Virtual Network Function Service Provisioning for Offloading Tasks in MEC by Trading off Computing and Communication Resource Usages

Yu Ma†, Weifa Liang†, Meitian Huang†, Yang Liu†, and Song Guo¶
† The Australian National University, Canberra, ACT 2601, Australia
¶ The Hong Kong Polytechnic University, Hong Kong

***Abstract*—Mobile edge computing (MEC) has emerged as a promising technology that offers resource-intensive yet delay-sensitive applications from the edge of mobile networks. With the emergence of complicated and resource-hungry mobile applications, offloading user tasks to cloudlets of nearby mobile edge-cloud networks is becoming an important approach to leverage the processing capability of mobile devices, reduce mobile device energy consumptions, and improve experiences of mobile users. In this paper we study the joint VNF instance deployment and offloading task request assignment in MEC, by explicitly exploring a non-trivial tradeoff between usages of different types of resources. We aim to maximize the number of request admissions while minimizing their admission cost. To this end, we first formulate the cost minimization problem that admits all requests, by assuming that there are sufficient computing resources to accommodate the requested VNF instances of all requests, for which we formulate an Integer Linear Program solution and an efficient heuristic. We then deal with the throughput maximization problem by admitting as many requests as possible, subject to computing resource capacity at each cloudlet, for which we devise an efficient algorithm. We finally evaluate the performance of the proposed algorithms through experimental simulations. Experimental results demonstrate that the proposed algorithms are promising.**

I. Introduction

There is a substantial growth in the usage of mobile devices. These devices, including smartphones, sensors, and wearables, are limited by their computational and energy capacities, due to their portable sizes. Leveraging rich computing and storage resources in clouds, mobile devices can extend their capability by offloading user tasks to the clouds for processing. However, clouds are usually far away from most end-users, resulting in long communication delays between end-users and the cloud, and high service cost. Mobile edge computing (MEC), which provides cloud resources at the edge of mobile networks in close proximity to mobile users, is a promising technology to reduce response delays, ensure network operation efficiency, and improve user service satisfaction [9]. Meanwhile, Network Function Virtualization (NFV) has been envisaged as the next-generation networking paradigm. It leverages generic servers/server clusters to implement various network functions as software components instead of purpose-specific hardware middleboxes, which introduces a new dimension of cost savings and network function deployment flexibility [10].

Although MEC has received significant attention, most existing studies focused on minimizing the maximum delay, maximizing the throughput, or the linear combination of multiple performance metrics such as delays, and energy consumptions, without taking into account deploying VNF instances requested by offloading task requests in MEC [1], [4], [8], [11], [12]. For example, Ceselli *et al.* [1] considered VM placement and migration with user request admission by formulating a Mixed Integer Linear Programming solution for it. There are several researches that considered VNF instance provisioning that the VNF instance requested by each request is dedicated to its user only. For example, Ma *et al.* [7] studied VNF instance deployment for users with stringent delay requirements and move frequently within an MEC network. In practice, most users may request the same service from network service providers. Thus, the VNF instances of such services can be shared by multiple users [3], [5]. For example, He *et al.* [3] considered the problem of joint service placement and request scheduling in order to optimally provision edge services while taking into account the demands of both sharable and non-sharable resources in MEC, with an aim to maximize the throughput. However, none of these studies explored the usage tradeoff between computing resource and communication resource in the VNF instance deployment, in order to minimize the admission cost of offloading task requests with specified service requirements which is a fundamental problem of network service provisioning in MEC.

To meet various service demands of offloading tasks from mobile users, network service providers usually instantiate frequently demanded VNF instances of network functions (services) at cloudlets in MEC networks. The deployment of VNF instances at the network edge can not only shorten the latency of user service access but also reduce their cost on the services. However, provisioning network services with different types of VNFs in an MEC network poses many challenges. For example, how many VNF instances need to be instantiated to meet service demands of offloading task requests? how to optimally place the services in cloudlets and assign which requests to the services in order to minimize their admission cost? and how to strive for a non-trivial usage tradeoff between different types of resources in MEC to minimize their admission cost? In this paper we will address the aforementioned challenges.

In this paper we deal with the cost minimization problem in MEC by admitting user requests of offloading tasks with network function service requirements. We strive for a non-trivial tradeoff between the computing resource consumption and the communication resource consumption in user request admissions dynamically. Intuitively, when computing resources in the system become the bottleneck in terms of their availabilities while the communication resources are sufficient, we may deploy fewer VNF instances for network function services in cloudlets (thereby reducing the usage of computing resources),

then the processing of the data packet traffic of each admitted request takes a longer routing path to its VNF instance and consumes more communication resources. Alternatively, if communication resources are the bottleneck in MEC, we may deploy more VNF instances to ensure each request offloaded to its nearby cloudlet for service processing in order to shorten the routing path of its data packet traffic. Such a non-trivial tradeoff between the usages of different resources can be achieved through the introduction of a *load factor* λ at every VNF instance with $0 < \lambda \leq 1$. That is, if the computing resource is expensive at the moment, we may set the load factor higher such that every VNF instance can serve more requests; otherwise, we may set the load factor lower, and more VNF instances then are instantiated.

The novelty of this work lies in exploring a non-trivial tradeoff between different types of resource usages in MEC to minimize the request admission cost, by introducing the load factor concept. To the best of our knowledge, we are the first to explore non-trivial usage tradeoffs between different types of resources for offloading task request admissions while minimizing their admission cost. That is, we use inexpensive resource to replace expensive resource for request admissions through the load factor concept.

The main contributions of this paper are described as follows. In this paper we study the provisioning of VNF services for offloading tasks in an MEC network by jointly considering service placement and request scheduling. We consider the problem under two scenarios: admit all requests when there are abundant computing resource; or admit as many requests as possible if the computing resource at each cloudlet is capacitated while minimizing the admission cost of all admitted requests. Since the problem is NP-hard, we then provide an ILP solution and devise an efficient algorithm for the problem if there are sufficient computing resources. Otherwise, we develop an algorithm for the throughput maximization problem. We finally evaluate the performance of the proposed algorithms through experimental simulations.

The rest of the paper is organized as follows. Section II introduces notions, notations, and the problem definition. Section III formulates an ILP solution and devises an efficient algorithm for the problem if there are abundant computing resources in MEC. Section IV develops an efficient heuristic for the problem if computing resources in cloudlets are capacitated. Section V evaluates the performance of the proposed algorithms empirically, and Section VI concludes the paper.

II. Preliminaries

In this section, we first introduce the system model, notions and notations. We then define the problems precisely.

A. System model

We consider a metropolitan mobile edge cloud computing network (MEC), which is represented by an undirected graph $G = (V, E)$, where V is a set of access points (APs) located at different locations in the metropolitan region, e.g., schools, shopping malls, bus stations, and hospitals. There is a cloudlet with computing capacity C_v attached with each AP node $v \in V$, for implementing virtualized network functions (VNFs) requested by mobile users. E is the set of wired links between APs. The two endpoints of each link $e \in E$ are connected by a high-speed optical cable, which implies that there is plenty of bandwidth on link e. We assume that each AP node covers a certain area in which mobile users can access the MEC wirelessly through it.

B. User requests with VNF requirements and admission cost

Assume that there is a set U of users accessing MEC through their nearby APs. Each user $j \in U$ issues an offloading task request $r_j = \langle v_j, f_j^{(k)}, \rho_j \rangle$ that demands for a specified VNF service $f_j^{(k)}$, where v_j is the AP location of the request, and ρ_j is its data packet rate. Denote by U_k and $U_{k,v}$ the sets of users requested for network function $f^{(k)}$ in G and at AP $v \in V$, respectively. It can be seen that $U_{k,v} \subset U_k$ and $U_k \subseteq U$ for each k with $1 \leq k \leq K$.

Each VNF is implemented by a VNF instance of its type that consumes a certain amount of computing resource. Without loss of generality, we assume that different types of VNFs among all requests can be classified into K types. The VNF instance of $f_j^{(k)}$ for user request r_j can be implemented in a cloudlet $v_l \in V$ that is not necessarily co-located with its user at AP v_j, i.e., $j \neq l$. Denote by $f^{(k)}$ and $C(f^{(k)})$ the virtualized network function of type k and the amount of computing resource consumed for its VNF instance implementation for each k with $1 \leq k \leq K$. We further assume that each VNF instance of $f^{(k)}$ has a maximum packet processing rate $\mu^{(k)}$, and the data packet traffic of each request is not splittable and must be processed by a single VNF instance.

Instantiating VNF instances at cloudlets consumes computing and storage resources of cloudlets and thus incurs the implementation cost. The instantiation of a VNF instance of network function $f^{(k)}$ in a cloudlet v incurs the instantiation cost $c_{ins}(f^{(k)}, v)$, while the processing of data packet traffic of a request r_j at a VNF instance of $f_j^{(k)}$ at cloudlet v has the computing resource usage cost $\rho_j \cdot c_{proc}(f^{(k)}, v)$, where $c_{proc}(f^{(k)}, v)$ is the cost of processing a packet by a VNF instance $f^{(k)}$ at cloudlet v. In addition to the processing cost of its data packet traffic at a VNF instance in a cloudlet, the data packet traffic of each request r_j is routed along a routing path in the network between the request's AP location and the cloudlet hosting its VNF instance, which incurs the communication cost. Let $P(v_j, v_l)$ be the shortest routing path for request r_j between its AP node v_j and cloudlet v_l in which its data packet traffic is processed by a VNF instance of $f_j^{(k)}$. The routing cost of a packet along path $P(v_j, v_l)$ thus is $c_{bw}(P(v_j, v_l)) = \sum_{e \in P(v_j, v_l)} c_e$, where c_e is the unit transmission cost on each link $e \in E$.

C. Problem formulations

In this paper, we consider two task offloading problems in MEC: one is *the cost minimization problem* if there are sufficient computing resources in cloudlets to meet all user requests' resource demands; and the other is *the throughput maximization problem* that aims to maximize the number of request admissions while minimizing their admission cost, subject to the computing capacity constraint on each cloudlet.

Definition 1: Given an MEC $G = (V, E)$ with a set V of APs, a cloudlet is co-located with each AP for implementing VNF instances, a set of users U with each user $j \in U$ having an offloading task $r_j = \langle v_j, f_j^{(k)}, \rho_j \rangle$ to be offloaded to MEC, *the cost minimization problem* is to admit all user requests through deploying a certain number of VNF instances

of network functions in cloudlets to meet the service demands of all requests such that their admission cost is minimized, assuming that there are sufficient computing resources at cloudlets for the admissions of all requests.

The cost minimization problem can be mathematically formulated as follows. Let $\mathcal{F} = \{f^{(1)}, f^{(2)}, \ldots, f^{(K)}\}$ be the set of network functions provided by the network. Recall that U_k and $U_{k,v}$ are the sets of user requests that request for network function $f^{(k)}$ in the network and at an AP node $v \in V$ respectively. For each type network function k with $1 \leq k \leq K$, let $N^{(k)}$ be the number of VNF instances to be deployed for admitting all requests in U_k. Let $I^{(k)} = \{I_1^{(k)}, I_2^{(k)}, \ldots I_{N^{(k)}}^{(k)}\}$ be the set of VNF instances of $f^{(k)}$. Let $V = \{v_1, v_2, \ldots, v_{|V|}\}$ be the set of APs or cloudlets and $I = \cup_{k=1}^{K} I^{(k)}$.

Define a function $\phi : \mathcal{F} \times I \mapsto V$. Specifically, the variable domain of $\phi(\cdot)$ consists of K disjoint subdomains, i.e., $\phi : \mathcal{F} \times I^{(k)} \mapsto V$ for each k with $1 \leq k \leq K$, the ith VNF instance $I_i^{(k)}$ of $f^{(k)}$ will be deployed to a cloudlet indexed at $\phi(k, i)$. Define another function $\psi : \mathcal{F} \times U \mapsto I$. Specifically, the variable domain of $\psi(\cdot)$ consists of K disjoint subdomains, i.e., $\psi : \mathcal{F} \times U_k \mapsto I^{(k)}$ for each k with $1 \leq k \leq K$, each request $r_j \in U_k$ of $f^{(k)}$ is assigned to the $\psi(k, j)$th VNF instance of $f^{(k)}$, which also implies that the VNF instance is hosted in a cloudlet indexed at $\phi(k, \psi(k, j))$.

Then, the cost minimization problem is to admit all requests while minimizing their admission cost in terms of the VNF instance instantiation cost, the VNF instance processing cost, and the communication cost, that is, $\sum_{k=1}^{K} \sum_{i=1}^{N^{(k)}} c_{ins}(f^{(k)}, v_{\phi(k,i)}) + \sum_{k=1}^{K} \sum_{i=1}^{N^{(k)}} \sum_{r_j \in U_k} \rho_j \cdot \big(c_{bw}(P(v_j, v_{\phi(\psi(k,j))})) + c_{proc}(f^{(k)}, v_{\phi(k,\psi(k,j))})\big)$, and the number $N^{(k)}$ of VNF instances of $f^{(k)}$ for all requests in U_k must meet $\max\{\lceil \frac{\sum_{r_j \in U_k} \rho_j}{\lambda \cdot \mu^{(k)}} \rceil, n_B(\lambda \cdot \mu^{(k)}, U_k)\} \leq N^{(k)} \leq \min\{\sum_{v \in V} n_B(\lambda \cdot \mu^{(k)}, U_{k,v}), |U_k|\}$, where $n_B(\lambda \cdot \mu, U')$ is the minimum number of bins that can pack all elements in U', assuming that each bin has a capacity of $\lambda \cdot \mu$ and λ is a given load factor within $0 < \lambda \leq 1$, and there is a set U' of elements with each element r_j having a profit 1 and weight ρ_j. The correctness of the inequality will be shown in Lemma 1.

Definition 2: Given an MEC $G = (V, E)$ with a set V of APs, a cloudlet $v \in V$ with computing capacity C_v is co-located with every AP v, a set U of users with each user $j \in U$ having an offloading task $r_j = \langle v_j, f_j^{(k)}, \rho_j \rangle$, *the throughput maximization problem* is to maximize the number of requests admitted while minimizing the total admission cost, subject to the computing capacity on each cloudlet in G.

The two defined problems are NP-hard, which can be shown by a reduction from the generalized assignment problem [2]. The detailed reduction is omitted, due to space limitation.

III. Algorithm for the cost minimization problem

In this section, we deal with the cost minimization problem. We first formulate the problem as an integer linear program (ILP). We then devise an efficient algorithm for it.

A. Integer linear program formulation

For each request $r_j \in U$, there are K ($K = |\mathcal{F}|$) binary constants a_j^k where $a_j^k = 1$ if r_j requests network function $f^{(k)}$; $a_j^k = 0$ otherwise for all k with $1 \leq k \leq K$. For the ith VNF instance $I_i^{(k)} \in I^{(k)}$ of $f^{(k)}$ and each cloudlet $v_l \in V$ with $1 \leq i \leq N^{(k)}$, $1 \leq k \leq K$, and $1 \leq l \leq |V|$, there is a binary decision variable $x_{i,k,l}$ where $x_{i,k,l} = 1$ if the ith VNF instance of $f^{(k)}$ is deployed in cloudlet v_l. Furthermore, there is a binary decision variable $y_{i,j,l}$, which indicates if the data packet traffic of a request r_j is processed by the ith VNF instance $I_i^{(k)}$ of $f^{(k)}$ in cloudlet v_l. The optimization objective (1) of the cost minimization problem is to minimize the admission cost of all requests in G, i.e.,

$$minimize \quad \sum_{k=1}^{K} \sum_{i=1}^{N^{(k)}} \sum_{l=1}^{|V|} c_{ins}(f^{(k)}, v_l) \cdot x_{i,k,l} + \sum_{r_j \in U} \sum_{k=1}^{K} \sum_{i=1}^{N^{(k)}} \sum_{l=1}^{|V|} \rho_j a_j^k (c_{proc}(f^{(k)}, v_l) + c_{bw}(P(v_j, v_l))) \cdot y_{i,j,l}, \tag{1}$$

subject to:

$$\sum_{i=1}^{N^{(k)}} \sum_{l=1}^{|V|} a_j^k \cdot y_{i,j,l} = 1, \qquad \forall r_j \in U,\ 1 \leq k \leq K \tag{2}$$

$$\sum_{r_j \in U} \rho_j \cdot a_j^k \cdot y_{i,j,l} \leq \lambda \cdot \mu^{(k)} \cdot x_{i,k,l}, \qquad \forall v \in V,\ 1 \leq k \leq K,\ 1 \leq i \leq N^{(k)} \tag{3}$$

$$\max\{\lceil \frac{\sum_{r_j \in U_k} \rho_j}{\lambda \cdot \mu^{(k)}} \rceil, n_B(\lambda \cdot \mu^{(k)}, U_k)\} \leq N^{(k)} \leq \min\{\sum_{v \in V} n_B(\lambda \cdot \mu^{(k)}, U_{k,v}), |U_k|\}, \forall\ 1 \leq k \leq K \tag{4}$$

$$a_j^k \cdot y_{i,j,l} \leq x_{i,k,l}, \forall 1 \leq l \leq |V|,\ r_j \in U,\ 1 \leq k \leq K,\ 1 \leq i \leq N^{(k)} \tag{5}$$

$$x_{i,k,l} \in \{0, 1\}, \forall 1 \leq l \leq |V|, 1 \leq k \leq K, 1 \leq i \leq N^{(k)} \tag{6}$$

$$y_{i,j,l} \in \{0, 1\}, \forall 1 \leq l \leq |V|,\ r_j \in U,\ 1 \leq i \leq N^{(k)}. \tag{7}$$

where $n_B(\lambda \cdot \mu, U')$ is the minimum number of bins with capacity $\lambda \cdot \mu$ to pack all elements in U', and $u(k, v)$ is an approximation (an upper bound) on $n_B(\lambda \cdot \mu^{(k)}, U_{k,v})$ that can be obtained by invoking an approximation algorithm for the GAP problem such as the one in [2] (see Lemma 1).

Within the ILP, constraint (2) ensures that each request r_j will be admitted and assigned to one VNF instance of its requested network function $f^{(k)}$ at a cloudlet v_l. Constraint (3) ensures that the accumulative packet rate of all requests requested for $f^{(k)}$ that are assigned to the ith VNF instance of $f^{(k)}$ in cloudlet v_l is no greater than the maximum packet processing rate $\lambda \cdot \mu^{(k)}$ of the VNF instance. Constraint (4) ensures that all requests in U_k requesting for $f^{(k)}$ can be admitted and their data packet traffic can be processed by these $N^{(k)}$ VNF instances for each k. Constraint (5) enforces that if the data packet traffic of request r_j is processed in the ith VNF instance of $f^{(k)}$ in cloudlet v_l, then the ith VNF instance of $f^{(k)}$ must be deployed in cloudlet v_l.

The rest is to show why the value of $N^{(k)}$ is in a given range by the following lemma.

Lemma 1. *For each k with $1 \leq k \leq K$, we have* $\max\{\lceil \frac{\sum_{r_j \in U_k} \rho_j}{\lambda \cdot \mu^{(k)}} \rceil, n_B(\lambda \cdot \mu^{(k)}, U_k)\} \leq N^{(k)} \leq \min\{|U_k|,$

$\sum_{l=1}^{|V|} n_B(\lambda \cdot \mu^{(k)}, U_{k,v_l})\} \leq \min\{\sum_{l=1}^{|V|} u(k, v_l), |U_k|\}$.

Proof: To admit all requests in U_k, we first estimate the lower and the upper bounds on the number of VNF instances of $f^{(k)}$ needed. Denote by $N_{low}^{(k)}$ and $N_{upp}^{(k)}$ the lower and upper bounds on $N^{(k)}$. It can be seen that a naive lower bound of $N^{(k)}$ is $\lceil \frac{\sum_{r_j \in U_k} \rho_j}{\lambda \cdot \mu^{(k)}} \rceil$, as the maximum packet processing rate of a VNF instance is $\lambda \cdot \mu^{(k)}$. However, this bound might not be tight. To ensure that all requests in U_k will be admitted, the minimum number of VNF instances of $f^{(k)}$ needed is equivalent to the minimum number of bins to pack the data packet rates of these requests. Unfortunately the calculation of this minimum number of bins is NP-hard. We instead adopt an approximation algorithm for the GAP [2] to find an approximate value of $n_B(\lambda \cdot \mu^{(k)}, U_k)$, which proceeds as follows. We start with $n_L^{(k)} = n_B^{(k)} = \lceil \frac{\sum_{r_j \in U_k} \rho_j}{\lambda \cdot \mu^{(k)}} \rceil$ bins to pack all requests in U_k. If all requests can be packed, this value is optimal, i.e., $n_B(\lambda \cdot \mu^{(k)}, U_k) = n_L^{(k)}$; otherwise, we set an upper bound $n_H = 2n_L$ of $n_B(\lambda \cdot \mu^{(k)}, U_k)$, we then examine whether all requests in U_k can be packed by $n_H^{(k)}$ bins. If not, we double that number again. Otherwise, we find a proper number for the approximation of $n_B(\lambda \cdot \mu^{(k)}, U_k)$ by binary search in $[n_L^{(k)}, n_H^{(k)}]$.

On the other hand, if the data packet traffic of each request is processed in the cloudlet co-located with the AP it was issued, this will not incur any routing cost of its admission. Thus, there must be enough VNF instances in that cloudlet for the request admissions. Consider the sum of data packet rates $\sum_{r_j \in U_{k,v}} \rho_j$ of all requests in $U_{k,v}$ at AP v. Determining the minimum number of VNF instances at cloudlet v for the request admissions is equivalent to determining the minimum number of bins $n_B(\lambda \cdot \mu^{(k)}, U_{k,v})$ to pack the data packet rates of these requests with bin capacity $\mu^{(k)}$. The range of the value of $n_B(\lambda \cdot \mu^{(k)}, U_{k,v})$ is between $\lceil \frac{\sum_{r_j \in U_{k,v}} \rho_j}{\lambda \cdot \mu^{(k)}} \rceil$ and $|U_{k,v}|$. However, finding its exact value is NP-hard. Thus, a naive upper bound $N_{upp}^{(k)}$ on $N^{(k)}$ is $|U_k|$, an improved upper bound is to apply an approximation algorithm for the GAP at each AP [2] to identify the number of bins needed. Let $u(k, v)$ be the solution delivered by the approximation algorithm at cloudlet $v \in V$. Then, $N_{upp}^{(k)} = \sum_{v \in V} u(k, v)$. ■

Since the admission cost of all requests in U_k is the sum of the processing and routing costs of the requests that is not a monotonic function of the number of VNF instances, we need to find a proper value of $N^{(k)}$ for each k with $1 \leq k \leq K$ such that the admission cost of all requests in U is minimized. Thus, the exact solution is obtained, by exploring each integer value in the interval, and then choosing the one with the minimum cost as the problem solution.

B. Algorithm

Although the formulated ILP for the cost minimization problem can deliver an exact solution when the problem size is small, its running time is prohibitively high with the growth of problem size, and thus not scalable. In the following we propose an efficient algorithm for the cost minimization problem.

The rationale behind the proposed algorithm is to explore the non-trivial tradeoff between the usages of computing and communication resources in request admissions dynamically. If computing resource is relatively inexpensive, we can increase the deployment of the number $N^{(k)}$ of VNF instances by lowering its load factor λ (e.g., $\lambda = 0.5$). Thus, most requests can be served by the VNF instances instantiated at their nearby cloudlets, thereby reducing the routing cost of their data packet traffic. On the other hand, if computing resource becomes more expensive (less computing resource is available), we can reduce the deployment of the number $N^{(k)}$ of VNF instances by increasing its load factor λ (e.g., $\lambda = 0.95$). Thus, less computing resource is consumed on VNF instance instantiations, while the routing path of each request from its location to the hosting cloudlet becomes longer. Thus, more communication resource is consumed to accommodate the request.

The proposed algorithm is an iterative algorithm. Specifically, for each k with $1 \leq k \leq K$, we identify a proper value $N^{(k)}$, and deploy the $N^{(k)}$ VNF instances of $f^{(k)}$ to different cloudlets. We deploy the VNF instances one by one and assign a subset of unassigned requests in U_k to the newly deployed VNF instance. This procedure continues until all requests in U_k are assigned. Consider the deployment of the ith VNF instance of $f^{(k)}$ that can be deployed in one of the $|V|$ cloudlets. Let $R_{v_l}^{(k)}$ be the set of requests in U_k' requested for $f^{(k)}$ that have not been admitted in the previous $(i-1)$ iterations of the algorithm but will be admitted in the ith iteration and processed by the ith VNF instance of $f^{(k)}$ in cloudlet v_l, where $U_k' \subseteq U_k$. In order to identify which requests in U_k' to be assigned to the ith newly instantiated VNF instance $I_i^{(k)}$ of $f^{(k)}$ in cloudlet v_l, we aim to assign as many requests as possible in U_k' to $I_i^{(k)}$ such that the sum of the processing and routing costs of these requests is minimized, subject to the maximum packet processing rate $\lambda \cdot \mu^{(k)}$ of $I_i^{(k)}$.

If all VNF instances of $f^{(k)}$ have been instantiated in cloudlets, denote by $cost(r_j) = \rho_j \cdot (c_{proc}(f^{(k)}, v_l) + c_{bw}(P(v_j, v_l)))$ the processing and routing cost of admitting request r_j. To this end, we treat $I_i^{(k)}$ as a bin with capacity $\lambda \cdot \mu^{(k)}$, and each request $r_j \in U_k'$ as an element in U_k' with size ρ_j, and a profit $profit(r_j) = \frac{1}{cost(r_j)}$ that is the inverse of the processing and routing cost $cost(r_j)$ of a newly admitted request, assuming that instance $I_i^{(k)}$ has been instantiated in cloudlet $v_l \in V$ already. A solution $R_{v_l}^{(k)}$ $(\subseteq U_k')$ to this knapsack problem is obtained by applying an approximation algorithm for the knapsack problem. Notice that the profit maximization in this knapsack problem is roughly equivalent to the cost minimization of admitting all requests in $R_{v_l}^{(k)}$. Furthermore, the instantiation cost of $I_i^{(k)}$ in cloudlet v_l has not been taken into account. The rest is to choose a cloudlet v_{l_0} hosting $I_i^{(k)}$ if the ratio of the number of newly admitted requests to the admission cost of the requests in $R_{v_{l_0}}^{(k)}$ is maximized, i.e.,

$$v_{l_0} = \operatorname*{argmax}_{v_l \in V} \frac{|R_{v_l}^{(k)}|}{c_{ins}(f^{(k)}, v_l) + \sum_{r_j \in R_{v_l}^{(k)}} cost(r_j)}. \quad (8)$$

The detailed algorithm for the cost minimization problem is given in `Algorithm 1`.

C. Analysis of the proposed algorithm

Theorem 1. *Given an MEC $G = (V, E)$, and a set U of users with each $j \in U$ having an offloading task $r_j =$*

Algorithm 1 An algorithm for the cost minimization problem

Input: Given an MEC $G = (V, E)$ and a set U of users with each $j \in U$ having an offloading task $r_j = \langle v_j, f_j^{(k)}, \rho_j \rangle$, and a given load factor λ with $0 < \lambda \leq 1$.

Output: A solution to minimize the admission cost of all requests.

1: $cost \leftarrow 0$; /* the cost of all request admissions */
2: $S \leftarrow \emptyset$; /* different VNF instance placements in cloudlets */
3: $A \leftarrow \emptyset$; /* different user request assignment to different VNF instances at different cloudlets */
4: **for** $k \leftarrow 1$ to K **do**
5: $i \leftarrow 0$; /* the ith VNF instance $I_i^{(k)}$ of $f^{(k)}$ to be deployed */
6: $U'_k \leftarrow U_k$; /* the set of unassigned requests in U_k */
7: $I^{(k)} \leftarrow \emptyset$; /* the set of VNF instances of $f^{(k)}$ */
8: **while** $U'_k \neq \emptyset$ **do**
9: $i \leftarrow i + 1$;
10: Compute the ratio in formula (8) if the ith VNF instance $I_i^{(k)}$ of $f^{(k)}$ is deployed in cloudlet v_l for each $v_l \in V$ by invoking an approximation algorithm for the knapsack problem with capacity $\lambda \cdot \mu^{(k)}$ and set U'_k, and let cloudlet v_{l_0} be chosen to the host of the ith VNF instance $I_i^{(k)}$;
11: $I^{(k)} \leftarrow I^{(k)} \cup \{I_i^{(k)}\}$;
12: $U'_k \leftarrow U'_k \setminus R_{v_{l_0}}^{(k)}$; /* all requests in $R_{v_{l_0}}^{(k)}$ be admitted and assigned to VNF instance $I_i^{(k)}$ in cloudlet v_{l_0} */
13: $cost \leftarrow cost + c_{ins}(f^{(k)}, v_{l_0}) + \sum_{r_j \in R_{v_{l_0}}^{(k)}} cost(r_j)$;
14: $S \leftarrow S \cup \{I_i^{(k)}$ in cloudlet $v_{l_0}\}$; /* i.e., $\phi(k, i) = l_0$ */
15: $A \leftarrow A \cup \{r_j$ is assigned to $I_i^{(k)}$ in cloudlet $v_{l_0} \mid r_j \in R_{v_{l_0}}^{(k)}\}$; /* i.e., $\psi(k, j) = i$ and $\phi(k, \psi(k, j)) = l_0$ */
16: **return** $cost$, S, and A.

$\langle v_j, f_j^{(k)}, \rho_j \rangle$, assuming that there are sufficient computing resources in cloudlets to accommodate all VNF instances of different network functions, there is an efficient algorithm, `Algorithm 1`*, for the cost minimization problem, which takes $O(|V| \cdot \frac{|U| \log |U|}{\epsilon} + |V|^3)$ time and delivers a feasible solution to the problem, where ϵ is a constant with $0 < \epsilon \leq 1$.*

Proof: As there are K iterations of `Algorithm 1`. For iteration k with $1 \leq k \leq K$, all requests in U_k are admitted, and $N^{(k)}$ VNF instances of $f^{(k)}$ are deployed, and the sum of packet rates of all requests in U_k assigned to any of the $N^{(k)}$ VNF instances is no greater than its maximum packet processing rate $\lambda \cdot \mu^{(k)}$. The solution delivered by `Algorithm 1` thus is feasible.

The time complexity analysis of `Algorithm 1` is omitted, due to space limitation. ■

IV. Algorithm for the throughput maximization problem

In this section, we study *the throughput maximization problem* with the aim to maximize the throughput, by admitting as many requests as possible while minimizing their admission cost, subject to computing capacity of each cloudlet.

A. Algorithm

We propose a greedy algorithm for the problem, which proceeds iteratively. Within iteration i, one VNF instance of a network function $f^{(k_0)}$ deployed at a cloudlet v_{l_0} is chosen if its ratio η_{k_0,l_0} of the number of newly admitted requests that request for $f^{(k_0)}$ to their admission cost is the maximum one, assuming that cloudlet v_{l_0} has sufficient residual computing resource to accommodate the VNF instance of $f^{(k_0)}$, where both network function $f^{(k_0)}$ and cloudlet v_{l_0} for the new VNF instance deployment are determined by the following formula.

$$\eta_{k_0,l_0} = \underset{l,k}{\operatorname{argmax}} \frac{|R_{v_l}^{(k)}|}{c_{ins}(f^{(k)}, v_l) + \sum_{r_j \in R_{v_l}^{(k)}} cost(r_j)}, \quad (9)$$

where $R_{v_l}^{(k)}$ is the set of requests of $f^{(k)}$ that have not been admitted in previous iterations and will be admitted and processed by the VNF instance of $f^{(k)}$ in cloudlet v_l for each k with $1 \leq k \leq K$. Notice that identifying the set $R_{v_l}^{(k)}$ is NP-hard, due to this is a knapsack problem with bin capacity $\lambda \cdot \mu^{(k)}$ and the set of elements in U'_k ($\subseteq U_k$), where each element $r_j \in U'_k$ has a size ρ_j and a profit $\frac{1}{\rho_j \cdot (c_{proc}(f^{(k)}, v_l) + c_{bw}(P(v_j, v_l)))}$. The admission cost of newly admitted requests in $R_{v_0}^{(k_0)}$ is the sum of the instantiation cost of the VNF instance of $f^{(k_0)}$ in cloudlet v_{l_0}, and the sum of the processing and routing costs of all newly admitted requests in $R_{v_0}^{(k_0)}$, using the newly instantiated VNF instance.

The above procedure repeats until either all requests in U are admitted or no more VNF instances can be instantiated in any cloudlet without violating its computing capacity. The detailed algorithm is given in `Algorithm 2`.

B. Algorithm analysis

Theorem 2. *Given an MEC $G = (V, E)$, there is a cloudlet $v \in V$ with computing capacity C_v co-located with an AP v, a set U of users with each $j \in U$ having an offloading task request $r_j = \langle v_j, f_j^{(k)}, \rho_j \rangle$, there is an algorithm,* `Algorithm 2`*, for the throughput maximization problem in G, which takes $O(K \cdot |V| \cdot |U|^2 \log |U|/\epsilon + |V|^3)$ time, where ϵ is constant with $0 < \epsilon \leq 1$.*

The time complexity analysis of `Algorithm 2` is omitted, due to space limitation.

V. Performance evaluation

In this section, we evaluate the performance of the proposed algorithms through experimental simulations.

A. Experimental environment settings

We consider an MEC $G = (V, E)$ consisting of from 10 to 250 APs (cloudlets). We assume that the computing capacity of each cloudlet varies from 3,000 MHz to 6,000 MHz [5]. Given a cloudlet, the instantiation cost of a VNF instance in it is randomly drawn from $[0.50, 2.0]$, while the processing cost of per packet data traffic by a VNF instance is a random value within $[0.01, 0.1]$. The routing cost per data packet along a link is a value drawn from $[0.01, 0.1]$. The number K of different types of VNFs in MEC is set at 6, and computing demands of different types of VNFs is set from 400 MHz to 1,000 MHz [5], while the processing rate (capacity) of one type of VNF instance is randomly drawn from 50 to 100 data packets per millisecond [10]. Each offloading task request r_j is randomly generated as follows. A node $v_j \in V$ in G is randomly chosen as its AP r_j, its data packet rate ρ_j is randomly drawn from 2 to 10 packets per millisecond [6], and its type of VNF $f_j^{(k)} \in \mathcal{F}$ is randomly chosen from one of the K network functions. The value in each figure is the mean of the results out of 30 MEC instances of the same size. The running time of an algorithm is obtained based on a machine with 3.4 GHz Intel i7 Quad-core CPU and 16GB RAM.

Algorithm 2 An algorithm for the throughput maximization problem

Input: Given an MEC $G = (V, E)$, a set of users U with each $j \in U$ having an offloading task $r_j = \langle v_j, f_j^{(k)}, \rho_j \rangle$, and a given load factor λ with $0 < \lambda \leq 1$.

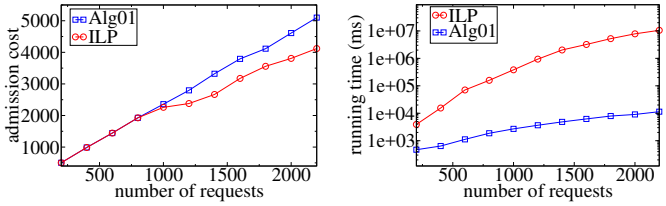
Output: A solution that maximizes the throughput while minimizing the admission cost of admitted requests.

$flag \leftarrow true$;
$cost \leftarrow 0$; /* the admission cost of all admitted requests */
$S \leftarrow \emptyset$; /* different VNF instance placements in cloudlets */
$A \leftarrow \emptyset$; /* all admitted requests so far, and different user request assignment to different VNF instances at different cloudlets */
while $flag$ **do**
  /* if there are unadmitted requests and there are computing resources in cloudlets for VNF instance instantiation */
  $\eta_{max} \leftarrow 0$;
  $sign \leftarrow' No'$; /* terminate the nested loop */
  for $k \leftarrow 1$ to K **do**
    for each cloudlet $v_l \in V$ **do**
      $U'_k \leftarrow U_k - (U_k \cap A)$; /* unassigned requests in U_k */
      if (cloudlet v_l has residual computing capacity to accommodate a VNF instance of $f^{(k)}$) and ($U'_k \neq \emptyset$) **then**
        Compute $R_{v_l}^{(k)}$, by invoking an approximation algorithm for the knapsack problem with capacity $\lambda \cdot \mu^{(k)}$ and set U'_k;
        Compute the ratio $\eta_{k,l}$ in Eq. (9);
        if $\eta_{k,l} > \eta_{max}$ **then**
          $\eta_{max} \leftarrow \eta_{k,l}$; $k_0 \leftarrow k$; $l_0 \leftarrow l$;
          $sign \leftarrow' Yes'$; /* whether there is any update */
  if $sign =' Yes'$ **then**
    $C_{v_{l_0}} \leftarrow C_{v_{l_0}} - C(f^{(k_0)})$; /* update the residual computing resource at cloudlet v_{l_0} */
    $U_{k_0} \leftarrow U_{k_0} \setminus R_{v_{l_0}}(k_0)$;
    $cost \leftarrow cost + c_{ins}(f^{(k_0)}, v_{l_0}) + \sum_{r_j \in R_{v_{l_0}}^{(k_0)}} \rho_j \cdot (c_{proc}(f^{(k_0)}, v_{l_0}) + c_{bw}(P(v_j, v_{l_0})))$;
    $S \leftarrow S \cup \{$a VNF instance of $f^{(k_0)}$ created in cloudlet $v_{l_0}\}$; /* i.e., its ith instance, $\phi(k_0, i) = l_0$ */
    $A \leftarrow A \cup \{r_j$ is assigned to $I_i^{(k_0)}$ in cloudlet $v_{l_0} \mid r_j \in R_{v_{l_0}}^{(k_0)}\}$; /* i.e., $\psi(k_0, j) = i$ and $\phi(k_0, \psi(k_0, j)) = l_0$ */
  else
    $flag \leftarrow false$;
return $cost$, S, and A.

In the following, we first evaluate the performance of `Algorithm 1` for the cost minimization problem against its optimal solution - the ILP solution in the small-scale and a baseline heuristic `GreedyNonCap` respectively. Algorithm `GreedyNonCap` examines requests one by one, and a request is admitted if there is a cloudlet with sufficient computing resource to accommodate its VNF instance. If there are multiple such cloudlets, we then choose the cloudlet with the minimum admission cost. Otherwise, if there is no VNF instance of its type has sufficient residual processing capacity for the request, a new VNF instance with the minimum instantiation cost will be instantiated in some cloudlet. This process continues until all user requests being admitted. Another benchmark heuristic `GreedyCap` is also proposed, which follows similar idea with the one for algorithm `GreedyNonCap`, omitted.

B. Performance evaluation of algorithms for the cost minimization problem

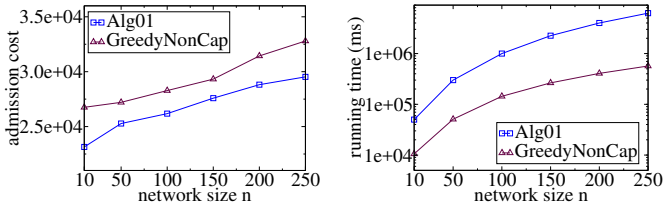
We first investigate the performance of `Algorithm 1` against the optimal solution of the ILP for the cost minimization problem, by varying the number of requests from 200 to 2,200 while fixing the number $|V|$ of APs at 10. Fig. 1 illustrates the performance curves of the two mentioned algorithms. It can be seen from Fig. 1(a) that `Algorithm 1` can achieve a near optimal admission cost, i.e., its admission cost is no more than 120.79% of the optimal one. Fig. 1(b) demonstrates the running time of these two algorithms. It can be seen that the running time of `Algorithm 1` is only a small fraction of that of algorithm `ILP`, while its solution is comparable with the optimal one. In particular, `Algorithm 1` takes less than 20 seconds while the `ILP` solution takes more than two hours, when the number of requests reaches 2,200. With the increase on the number of requests, the running time of algorithm `ILP` grows exponentially, and the algorithm is only applicable when the problem size is small.



(a) The admission cost (b) The running time

Fig. 1. Performance of `Algorithm 1` and algorithm `ILP` for the cost minimization problem, by varying the number of requests from 200 to 2,200.

We then evaluate the performance of `Algorithm 1` against the benchmark heuristic `GreedyNonCap`, by varying network size from 10 to 250 for 10,000 user requests. Fig. 2 depicts the performance curves of the two mentioned algorithms. From Fig. 2(a) we can see that `Algorithm 1` outperforms its benchmark counterpart `GreedyNonCap`. Specifically, with the increase on network size, the admission costs of both algorithms grow too. However, the performance gap between them becomes larger and larger. As shown in the figure, `Algorithm 1` has 9.43% and 12.95% less admission cost than that by algorithm `GreedyNonCap` when the network size is 100 and 250, respectively. Fig. 2(b) shows the running time curves of the two algorithms. It can be seen that algorithm `GreedyNonCap` takes less time than algorithm `Algorithm 1` in all network sizes. This is due to the fact that `Algorithm 1` strives for finding a set of requests to share an existing VNF instance, while algorithm `GreedyNonCap` only places each request greedily to a VNF instance with the minimum admission cost.



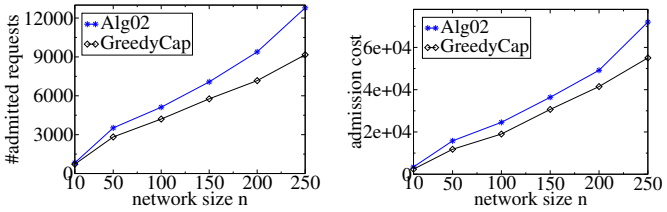
(a) The admission cost (b) The running time

Fig. 2. Performance of `Algorithm 1` and algorithm `GreedyNonCap` when admitting 10,000 requests, by varying the number of APs from 10 to 250.

C. Performance evaluation of algorithms for the throughput maximization problem

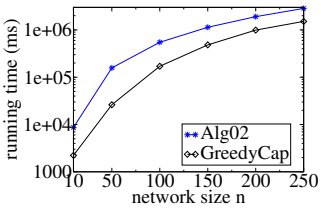
In the following we study the performance of `Algorithm 2` for the throughput maximization problem against a baseline heuristic `GreedyCap`, by varying the network size from 10 to 250 for 20,000 requests. The results are shown in Fig. 3. We can see from Fig. 3 that both algorithms deliver solutions with increasing throughput, along with the increase in the network size. However, `Algorithm 2`

outperforms algorithm GreedyCap significantly. Specifically, Algorithm 2 can admit on average 3,522.73 requests with admission cost 15796.84, while algorithm GreedyCap can only admit on average 2,823.50 requests with admission cost 11,805.73. Algorithm 2 can admit 34.70% more requests than that by algorithm GreedyCap with only 25.91% more admission cost, when the network size is 50. Fig. 3(b) plots the running time curves of both algorithms, where algorithm GreedyCap takes less time than Algorithm 2 in all network sizes.



(a) The network throughput

(b) The admission cost

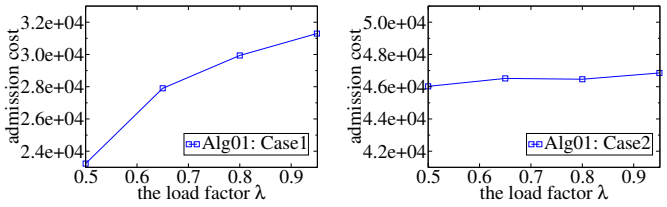


(c) The running time

Fig. 3. Performance of Algorithm 2 and algorithm GreedyCap by varying the number of APs from 10 to 250.

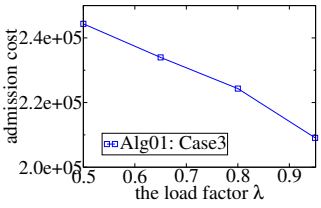
D. Impact of λ on the performance of Algorithm 1

We finally evaluate the impact of the load factor λ on the performance of Algorithm 1 for a set of 10,000 requests in MEC with 100 APs, and we draw the value of λ between 0.5 and 0.95. The impact of λ on the algorithm performance is shown in Fig. 4. We investigate the impact of λ on the algorithm performance by the following three cases.



(a) Case 1

(b) Case 2



(c) Case 3

Fig. 4. Impact of the load factor λ on the performance of Algorithm 1.

Case (1). The instantiation cost of a VNF instance is much cheaper than the bandwidth usage cost. In this case, the cost of a unit computing resource usage in a VNF instance instantiation is drawn from $[0.01, 0.1]$. As a VNF instance can be shared by multiple requests of its type, the instantiation cost of a VNF instance is much cheaper than the usage cost per unit bandwidth. As can be seen from Fig. 4(a), when the instantiation cost of a VNF instance is cheap, a small value of the load factor λ implies more VNF instances can be instantiated in each cloudlet. Case (2). The instantiation cost of a VNF instance is comparable with the bandwidth usage cost. In this case, the cost of a unit computing resource usage in a VNF instance instantiation is drawn from $[0.5, 2.0]$. It can be seen from Fig. 4(b) that the admission cost of requests fluctuates but keeps steady with the changes of load factor λ. Case (3). The instantiation cost of a VNF instance is much more expensive than the bandwidth usage cost. In this case, the cost per unit of computing resource consumed in a VNF instance instantiation is drawn from $[5.0, 20.0]$. As can be seen from Fig. 4(c) that the admission cost of requests decreases rapidly with the increase on the load factor λ. For example, when the load factor λ is set at 0.5 and 0.95, the admission cost is 244,357.76 and 209,091.10, respectively. In other words, around 14.43% of the admission cost can be saved by fine tuning on the value of the load factor λ.

VI. Conclusions

In this paper we studied the provisioning of VNF services for mobile users by offloading their tasks to an MEC network for processing. We aim to maximize the number of request admissions while minimizing their admission cost. We first formulated the cost minimization problem for request admissions by formulating an ILP solution and an efficient algorithm to it. We then dealt with the throughput maximization problem by admitting as many requests as possible, subject to computing capacity at each cloudlet, for which we devised an efficient algorithm. We finally evaluated the performance of the proposed algorithms through experimental simulations. Experimental results demonstrate that the proposed algorithms are promising.

References

[1] A. Ceselli, M. Premoli, and S. Secci. Mobile edge cloud network design optimization. *IEEE/ACM Transactions on Networking*, vol.25, no.3, pp.1818–1831, 2017.

[2] R. Cohen *et al.* An efficient approximation for the generalized assignment problem. *Info. Proc. Lett.*, vol.100, pp.162–166, Elsevier, 2006.

[3] T. He, H. Khamfroush, S. Wang, T. La Porta, and S. Stein. It's hard to share: joint service placement and request scheduling in edge clouds with sharable and non-sharable resources. *Proc. of ICDCS*, IEEE, 2018.

[4] M. Huang *et al.* Throughput maximization of delay-sensitive request admission via virtualized network function placements and migrations.. *Proc. of ICC*, IEEE, 2018.

[5] M. Jia *et al.* QoS-aware task offloading in distributed cloudlets with virtual network function services. *Proc. of MSWiM*, ACM, 2017.

[6] Y. Li, L. T. X. Phan, and B. T. Loo. Network functions virtualization with soft real-time guarantees. *Proc. of INFOCOM*, IEEE, 2016.

[7] Y. Ma, W. Liang, and S. Guo. Mobility-aware delay-sensitive service provisioning for mobile edge computing. *Proc. of INFOCOM WKSHPs IOT4Health*, IEEE, 2019.

[8] Y. Ma, W. Liang, Z. Xu, and S. Guo. Profit maximization for admitting requests with network function services in distributed clouds. To appear in *IEEE Transactions on Parallel and Distributed Systems*.

[9] Y. Mao *et al.* A survey on mobile edge computing: the communication perspective. *IEEE Commun. Surv. Tutor.*, vol.19, pp.2322–2358, 2017.

[10] J. Martins *et al.* ClickOS and the art of network function virtualization. *Proc. of NSDI*, USENIX, 2014.

[11] Z. Xu, W. Liang, M. Jia, M. Huang, and G. Mao. Task offloading with network function services in a mobile edge-cloud network. To appear in *IEEE Transactions on Mobile Computing*.

[12] Z. Xu, W. Liang, W. Xu, M. Jia, and S. Guo. Efficient algorithms for capacitated cloudlet placements. *IEEE Transactions on Parallel and Distributed Systems*, vol.27, no.10, pp.2866–2880, 2016.