$ and are not yet rejected by f_j in the beginning of round k. The set of MEC systems whose requests considered by f_j in round k is $A_j^{(k)} \cup \mathcal{E}^{(k)}$. Theoretically, f_j should select a subset of this set E' to grant requests so as to maximize $\sum_{e_i \in E'} m_{i,j}$ (subject to f_j's resource constraint). Formally, given a set of MEC systems E, let $\Omega_j(E)$ be the set of all subsets of E with aggregated demand not exceeding f_j's supply. By (20).

$$C(E, \succ_{f_j}) = \underset{E' \in \Omega_j(E)}{\arg\max} \sum_{e_i \in E'} mv_{i,j} \quad (21)$$

It is not difficult to see that finding $C(E, \succ_{f_j})$ is exactly the 0/1 Knapsack problem, which is NP-complete. We take a greedy approach by granting MEC system's requests one by one in a non-increasing order of their marginal values. After determining $P_j = C(A_j^{(k)} \cup \mathcal{E}^{(k)}, \succ_{f_j})$, any request by MEC system in $P_j \setminus A_j^{(k)}$ should be accepted while those by MEC systems in $\mathcal{E}^{(k)} \setminus P_j$ and $A_j^{(k)} \setminus P_j$ should be rejected. Refer to Algorithm 6 for details.

The main loop of Algorithm 5 will be executed at most $|F|$ times, where Line 9 takes $O(|F|)$ and Line 14 takes

Algorithm 6 Procedure for each $f_j \in F$

1: $A_j \leftarrow \varnothing$ ▷ Set of MECs whose requests are accepted
2: **for all** round $t \in \{1, 2, \cdots\}$ **do**
3: $\mathcal{E} \leftarrow \{e_i \in E \mid req(n_j^i, mv_{i,j})$ is received in round $t\}$
4: **if** $\mathcal{E} \neq \varnothing$ **then**
5: $P_j \leftarrow C(A_j \cup \mathcal{E}, \succ_{f_j})$
6: **if** $P_j \succ_{f_j} A_j$ **then**
7: send ACCEPT to each $e_i \in P_j \setminus A_j$
8: send REJECT to each $e_i \in \mathcal{E} \setminus P_j$
9: send REJECT to each $e_i \in A_j \setminus P_j$
10: $A_j \leftarrow P_j$
11: **else**
12: send REJECT to each $e_i \in \mathcal{E} \setminus A_j$
13: **end if**
14: **end if**
15: **end for**

$O(\log(\lambda_i^{\text{in}}))$ time. Therefore, the time complexity of Algorithm 5 is $O(|F|^3(V_{\max}\log(V_{\max}) + \log(\lambda_i^{\text{in}})) + |F|m_i)$. For Algorithm 6, the maximal number of rounds is $|F|$. In each round, Line 5 takes $O(|E|\log(|E|))$ time. Therefore, the time complexity of Algorithm 6 is $O(|F||E|\log(|E|))$.

B. Running Examples

We use two examples to illustrate the execution of the proposed matching protocol. We assume two MEC systems and three VFs with settings shown in Table V.

TABLE V: Parameters (Case 1)

(a) Settings of MEC Systems

MEC	μ_i	c_i	λ_i^{in}	$L_i^{\max}$
e_1	100	100	10	1
e_2	200	200	20	1

(b) Settings of VFs

VF	n_j	μ_{v}	c_j^k
f_1	2	5	8
f_2	3	5	10
f_3	5	5	9

TABLE VI: Execution of the matching protocol (Case 1)

Round	MEC	$\lvert V_1^i\rvert, \lvert V_2^i\rvert, \lvert V_3^i\rvert$	Request	Result
1st	e_1	$2, 0, 1$	$req(2, 2)$ to f_1	Accept
	e_2	$2, 0, 3$	$req(3, 3)$ to f_3	Accept
2nd	e_1	$0, 0, 1$	$req(1, 1)$ to f_3	Reject
	e_2	$0, 0, 2$	$req(2, 2)$ to f_3	Accept
3rd	e_1	$0, 1, 0$	$req(1, 90)$ to f_2	Accept
	e_2	N/A	N/A	N/A

Table VI shows the execution of the proposed matching protocol. In the first round, e_1 and e_2 prefer and send requests to f_1 and f_3, respectively. Both requests are granted as there is no conflict between them. After the first round, both e_1 and e_2 have remaining workload. In the second round, they both contend for the vehicles in f_3. Because f_3 has only two vehicles left, it cannot grant both requests. Because $mv_{2,3} > mv_{1,3}$, f_3 grants e_2's request and rejects e_1's. In the third round, e_1 is the only requester whose request gets accepted due to adequate resource. Note that $mv_{1,2}$ in the last round is quite high. The reason is that without the resource provided by f_2, e_1 would have to activate its own costly servers to process the remaining workload.

TABLE VII: Parameters (Case 2)

(a) Settings of MEC Systems

MEC	μ_i	c_i	λ_i^{in}	$L_i^{\max}$
e_1	100	100	110	1
e_2	200	200	2	1

(b) Settings of VFs

VF	n_j	μ_{v}	c_j^k
f_1	2	5	8
f_2	3	6	10
f_3	5	8	12

We use the second example to show the impact of VF's on overall cost. We use the same setting as the previous example except λ_i^{in}'s and μ_{v}'s (Table VII). In the first round, e_1 and e_2 both send requests to f_1 which needs to reject one of these requests because f_1 does not have enough vehicles. If f_1's decision is based on marginal values as in the proposed approach, it will accept e_1's request while rejecting e_2's as shown in Table VIII. The total cost is \$32 with three vehicles allocated. If, alternatively, the decision is based on the number of vehicles requested, f_1 will reject e_1's request while accepting e_2's as shown in Table IX. The total cost becomes \$36 with four vehicles allocated. Taking marginal value as preference is better than the alternative in this example.

TABLE VIII: Execution of the matching protocol (Case 2)

Round	MEC	$\lvert V_1^i\rvert, \lvert V_2^i\rvert, \lvert V_3^i\rvert$	Request	Result
1st	e_1	$2, 1, 0$	$req(2, 0)$ to f_1	Reject
	e_2	$1, 0, 0$	$req(1, 2)$ to f_1	Accept
2nd	e_1	$0, 0, 2$	$req(2, 4)$ to f_3	Accept
	e_2	N/A	N/A	N/A

TABLE IX: Execution of the matching protocol with alternative preference (Case 2)

Round	MEC	$\lvert V_1^i\rvert, \lvert V_2^i\rvert, \lvert V_3^i\rvert$	Request	Result
1st	e_1	$2, 1, 0$	$req(2, 0)$ to f_1	Accept
	e_2	$1, 0, 0$	$req(1, 2)$ to f_1	Reject
2nd	e_1	$0, 1, 0$	$req(1, -)$ to f_2	Accept
	e_2	$0, 1, 0$	$req(1, -)$ to f_2	Accept

VI. Numerical Results

We conducted simulations to evaluate the performance of DOCP and the proposed matching protocol. All the simulation programs were written in Python. We assumed a disk-shaped coverage area with radius 100 km, where MEC systems and VFs were randomly distributed. Every result is an average of 50 trials.

A. Performance of DOCP

We first considered an EVF system consisting of an MEC system and 10 VFs. Other parameters are listed in Table Xa.

We varied mean user traffic rate λ_i^{in} to study how it relates to total cost. Fig. 4a shows the results of the proposed approach. We observe that when user traffic rate is low, say $\lambda_i^{\text{in}} < 500$ rps, the proposed approach prefers offloading user traffic to VFs instead of processing it locally by MEC servers. The reason is that when user traffic is far beyond the capacity

TABLE X: Simulation parameters

(a) Single-MEC EVF System

Parameter	Default Value
$\hat{m}_i$	5
n_j	$[10, 100]$
μ_i	200 rps
μ_{v}	5 rps
c_i	\$200
c_j^k	\$[1,50]
μ_{ef}	1250 rps
μ_{fe}	1250 rps
ϵ_i	0.01

(b) Multi-MEC EVF System

Parameter	Default Value
$\hat{m}_i$	$[1, 10]$
n_j	$[20, 100]$
μ_i	$[100, 200]$ rps
μ_{v}	5 rps
c_i	\$$[100, 200]$
c_j^k	\$10
μ_{ef}	1250 rps
μ_{fe}	1250 rps
ϵ_i	0.01

of a single MEC server, offloading the workload entirely to VFs provides a more cost-effective solution. As user traffic rate increases, exploiting MEC servers to take some workload becomes more cost-effective than offloading the entire workload to VFs. The offloading configuration found by DOCP is therefore a mixture of local computation and offloading. When the user traffic rate increases to some point, all MEC servers must be activated to handle user traffic first and then VFs handle all the remaining workload. Consequently, the total cost increases when $\lambda_i^{\text{in}} > 1000$ rps merely due to the contribution of VFs.

We used another setting to observe the scenario more clearly. We took the same configuration but replaced the five servers in e_i with a single big server. The service rate and cost of the big server were both five times of the original server. As shown in Fig. 4b, the result is identical to previous one when $\lambda_i^{\text{in}} < 500$ or $\lambda_i^{\text{in}} > 1000$ rps. When λ_i^{in} is set to some value in between, the total cost comes from a mixed use of the MEC server as VFs. However, the total cost is slightly higher than that of the previous setting. This is because the previous setting provides finer granularity of computation resource than this one.

We next varied the latency constraint $L_i^{\max}$ and measured total cost. Fig. 5a shows the result with $\lambda_i^{\text{in}} = 100$ rps. The amount of resource needed to meet a tight latency constraint is large, so MEC servers are more cost-effective than vehicles. When we loosen the latency constraint to $L_i^{\max} > 0.2$ sec., VFs are instead exploited because vehicles become more cost-effective than MEC servers. Doing so reduces the total cost from \$200 to around \$40.

Fig. 5b shows a result with heavy user traffic ($\lambda_i^{\text{in}} = 1500$ rps). With this workload, MEC servers alone are not capable to process all user traffic while meeting the latency constraint. Additional vehicles are always needed for offloading. Particularly, all vehicles are used when $L_i^{\max} < 0.3$ sec. However, the number of vehicles for offloading dramatically decreases as the latency constraint is loosened. This explains the sharp curve of the total cost.

B. Performance of Matching Protocol

We considered a scenario with five MEC systems and 20 VFs. Besides the parameters listed in Table Xb, we also assumed vehicle arrival rate a_j and departure rate d_j for each VF f_j. The values of a_j's and d_j's were randomly determined within the range $[0, 20]$. The service rate of each vehicle was 5 rps. All MEC servers in the same MEC systems had identical service rate, though service rates in different MEC systems were randomly determined.

We used normal distribution to set user traffic rate λ_i^{in} for each $e_i \in E$. We varied the mean of the distribution from 0 to 1000 rps, with the standard deviation set to one-fourth of the mean. Fig. 6a shows the result of using the proposed matching protocol. Reasonably, the total cost increases as the user traffic increases. When the mean of λ_i^{in} does not exceed 40 rps, all the workload is handled by VFs only. When the mean of λ_i^{in} is greater than 40 rps, MEC servers are introduced to share the workload so as to minimize overall cost. As the workload increases to some point, the MEC servers alone are unable to process all user traffic so unsatisfied workload are offloaded to VFs. Starting from that point, the total cost is in proportional to the cost of VFs (Fig. 6b).

We also used exponential distribution to generate user traffic, but do not observe significant difference.

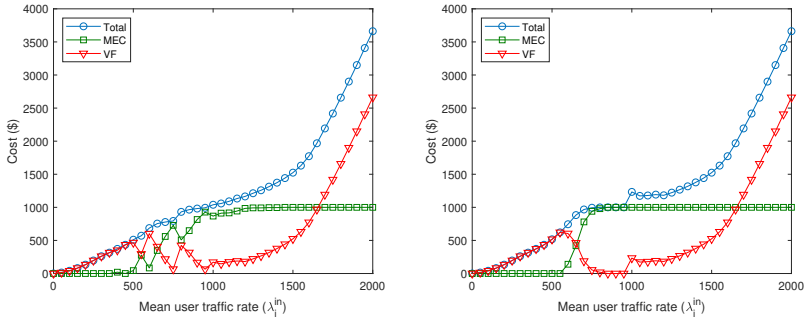
We next studied whether the granularity of MEC service rate affects the total cost. This was done by varying the number of servers in each MEC system and dividing the total service rate of an MEC system equally to each server. Fig. 7 shows how the total cost increases with increasing user traffic rate when each MEC system has 1, 2, or 4 MEC servers. The result indicates that finer granularity of MEC service rate generally leads to lower total cost, which is consistent with our results in single-MEC systems.

We investigated the impact of vehicle cost on the offloading ratio and thus the total cost. The cost of each vehicle was varied from \$0 to \$20. As Fig. 8 shows, when the cost of a vehicle is not higher than \$4, only vehicles are used so the total cost is contributed by VFs only. When the vehicle cost is higher than \$4, MEC servers become more cost-effective than vehicles and thus are used with priority. When all MEC servers are used but VFs are still needed for offloading, the total cost increases simply because the vehicle cost increases.

C. Comparisons With Other Approaches

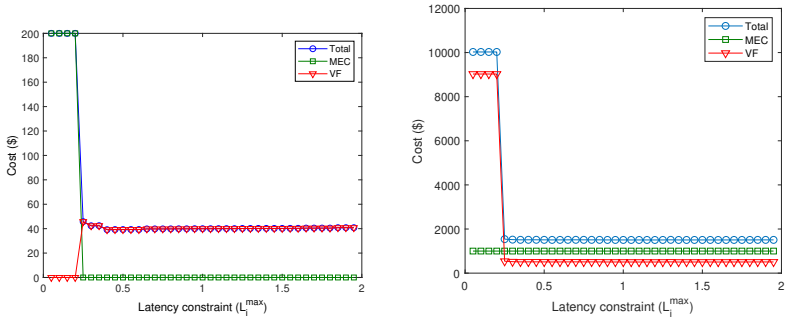
As a comparison, we consider three alternatives to the proposed matching algorithm. The first two, NumFirst and CostFirst, are greedy and centralized heuristics. They first determine the amount of user traffic to offload for each MEC system using Algorithm 1 and then match MEC systems with VFs in a greedy manner. Both approaches pick up an MEC system with the most remaining workload. For the VF matching with this MEC system, NumFirst finds a VF that has the most number of vehicles while CostFirst finds a VF that has the lowest cost. This matching process repeats until no MEC system has remaining workload to be offloaded.

The third alternative is Particle Swarm Optimization (PSO) algorithm. PSO as a meta-heuristic method was first introduced in [36] and has been applied to solve complicated and hard optimization problems in many applications. PSO represents potential solutions to an optimization problem as a swarm of particles or organisms that move like a flock of birds. It iteratively modifies existing solutions by moving the particles around the search-space with the new position of



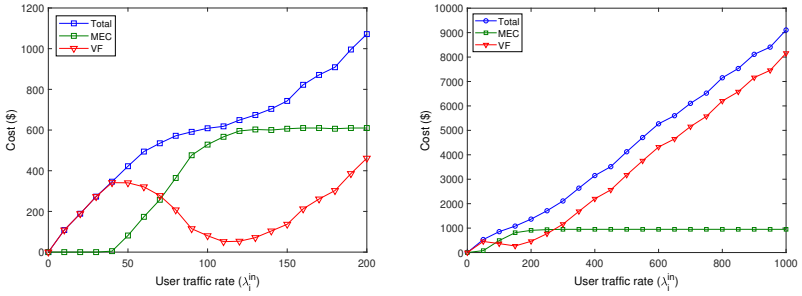
(a) (b)

Fig. 4: Total cost related to mean user traffic rate (a) $\hat{m}_i = 5$, $\mu_i = 200$ rps, $c_i = \$200$ (b) $\hat{m}_i = 1$, $\mu_i = 1000$ rps, $c_i = \$1000$.



(a) (b)

Fig. 5: Cost vs. latency constraint (a) $\lambda_i^{\text{in}} = 100$ rps (b) $\lambda_i^{\text{in}} = 1500$ rps.



(a) (b)

Fig. 6: Total cost vs. mean user traffic rate (a) $\lambda_i^{\text{in}} = 0$ to 200 rps (b) $\lambda_i^{\text{in}} = 0$ to 1000 rps.

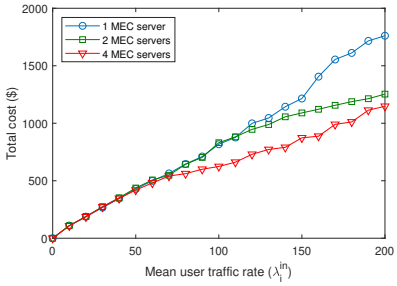


Fig. 7: Total costs with different numbers of MEC servers ($\hat{m}_i$)

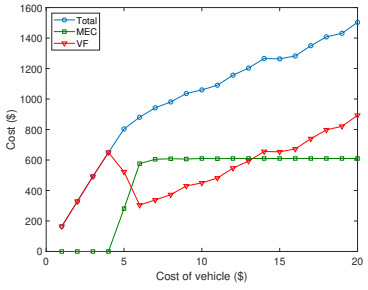


Fig. 8: Total cost affected by the cost of each vehicle

each individual particle determined by the particle's local best-known position, the particle's own moving velocity, and the global best-known position of all the particles. The solution-seeking process essentially mimics flocking behavior of birds. Generally speaking, the quality of solutions produced by PSO

depends on the maximum number of iterations allowed to move particles.

We first fixed vehicle cost and compared the proposed matching protocol with NumFirst and CostFirst. Fig. 9 shows the results. When user traffic rate is low, the proposed approach outperforms both NumFirst and CostFirst. The performance of NumFirst is similar to CostFirst when user traffic rate is low but becomes better than CostFirst with high user traffic rate. CostFirst does not perform well because vehicle costs are homogeneous in this setting.

We next conducted experiments with vehicle costs randomly determined as specified in Table Xb. The results are shown in Fig. 10. The proposed approach again has the lowest costs in all settings of user traffic rates. The superiority of our approach is particularly significant (around 40% cost reduction) with high user traffic rate. Here CostFirst performs better than NumFirst, which can be justified as vehicle costs are heterogeneous in these experiments.

Fig. 11 compares the costs of the proposed matching with those of PSO with two different settings of maximum iterations (*maxiter*). The proposed matching performs much better than PSO with $maxiter = 50$ and slightly worse than PSO with $maxiter = 500$, which is not a surprise. Despite the results, we emphasize that PSO is a centralized off-line approach not suitable to be used as a protocol executing in a dynamic environment. The proposed matching is inherently distributed and can serve as a practical solution in the proposed EVF environment.

VII. Conclusions

We have proposed a two-tier EVF architecture for the realization of computation offloading from MEC systems to VFs. The optimal offloading problem which minimizes overall cost has been modeled as a mixed integer programming problem. As a decentralized approach, we have proposed DOCP for each MEC system to independently decide its own offloading configuration. We also have developed a matching protocol for multiple MEC systems to contend VF nodes simultaneously. The outcomes of the proposed matching protocol are always individually rational for any MEC system.

Simulation results have demonstrated that DOCP and the proposed matching protocol successfully help cost reduction by leveraging the heterogeneity of cost and capacity between MEC systems and VFs. The proposed matching protocol outperforms greedy approaches that prefer offloading to a VF that has either the most number of vehicles or the lowest vehicle cost. The performance of the proposed matching protocol is also comparable to that of PSO-based algorithm.

As a future work, we shall study the impact of vehicle mobility on the efficiency of offloading. We also plan to study the economic model when MEC systems have to bid for the resources they need. An extension to the EVF architecture that includes cloud systems as potential offloading targets also deserves future study.

References

[1] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1628–1656, 2017.

[2] 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.

[3] W. Yu, F. Liang, X. He, W. G. Hatcher, C. Lu, J. Lin, and X. Yang, "A survey on the edge computing for the Internet of Things," *IEEE Access*, vol. 6, pp. 6900–6919, 2018.

[4] O. Kaiwartya, A. H. Abdullah, Y. Cao, A. Altameem, M. Prasad, C.-T. Lin, and X. Liu, "Internet of vehicles: Motivation, layered architecture, network model, challenges, and future aspects," *IEEE Access*, vol. 4, pp. 5356–5373, 2016.

[5] X. Hou, Y. Li, M. Chen, D. Wu, D. Jin, , and S. Chen, "Vehicular fog computing: A viewpoint of vehicles as the infrastructures," *IEEE Trans. Veh. Technol.*, vol. 65, no. 6, pp. 3860–3873, Jun. 2016.

[6] Z. Zhou, H. Yu, C. Xu, Z. Chang, S. Mumtaz, and J. Rodriguez, "BEGIN: Big data enabled energy-efficient vehicular edge computing," *IEEE Commun. Mag.*, vol. 56, no. 12, pp. 82–89, 2018.

[7] X. Xu, Y. Xue, X. Li, L. Qi, and S. Wan, "A computation offloading method for edge computing with vehicle-to-everything," *IEEE Access*, vol. 7, pp. 131 068–131 077, 2019.

[8] X. Wang, Z. Ning, and L. Wang, "Offloading in internet of vehicles: A fog-enabled real-time traffic management system," *IEEE Trans. Ind. Informat.*, vol. 14, no. 10, pp. 4568–4578, Oct. 2018.

[9] Y.-D. Lin, J.-C. Hu, B. Kar, and L.-H. Yen, "Cost minimization with offloading to vehicles in two-tier federated edge and vehicular fog systems," in *Proc. IEEE Veh. Technol. Conf.*, Sep. 2019.

[10] D. Gale and L. S. Shapley, "College admissions and the stability of marriage," *Amer. Math. Monthly*, vol. 69, no. 1, pp. 9–15, 1962.

[11] A. E. Roth, "Deferred acceptance algorithms: History, theory, practice, and open questions," *Int. J. Game Theory*, vol. 36, pp. 537–569, 2008.

[12] W. Zhang, Y. Wen, K. Guan, D. Kilper, H. Luo, and D. O. Wu, "Energy-optimal mobile cloud computing under stochastic wireless channel," *IEEE Trans. Wireless Commun.*, vol. 12, no. 9, pp. 4569–4581, Sep. 2013.

[13] V. Cardellini, V. De Nitto Personé, V. Di Valerio, F. Facchinei, V. Grassi, F. Lo Presti, and V. Picciall, "A game-theoretic approach to computation offloading in mobile cloud computing," *Math. Program.*, vol. 157, pp. 421–449, 2016.

[14] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile-edge computing: Partial computation offloading using dynamic voltage scaling," *IEEE Trans. Commu.*, vol. 64, no. 10, pp. 4268–4282, Oct. 2016.

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

[16] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 4, pp. 974–983, Apr. 2015.

[17] 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.

[18] A. Al-Shuwaili and O. Simeone, "Energy-efficient resource allocation for mobile edge computing-based augmented reality applications," *IEEE Wireless Commun. Lett.*, vol. 6, no. 3, pp. 398–401, Jun. 2017.

[19] L. Yang, J. Cao, H. Cheng, and Y. Ji, "Multi-user computation partitioning for latency sensitive mobile cloud applications," *IEEE Trans. Comput.*, vol. 64, no. 8, pp. 2253–2266, Aug. 2015.

[20] J. Zhang, W. Xia, F. Yan, and L. Shen, "Joint computation offloading and resource allocation optimization in heterogeneous networks with mobile edge computing," *IEEE Access*, vol. 6, pp. 19 324–19 337, 2018.

[21] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 1, pp. 856–868, Jan. 2019.

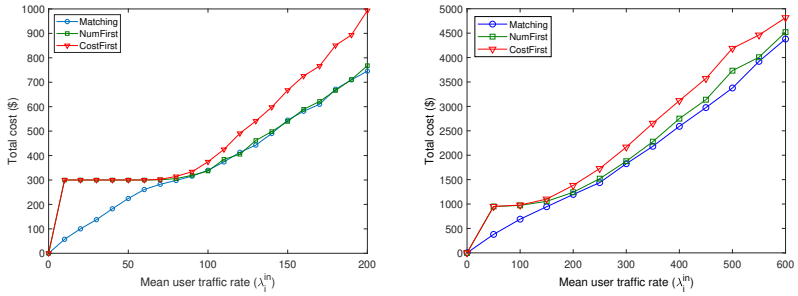
[22] Q. Zhu, B. Si, F. Yang, and Y. Ma, "Task offloading decision in fog computing system," *China Commu.*, vol. 14, pp. 59–68, 2017.

[23] D. Huang, P. Wang, and D. Niyato, "A dynamic offloading algorithm for mobile computing," *IEEE Trans. Wireless Commu.*, vol. 11, no. 6, pp. 1991–1995, Jun. 2012.

[24] L. Liu, Z. Chang, X. Guo, S. Mao, and T. Ristaniemi, "Multi-objective optimization for computation offloading in fog computing," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 283–294, Feb. 2018.

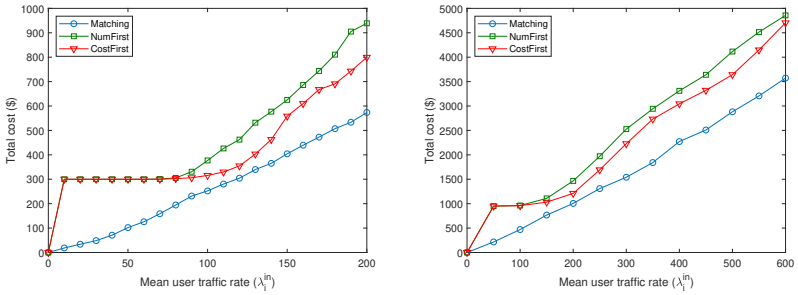
[25] L. Gu, D. Zeng, S. Guo, and B. Ye, "Leverage parking cars in a two-tier data center," *Proc. IEEE Wireless Commun. and Netw. Conf.*, pp. 4665–4670, 2013.

[26] Z. Wang, Z. Zhong, D. Zhao, and M. Ni, "Vehicle-based cloudlet relaying for mobile computation offloading," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 11 181–11 191, Nov. 2018.



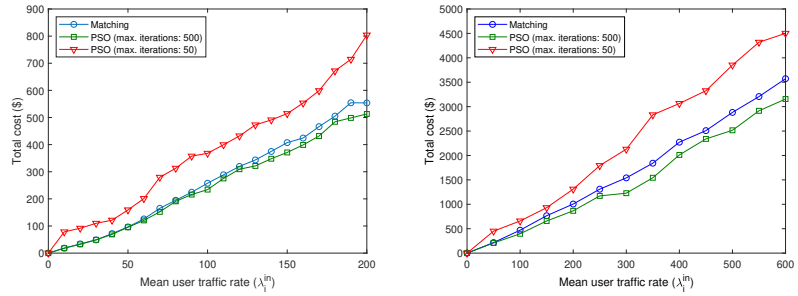
(a) (b)

Fig. 9: Total cost comparison with fixed vehicle cost (a) $\lambda_i^{\text{in}} \in [0, 200]$ (b) $\lambda_i^{\text{in}} \in [0, 600]$.



(a) (b)

Fig. 10: Total cost comparison with dynamic vehicle cost (a) $\lambda_i^{\text{in}} \in [0, 200]$ (b) $\lambda_i^{\text{in}} \in [0, 600]$.



(a) (b)

Fig. 11: Total cost comparison with PSO (a) $\lambda_i^{\text{in}} \in [0, 200]$ (b) $\lambda_i^{\text{in}} \in [0, 600]$.

[27] K. Zhang, Y. Mao, S. Leng, S. Maharjan, and Y. Zhang, "Optimal delay constrained offloading for vehicular edge computing networks," *IEEE Int. Conf. Commun.*, May 2017.

[28] M. Caliskan, A. Barthels, B. Scheuermann, and M. Mauve, "Predicting parking lot occupancy in vehicular ad hoc networks," in *Proc. IEEE VTC2007-Spring*, 2007.

[29] J. Xiao, Y. Lou, and J. Frisby, "How likely am I to find parking? a practical model-based framework for predicting parking availability," *Transportation Research Part B*, vol. 112, pp. 19–39, Apr. 2018.

[30] A. I. Portilla, B. A. Oreña, J. L. Berodia, , and F. J. Díaz, "Using m/m/∞ queueing model in on-street parking maneuvers," *Journal of Transportation Engineering*, vol. 135, no. 8, pp. 527–535, 2009.

[31] L. Kleinrock, *Queueing Systems. Volume 1: Theory*. Wiley, 1975.

[32] R. Kannan and C. L. Monma, "On the computational complexity of integer programming problems," in *Optimization and Operations Research*, ser. Lecture Notes in Economics and Mathematical Systems, R. Henn, B. Korte, and W. Oettli, Eds. Springer, Berlin, 1978, vol. 157, pp. 161–172.

[33] R. L. Burden and J. D. Faires, *Numerical Analysis*, 7th ed. Brooks/Cole, 2000.

[34] M. Sotomayor, "Three remarks on the many-to-many stable matching problem," *Math. Soc. Sci.*, vol. 38, pp. 55–70, 1999.

[35] R. Maheswaran and T. Basar, "Efficient signal proportional allocation (ESPA) mechanisms: Decentralized social welfare maximization for divisible resources," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 5, pp. 1000–1009, May 2006.

[36] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," *Proc. 6th Int. Symp. Micro Machine and Human Science*, pp. 39–43, 1995.

APPENDIX A
INDIVIDUAL RATIONALITY FOR MEC SYSTEMS

To ensure the individual rationality of any matching outcome for an MEC system e_i, e_i's preferences must be *substitutable* defined as follows.

Definition 1: An MEC system e_i's preference relation $\succ_{e_i}$ is *substitutable* if, for any $f_j \in F$ and any two sets of VFs, F' and F'', with $F'' \subseteq F' \subseteq F$, we have

$$f_j \in C(F' \cup \{f_j\}, \succ_{e_i}) \rightarrow f_j \in C(F'' \cup \{f_j\}, \succ_{e_i}). \quad (22)$$

Intuitively, e_i's preference is substitutable if whenever a VF f_j is in e_i's most-preferred set when VF set F' plus f_j is

considered, f_j must also be in e_i's most-preferred set when any smaller set $F'' \subseteq F'$ plus f_j is considered. The following theorem shows that the preference of an MEC system is substitutable.

Theorem 1: The preference of any MEC system is substitutable.

Proof: For every $e_i \in E$, let $f_j \in C(F \cup \{f_j\}, \succ_{e_i})$. By (18), $|V_j^i| \geq |V_{j'}^i|$ for all $f_{j'} \in F \cup \{f_j\}$. Therefore, $|V_j^i| \geq |V_{j'}^i|$ for all $f_{j'} \in F' \cup \{f_j\}$, where $F' \subseteq F$. Consequently, $f_j \in C(F' \cup \{f_j\}, \succ_{e_i})$ for every $F' \subseteq F$. ■

We now justify why all matching results are individually rational for any MEC system. For each $e_i \in E$, let $\mathcal{F}^{(k)}$ denote the set of VFs that are considered by e_i as the potential targets of its offloading requests in round k. We know that $\mathcal{F}^{(1)} = F$ for all $e_i \in E$. For $k \geq 1$, if e_i ever submits a request in round $k+1$, some $f_j \in C(\mathcal{F}^{(k)}, \succ_{e_i})$ must have rejected e_i'request in round k and thus f_j has been removed from e_i's consideration in round $k+1$. Therefore, $\mathcal{F}^{(k+1)} \subset \mathcal{F}^{(k)}$. Now, if some $f_j \in C(\mathcal{F}^{(k)}, \succ_{e_i})$ accepts e_i's request, it must be in $f_j \in C(\mathcal{F}^{(k+1)}, \succ_{e_i})$ because e_i's preference is substitutable. We can prove that this holds for every $k \geq 1$ by mathematical induction. We conclude that e_i definitely cannot be better off by decontracting the matching with any f_j that accepts e_i's request. So all matching results are individually rational for all MEC systems.