Virtual Network Function Service Provisioning in MEC via Trading Off the Usages between Computing and Communication Resources

Yu Ma, Weifa Liang, *Senior Member, IEEE,* Meitian Huang, Wenzheng Xu, *Member, IEEE,* and Song Guo, *Fellow, IEEE*

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 first study the provisioning of virtualized network function (VNF) services for user requests in an MEC network, where each user request has a demanded data packet rate with a specified network function service requirement, and different user requests need different services that are represented by virtualized network functions instantiated in cloudlets. We aim to maximize the number of user request admissions while minimizing their admission cost, where the request admission cost consists of the computing cost on instantiations of requested VNF instances and the data packet traffic processing of requests in their VNF instances, and the communication cost of routing data packet traffic of requests between users and the cloudlets hosting their requested VNF instances. We study the joint VNF instance deployment and user requests assignment in MEC, by explicitly exploring a non-trivial usage tradeoff between different types of resources. To this end, we first formulate the cost minimization problem that admits all requests by assuming that there is sufficient computing resource in MEC to accommodate the requested VNF instances of all requests, for which we formulate an Integer Linear Programming solution and two efficient heuristic algorithms. We then deal with the problem under the computing resource constraint. We term this problem as the throughput maximization problem by admitting as many as requests, subject to computing resource capacity on each cloudlet, for which we formulate an ILP solution when the problem size is small; otherwise, we devise efficient algorithms for it. We finally evaluate the performance of the proposed algorithms through experimental simulations. Experimental results demonstrate that the proposed algorithms are promising. To the best of our knowledge, we are the first to explicitly explore the usage tradeoff between computing and communication resources in the admissions of user requests in MEC through introducing a novel load factor concept to minimize the request admission cost and maximize the network throughput.

Index Terms—Mobile edge computing networks (MEC); network function virtualization (NFV) services; resource allocations of cloudlets; request admission cost minimization; throughput maximization; generalized assignment problem (GAP); VNF instance placement and sharing; usage tradeoffs between computing and communication resources.

1 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, ameliorating their limited computing, storage and battery capacities. However, clouds are usually far away from most end-users, resulting in high 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 network in close proximity to mobile users, is a promising technology to reduce response delays, ensure network operation efficiency, and improve user service satisfaction. Meanwhile, Network Function Virtualization (NFV) has been envisaged as the next-generation networking paradigm. It leverages generic server/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.

To meet various service demands from mobile users, network service providers usually instantiate frequently demanded virtualized network function (VNF) instances 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 user requests? how to optimally place the services in cloudlets and assign which requests to the services in order to minimize their admission costs? and how to strive for a non-trivial usage tradeoff between

- *Y. Ma, W. Liang, and M. Huang are with Research School of Computer Science, The Australian National University, Canberra, ACT 2601, Australia. E-mails: yu.ma@anu.edu.au, wliang@cs.anu.edu.au, and meitian.huang@anu.edu.au*
- *W. Xu is with College of Computer Science, Sichuan University, Chengdu, 610065, P.R. China. E-mail: wenzheng.xu3@gmail.com*
- *S. Guo is with the Hong Kong Polytechnic University Shenzhen Research Institute and Department of Computing, The Hong Kong Polytechnic University, Hong Kong. E-mail: song.guo@polyu.edu.hk*

Manuscript received September XX, 2019; revised YY, 2020.

different types of resources in MEC in the admissions of user requests to minimize the request 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 with network function service requirements. We strive for *a non-trivial tradeoff* between the computing resource consumption and the communication resource consumption in admissions of user requests dynamically. Intuitively, when computing resource in the system becomes the bottleneck in terms of its availability while the communication resource is sufficient, we may deploy fewer VNF instances for network function services in cloudlets, thereby reducing the usage of computing resource, 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 resource. Alternatively, if communication resource is the bottleneck in MEC, we may deploy more VNF instances to ensure each user request routed 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 available. Thus, the routing path of each request between its location and the VNF instance to process its data packet traffic becomes shorter. Notice that during the course of network operations, the consumptions of the two mentioned resources dynamically change overtime.

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 very first to explore non-trivial usage tradeoffs between different types of resources for user request admissions while minimizing their admission cost or maximizing the network throughput. 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 user requests in an MEC network by jointly considering service placement and request scheduling, where each user request has a demanded packet rate and a specified network function service requirement. We aim to maximize the number of request admissions while minimizing their admission cost. We consider the problem under two scenarios: admit all requests when there are abundant computing resource; or admit as many as requests if the computing resource at each cloudlet is capacitated while minimizing the admission cost of admitted requests. Specifically, we first formulate the cost minimization problem. Since the problem is NP-hard, we then provide an ILP solution and devise two efficient algorithms for the problem if there are sufficient computing resource in the MEC network. Otherwise, we formulate an ILP solution for the throughput maximization problem when the problem size is small, and develop two efficient algorithms for it. We finally evaluate the performance of the proposed algorithms through experimental simulations. Experimental results demonstrate that the proposed algorithms are promising.

The rest of the paper is organized as follows. Section 2 deals with related works. Section 3 introduces notions, notations, and the problem definitions. Section 4 formulates an ILP solution and devises two efficient algorithms for the problem if there is abundant computing resource in MEC. Section 5 formulates an ILP solution first and then develops two efficient heuristics for the problem if computing resource of cloudlets is capacitated. Section 6 evaluates the performance of the proposed algorithms empirically, and Section 7 concludes the paper.

2 Related work

As a key enabling technology of 5G, mobile edge-cloud networks have gained tremendous attention from the research community recently [21]. Also, with the emergence of complicated and resource-hungry mobile applications, offloading user tasks to cloudlets of a nearby mobile edge-cloud network is becoming an important approach to reduce mobile device energy consumption and improve mobile user experience.

There are extensive studies on user request processing in mobile edge computing networks [4], [5], [12], [14], [15], [16], [24], [25]. Although MEC has received significant attention, most existing studies focused on minimizing the maximum/average delay, maximizing the throughput, or the linear combination of multiple performance metrics such as delays, energy consumptions and throughput. For example, Fan *et al.* [8] studied an application aware workload allocation problem in MEC with the aim to minimize the latency of IoT applications by jointly considering computing resource allocations and request assignments. Their solution takes into account both the communication delay and cloudlet processing delay, while considering workload dynamics in cloudlets. They also assumed each IoT application to be handled by a dedicated virtual machine in a cloudlet. They [9] also explored the cloudlet placement problem with the aim to minimize the cloudlet deployment cost. They proposed a Lagrangian heuristic to achieve a suboptimal solution, which considered both the cloudlet placement cost and average end-to-end delay cost. They further designed a resource allocation scheme to minimize the end-to-end delays between users and their cloudlets after cloudlets being deployed in the network. Alameddine *et al.* [1] investigated joint resource allocation and request scheduling problem for delay-sensitive IoT services. They formulated the problem of concern as a Mixed Integer Program (MIP) when the problem size is small. Otherwise, they devised a heuristic solution to decompose the problem into two stages, performing resource allocation in the first stage, and addressing task offloading for each single IoT application in the second stage. Ren *et al.* [23] considered a task offloading problem with the aim to minimize the overall task computation time in MEC networks. They proposed a distributed algorithm based on the game theory approach to split tasks and offload appropriate percentage of tasks to the MEC cloudlets. Hu *et al.* [13] studied the joint resource allocation and request offloading problem in MEC networks with the aim to

minimize user request response delay. They divided the problem of concern into two sub-problems, and analyzed the problem as a double decision-making problem and then formulated a mixed integer nonlinear program for it by applying the quasi-convex technique based on game theory.

Chen *et al.* [3] investigated the problem of task offloading for mobile edge computing in a software-defined ultra-dense network, with an aim to minimize the total execution duration of all tasks. However, they assumed that edge clouds are deployed at each base station, and did not deal with the VNF placement in the paper. Feng *et al.* [10] proposed an algorithm with performance guarantees for placing VNF in distributed cloud networks and routing service flows among the placed VNFs under the constraints of service function chains of the requests. Chen *et al.* [4] investigated the problem of multi-user computation offloading for mobile edge clouds in a multi-channel wireless interference environment, which is formulated as a game among multiple users and a mechanism with a Nash equilibrium is proposed. Their mechanism however may not be applicable to a mobile edge-cloud computing network located between wireless access points and the core network.

The above researches considered VNF instance provisioning in which the VNF instances requested by each request is dedicated to its user only. 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 [12], [15]. For example, Jia *et al.* [15] investigated a task offloading problem in a wireless metropolitan area network. They aimed to maximize the throughput while minimizing the admission cost of requests. They developed an effective prediction mechanism to instantiate and release VNF instances of network functions for cost savings. He *et al.* [12] 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. They proposed an approximation algorithm for a special case of their problem where both services and cloudlets are homogeneous. For the general case of their problem, they devised a greedy heuristic based on linear program relaxation and rounding. 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. Ma *et al.* [18], [19] studied NFV-enabled multicasting in an MEC network subject to resource capacities on both its cloudlets and links. They devised an approximation algorithm for the cost minimization problem of admitting a single NFV-enabled multicast request by reducing it to a directed multicast Steiner Tree problem and exploring VNF instance sharing. They also proposed an online algorithm with a provable competitive ratio for the online throughput maximization problem.

In this study, we jointly consider VNF instance deployment and request assignment while meeting resource capacity constraints. In contrast with the above studies, to the best of our knowledge, we are the very first to explore a brand new research topic in MEC that is the non-trivial usage tradeoff between different types of resources for the user request admissions in MEC to minimize their admission cost. That is, we use inexpensive resource to replace expensive resource for request admissions. This tradeoff purpose can be realized through introducing a novel concept that is the load factor concept at VNF instances.

3 PRELIMINARIES

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

3.1 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, cafe shops, shopping malls, public libraries, bus stations, train stations, and hospitals. There is a cloudlet with computing capacity C_v that is 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. For simplicity, an AP node and the area covered by it will be used interchangeably if no confusion arises. In case a mobile user in an overlapping region among multiple APs, we assume that the mobile user will connect to its nearest AP or the AP with the strongest signal strength. Figure 1 is an example of an MEC network.

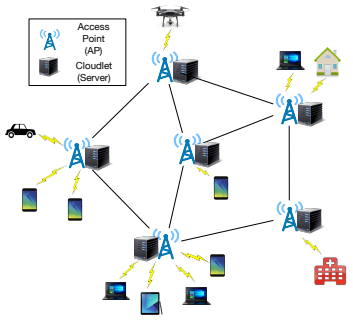


Fig. 1. An illustrative example of an MEC network consisting of 6 APs with each co-located with a cloudlet.

3.2 User requests with VNF service requirements

We consider the provisioning of virtualized network function services to mobile users. Let $\mathcal{F} = \{f^{(1)}, f^{(2)}, \ldots, f^{(K)}\}$ be the set of network functions provided by the network service provider. Assume that there is a set U of users accessing the MEC network through their nearby APs. Each user $j \in U$ issues a user request r_j that demands for a specified VNF service, and r_j is represented by a tuple $r_j = \langle v_j, f_j^{(k)}, \rho_j \rangle$, where v_j is the AP location of the request, $f_j^{(k)}$ is the VNF service requested by the request, and ρ_j is the demanded data packet rate of the request. 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 $U_{k,v} \subseteq U_k$ and $U_k \subseteq U$ for each k with $1 \leq k \leq K$.

We assume that computing resource of cloudlets in the MEC is virtualized. 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 virtualized network functions 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 $v \in V$ that is not necessarily co-located with its user at AP v_j, i.e., $v_j \neq v$. 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.

3.3 Admission cost of a request

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 usage cost of computing resource $\rho_j \cdot c_{proc}(f^{(k)}, v)$, where $c_{proc}(f^{(k)}, v)$ is the cost of processing a packet by a VNF instance $f^{(k)}$ at cloudlet v and ρ_j is the packet rate of r_j. Notice that the processing cost of a data packet $c_{proc}(f^{(k)}, v)$ of different VNF instances at different cloudlets may be significantly different, since different amounts of computing resource is consumed by different VNF instances. 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 $P(v_j, v)$ be the shortest routing path for request r_j between its AP node v_j and cloudlet v in which its data packet traffic is processed by a VNF instance of $f_j^{(k)}$. The routing cost of a packet along path $P(v_j, v)$ thus is $c_{bw}(P(v_j, v)) = \sum_{e \in P(v_j,v)} c_e$, where c_e is the unit transmission cost on each link $e \in E$. Notice that if the data packet traffic of request r_j is processed by a VNF instance in the cloudlet attached to its AP v_j, the communication cost of the request is negligible.

3.4 The knapsack and generalized assignment problems

We here introduce the knapsack problem and the generalized assignment problem as they will be used in the rest of this paper, and defined as follows.

Given a set $\mathcal{I}$ of elements and a bin B with capacity $cap(B)$ where each element $i \in \mathcal{I}$ has a size $size(a_i)$ and a profit $profit(a_i)$ if element $a_i \in \mathcal{I}$ is placed into bin B, *the knapsack problem* is to place as many as elements of $\mathcal{I}$ into B such that the sum of profits of the placed elements is maximized, subject to the capacity of bin B. There is an approximation algorithm for the knapsack problem with an approximation ratio of $\frac{1}{1+\epsilon}$ [7], which takes $O(\frac{|\mathcal{I}| \log |\mathcal{I}|}{\epsilon})$ time, where ϵ is a constant with $0 < \epsilon \leq 1$.

Given a set $\mathcal{I}$ of elements and a set $\mathcal{B}$ of bins where each bin $b_j \in \mathcal{B}$ has computing capacity of $cap(b_j)$, and each element $a_i \in \mathcal{I}$ has a size $size(a_i, b_j)$ and a profit $profit(a_i, b_j)$ if element a_i is placed into bin b_j, *the Generalized Assignment Problem* (GAP) is to place as many as elements in $\mathcal{I}$ into the bins in $\mathcal{B}$ such that the profit sum of all placed elements is maximized, subject to the capacity on each bin in $\mathcal{B}$. There is an approximation algorithm for the GAP with an approximation ratio of $\frac{1}{2+\epsilon}$ due to Cohen *et al.* [6], which takes $O(\frac{|\mathcal{I}| \cdot |\mathcal{B}|}{\epsilon} + \frac{|\mathcal{B}|}{\epsilon^4})$ time, where ϵ is a constant with $0 < \epsilon \leq 1$.

3.5 Problem definitions

To admit a user request, we may utilize an existing VNF instance or instantiate a new one for the processing of the data packet traffic of the request. The admission of a request not only consumes the computing resource but also the communication (bandwidth) resource of the network, and there is a non-trivial tradeoff between the usages of these two types of resources. For example, if the computing resource in MEC is more expensive, we may instantiate less numbers of VNF instances for the admissions of user requests. Otherwise (if the communication resource in MEC is more expensive), we may instantiate more VNF instances at cloudlets to reduce the communication resource consumption of admitted requests.

In this paper, we consider two NFV-enabled request admission problems in MEC: one is *the cost minimization problem* if there is sufficient computing resource in cloudlets to meet all user requests' resource demands; and another is *the throughput maximization problem* that aims to maximize the number of user request admissions while minimizing their admission cost, subject to the computing capacity constraint at each cloudlet. The precise definitions of these two problems are given as follows.

Definition 1: Given an MEC network $G = (V, E)$ 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 NFV-enabled request $r_j = \langle v_j, f_j^{(k)}, \rho_j \rangle$, where v_j is the AP location of the request, $f_j^{(k)}$ is its requested network function, and ρ_j is its data packet rate if the request is admitted, *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 is sufficient computing resource at cloudlets for the admissions of all requests.

In Definition 1, we assumed that the computing resource in each cloudlet is unlimited. However, in practice, the computing resource in each cloudlet is capacitated. This implies that not all requests can be admitted due to lack of computing resource in the network. We instead aim to maximize the number of requests admitted while minimizing their admission cost, subject to the computing capacity at each cloudlet in G, by presenting the following problem definition.

Definition 2: Given an MEC network $G = (V, E)$ with a set V of APs, a cloudlet $v \in V$ with computing capacity C_v

is co-located with each AP $v \in V$, a set U of users with each user $j \in U$ having a request $r_j = \langle v_j, f_j^{(k)}, \rho_j \rangle$, *the throughput maximization problem* is to maximize the number of requests admitted by deploying VNF instances in the cloudlets while minimizing the admission cost of the admitted requests, subject to the computing capacity on each cloudlet in G.

3.6 NP hardness of the defined problems

In the following we show that the two defined optimization problems are NP-hard.

Theorem 1. *The cost minimization problem in $G = (V, E)$ is NP-hard.*

Proof. We prove the NP hardness of the cost minimization problem by a reduction from a well known NP-hard problem - *the bin packing problem* that is defined as follows. Given a set $\mathcal{I}$ of items with each item $a_i \in \mathcal{I}$ having a specified size $size(a_i)$, the problem is to find a packing schedule for the items into bins each with capacity B that minimizes the number of bins used, assuming that $B \geq \max_{a_i \in \mathcal{I}} size(a_i)$.

We show that an instance of the bin packing problem can be reduced to an instance of the cost minimization problem. Specifically, each bin corresponds to a VNF instance with processing capacity B, a set $\mathcal{I} = \{a_i \mid 1 \leq i \leq n\}$ of items corresponds to a set of requests to be admitted and processed by VNF instances in cloudlets. Each request a_i has a packet rate $size(a_i)$. Here we assume only one type of VNF instances is considered and only the cost of the VNF instance instantiation is taken into account while the processing and routing costs of the packets of each request are ignored, and the aim is to minimize the number of VNF instances deployed to minimize the admission cost of admitted requests. It can be seen that a solution to this special case of the cost minimization problem is a solution to the bin packing problem. Thus, the theorem holds. □

Theorem 2. *The throughput maximization problem in $G = (V, E)$ is NP-hard.*

Proof. We prove the NP hardness of the throughput maximization problem by a reduction from a well known NP-hard problem - the Generalized Assignment Problem (GAP). We show that an instance of the generalized assignment problem can be reduced to an instance of the throughput maximization problem. Specifically, each bin $b_j \in \mathcal{B}$ corresponds to a cloudlet with capacity $cap(b_j)$, a set $\mathcal{I} = \{a_i\}$ of items corresponds to a set of requests to be admitted and processed by their VNF instances at a cloudlet. Each request a_i requires a VNF instance with specified computing resource $C(f_i)$ which is the size $size(a_i, b_j)$ of each item a_i in a bin b_j, and the profit $profit(a_i, b_j)$ received is 1 if it is admitted. Here we assume a VNF instance can only admit one request of its type. We aim to admit a subset of requests by creating VNF instances for them assuming there is no existing VNF instances deployed for them, so that the number of user requests been admitted is maximized, while the size of each cloudlet is bounded by its capacity $cap(b_j)$. It can be seen that a solution to this special profit maximization problem is a solution to the generalized assignment problem. Thus, this theorem holds. □

4 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) problem. We then devise an efficient algorithm for it, and we finally analyze the time complexity of the proposed algorithm.

4.1 An integer linear program solution

We start with formulating the problem as an integer linear program (ILP). We then determine the value range of $N^{(k)}$ which is the number of VNF instances of $f^{(k)}$ needed to admit all requests in U_k for each k with $1 \leq k \leq K$. 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, clearly $U_k \subseteq U$ and $U_{k,v} \subseteq U_k$.

Notice that for the sake of convenience, we here assume that each request r_j is located at AP v_j. In fact, it can be located at another AP v_l with $l \neq j$, this does not affect the applicability of the proposed algorithms in this paper.

Recall that $\mathcal{F} = \{f^{(1)}, f^{(2)}, \ldots, f^{(K)}\}$ is a set of network functions provided by the network. 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, in other words, the data packet traffic of all requests in U_k can be processed by these $N^{(k)}$ VNF instances. Let $\mathcal{I}^{(k)} = \{I_1^{(k)}, I_2^{(k)}, \ldots I_{N^{(k)}}^{(k)}\}$ 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 $\mathcal{I} = \cup_{k=1}^{K} \mathcal{I}^{(k)}$.

Define a function $\phi : \mathcal{F} \times \mathcal{I} \mapsto V$. Specifically, the variable domain of $\phi(\cdot)$ consists of K disjoint subdomains, i.e., $\phi : \mathcal{F} \times \mathcal{I}^{(k)} \mapsto V$ 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 $\phi(k, i)$. Notice that different VNF instances in $\mathcal{I}^{(k)}$ can be mapped to a single or multiple cloudlets.

Define another function $\psi : \mathcal{F} \times U \mapsto \mathcal{I}$. Specifically, the variable domain of $\psi(\cdot)$ consists of K disjoint subdomains, i.e., $\psi : \mathcal{F} \times U_k \mapsto \mathcal{I}^{(k)}$ 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 $\phi(k, \psi(k, j))$.

If all VNF instances of $f^{(k)}$ have been instantiated in cloudlets, *the processing and routing cost* of admitting a user request $r_j = \langle v_j, f_j^{(k)}, \rho_j \rangle$ is the sum of the routing cost of its data packet traffic $\rho_j \cdot c_{bw}(P(v_j, v_{\phi(k,\psi(k,j))})$ at the $\psi(k, j)$th VNF instance of $f^{(k)}$ that is accommodated at cloudlet $v_{\phi(k,\psi(k,j))}$ and the processing cost of its data packet traffic $c_{proc}(f^{(k)}, v_{\phi(k,\psi(k,j))})$ at the VNF instance, i.e., $\rho_j \cdot (c_{bw}(P(v_j, v_{\phi(k,\psi(k,j))})) + c_{proc}(f^{(k)}, v_{\phi(k,\psi(k,j))}))$.

The cost minimization problem is to admit all requests while minimizing their admission cost, that is, the optimization objective is to minimize

$$\sum_{k=1}^{K} \sum_{i=1}^{N^{(k)}} c_{ins}(f^{(k)}, v_{\phi(k,i)}) +$$

$$\sum_{k=1}^{K} \sum_{i=1}^{N^{(k)}} \sum_{r_j \in U_k} \rho_j \cdot c_{bw}(P(v_j, v_{\phi(\psi(k,j))})) +$$

$$\sum_{k=1}^{K}\sum_{i=1}^{N^{(k)}}\sum_{r_j\in U_k}\rho_j\cdot c_{proc}(f^{(k)},v_{\phi(k,\psi(k,j))}), \quad (1)$$

and the number of VNF instances $N^{(k)}$ of $f^{(k)}$ for all requests in U_k must meet

$$\max\{\lceil\frac{\sum_{r_j\in U_k}\rho_j}{\lambda\cdot\mu^{(k)}}\rceil, n_B(\lambda\cdot\mu^{(k)},U_k)\}\le N^{(k)}$$
$$\le\min\{\sum_{v\in V}n_B(\lambda\cdot\mu^{(k)},U_{k,v}),|U_k|\}, \quad (2)$$

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 computing capacity of $\lambda\cdot\mu$ and λ is a given load factor within $0<\lambda\le 1$, and there is a set U' of elements with each element r_j having a profit 1 and weight ρ_j.

The correctness of Ineq. (2) will be shown in Lemma 1 in Section 4.

In the optimization objective (1), the first term is the cost of instantiating all VNF instances to meet computing demands of all requests. The second term is the routing cost of data packet traffic of all requests between the location v_j of each request r_j and the location $v_{\phi(k,\psi(k,j))}$ of its VNF instance, and the third term is the processing cost of data packets of all requests at different VNF instances in different cloudlets.

The solution of the cost minimization problem thus consists of (i) the value of $N^{(k)}$; (ii) the deployment of the $N^{(k)}$ VNF instances at different cloudlets (the mapping function $\phi(\cdot)$); and (iii) the assignment of each request in U_k to one of the deployed $N^{(k)}$ VNF instances (the mapping function $\psi(\cdot)$) as well as its cloudlet indexed at $\phi(k,\psi(\cdot))$ such that the optimization objective (1) is minimized, for each k with $1\le k\le K$.

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 $1\le k\le K$. For the ith VNF instance $I_i^{(k)}\in\mathcal{I}^{(k)}$ of $f^{(k)}$ and each cloudlet $v_l\in V$ with $1\le i\le N^{(k)}$, $1\le k\le K$, and $1\le l\le|V|$, there is a binary decision variable x_{ikl} where $x_{ikl}=1$ if the ith VNF instance $I_i^{(k)}$ of $f^{(k)}$ is deployed in cloudlet v_l. Furthermore, there is a binary decision variable y_{ijl}, 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) can be rewritten into an equivalent optimization objective (3) of the cost minimization problem is to minimize the admission cost of all requests in $G(V,E)$ as all requests will be admitted by the assumption, i.e.,

$$minimize\ \sum_{k=1}^{K}\sum_{i=1}^{N^{(k)}}\sum_{l=1}^{|V|}c_{ins}(f^{(k)},v_l)\cdot x_{ikl}+$$
$$\sum_{r_j\in U}\sum_{k=1}^{K}\sum_{i=1}^{N^{(k)}}\sum_{l=1}^{|V|}\rho_j a_j^k\cdot(c_{proc}(f^{(k)},v_l)+c_{bw}(P(v_j,v_l)))\cdot y_{ijl}, \quad (3)$$

subject to:

$$\sum_{i=1}^{N^{(k)}}\sum_{l=1}^{|V|}a_j^k\cdot y_{ijl}=1,\qquad \forall r_j\in U,\ 1\le k\le K \quad (4)$$

$$\sum_{r_j\in U}\rho_j\cdot a_j^k\cdot y_{ijl}\le\lambda\cdot\mu^{(k)}\cdot x_{ikl},$$
$$\forall 1\le l\le|V|,\ 1\le k\le K,\ 1\le i\le N^{(k)} \quad (5)$$

$$\max\{\lceil\frac{\sum_{r_j\in U_k}\rho_j}{\lambda\cdot\mu^{(k)}}\rceil, n_B(\lambda\cdot\mu^{(k)},U_k)\}\le N^{(k)}$$
$$\le\min\{\sum_{l=1}^{|V|}u(k,v_l),|U_k|\},\forall\,1\le k\le K \quad (6)$$

$$a_j^k\cdot y_{ijl}\le x_{ikl},\forall 1\le l\le|V|,\ r_j\in U,\ 1\le k\le K,$$
$$1\le i\le N^{(k)} \quad (7)$$

$$x_{ikl}\in\{0,1\},\forall 1\le l\le|V|,1\le k\le K,1\le i\le N^{(k)} \quad (8)$$

$$y_{ijl}\in\{0,1\},\forall 1\le l\le|V|,\ r_j\in U,\ 1\le i\le N^{(k)}. \quad (9)$$

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 [6] (see the proof body of Lemma 1).

Constraint (4) 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 (5) ensures that the accumulative packet rate of all requests for $f^{(k)}$ that are assigned to the ith VNF instance of $f^{(k)}$ in cloudlet v_l is no greater than its maximum packet processing rate $\lambda\cdot\mu^{(k)}$. When $\lambda=1$, the computing resource in MEC is the most expensive one and it should be fully utilized.

Constraint (6) ensures that all requests in U_k requesting for $f^{(k)}$ can be admitted and their data packet traffic can be processed by the $N^{(k)}$ VNF instances of type k for each k with $1\le k\le K$. Notice that the $N^{(k)}$ VNF instances can be deployed into one or multiple cloudlets in MEC.

Constraint (7) 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 that the value of $N^{(k)}$ is in a given range by the following lemma.

Lemma 1. *For each k with $1\le k\le K$, we have*

$$\max\{\lceil\frac{\sum_{r_j\in U_k}\rho_j}{\lambda\cdot\mu^{(k)}}\rceil, n_B(\lambda\cdot\mu^{(k)},U_k)\}\le N^{(k)}$$
$$\le\min\{\sum_{l=1}^{|V|}n_B(\lambda\cdot\mu^{(k)},U_{k,v_l}),|U_k|\}$$
$$\le\min\{\sum_{l=1}^{|V|}u(k,v_l),|U_k|\}. \quad (10)$$

Proof. Recall that $N^{(k)}$ is such a positive integer that the data packet traffic of all requests in U_k can be processed by these $N^{(k)}$ VNF instances of $f^{(k)}$. 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 the minimum number of bins is NP-hard. We instead adopt an approximation algorithm for the GAP [6] 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 $n_H^{(k)} = 2n_L$ 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. We then 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, there will not incur any routing cost of its admission. Thus, there must have enough VNF instances in that cloudlet for the admissions of the requests. 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 in cloudlet v for the admissions of these requests is equivalent to determining the minimum number of bins $n_B(\lambda \cdot \mu^{(k)}, U_{k,v})$ with bin capacity $\mu^{(k)}$ to pack their data packet rates. The range 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 the exact value is NP-hard. A naive upper bound $N_{upp}^{(k)}$ on $N^{(k)}$ is $\sum_{v \in V} |U_{k,v}| = |U_k|$, an improved upper bound is to apply an approximation algorithm for the GAP at each AP [6] 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, $N_{upp}^{(k)} = \sum_{v \in V} u(k, v)$. □

As mentioned in the beginning of this paper, we aim at exploring a non-trivial usage tradeoff between computing resource (in cloudlets) and communication resource (at links) for request admissions. For a given type k network function $f^{(k)}$ with $1 \leq k \leq K$, in order to admit all requests in U_k, we note that with the increase on the number $N^{(k)}$ $(= |\mathcal{I}^{(k)}|)$ of VNF instances of $f^{(k)}$, the associated computing cost (including the instantiation cost of VNF instances and the processing cost of data packet traffic of requests at the VNF instances) increases, too. Contrarily, the routing cost of data packet traffic of admitted requests decreases or vice versa. 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 to the problem can be obtained, by exploring every possible integer value in the interval of Ineq. (10), and then choosing the one with the minimum cost as the problem solution.

4.2 Heuristic algorithm by deploying VNF instances one by one

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. We will use the ILP solution as a baseline for the performance evaluation of our proposed algorithm. In the following we propose an efficient algorithm for the cost minimization problem.

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

$$cost(r_j) = \rho_j \cdot (c_{proc}(f^{(k)}, v) + c_{bw}(P(v_j, v))) \qquad (11)$$

the processing and routing cost of admitting request r_j. To this end, we treat $I_i^{(k)}$ as a bin with capacity $\lambda \cdot \mu^{(k)}$, 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 of a newly admitted request, assuming that instance $I_i^{(k)}$ has been instantiated in cloudlet $v \in V$ already. A solution $R_v^{(k)}$ $(\subseteq U_k')$ to this knapsack problem is obtained by applying an approximation algorithm for knapsack problems.

Notice that the profit maximization in this knapsack problem is roughly equivalent to the cost minimization of admitting all requests in $R_v^{(k)}$. Furthermore, the instantiation cost of $I_i^{(k)}$ in cloudlet v has not been taken into account.

The rest is to choose a cloudlet v_0 hosting $I_i^{(k)}$ if the ratio of the number of newly admitted requests to the admission cost of the requests in $R_v^{(k)}$ is maximized, i.e.,

$$v_0 = \underset{v \in V}{\operatorname{argmax}} \frac{|R_v^{(k)}|}{c_{ins}(f^{(k)}, v) + \sum_{r_j \in R_v^{(k)}} cost(r_j)}, \quad (12)$$

where the maximum ratio in Eq. (12) implies that deploying the ith VNF instance $I_i^{(k)}$ in cloudlet v_0 results in the minimization of the admission cost of newly admitted requests $R_v^{(k)}$ in iteration i for network function $f^{(k)}$ with $1 \leq i \leq N^{(k)}$ and $1 \leq k \leq K$.

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

Algorithm 1 Deploying VNF instances one by one for the cost minimization problem

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

Output: A solution to minimize the admission cost of all requests, by deploying VNF instances one by one in cloudlets to admit all user requests.

$cost \leftarrow 0$; /* the cost of all request admissions */
$S \leftarrow \emptyset$; /* different VNF instance placements in cloudlets, i.e., function $\phi(\cdot)$ */
$A \leftarrow \emptyset$; /* different user request assignment to different VNF instances at different cloudlets, i.e., function $\psi(\cdot)$ */
for $k \leftarrow 1$ to K **do**
  $i \leftarrow 0$; /* the ith VNF instance $I_i^{(k)}$ of $f^{(k)}$ to be deployed in a cloudlet */
  $U'_k \leftarrow U_k$; /* the set of unassigned requests in U_k */
  $\mathcal{I}^{(k)} \leftarrow \emptyset$; /* the set of VNF instances of $f^{(k)}$ */
  while $U'_k \neq \emptyset$ **do**
    $i \leftarrow i + 1$;
    Compute the ratio in formula (12) if the ith VNF instance $I_i^{(k)}$ of $f^{(k)}$ is deployed in cloudlet v for each $v \in V$ by invoking an approximation algorithm for the knapsack problem with capacity $\lambda \cdot \mu^{(k)}$ and set U'_k, and let cloudlet v_0 be chosen to the host of the ith VNF instance $I_i^{(k)}$;
    $\mathcal{I}^{(k)} \leftarrow \mathcal{I}^{(k)} \cup \{I_i^{(k)}\}$;
    $U'_k \leftarrow U'_k \setminus R_v^{(k)}$; /* all requests in $R_v^{(k)}$ now have been admitted and assigned to VNF instance $I_i^{(k)}$ in cloudlet v_0 */
    $cost \leftarrow cost + c_{ins}(f^{(k)}, v_0) + \sum_{r_j \in R_v^{(k)}} \rho_j \cdot (c_{proc}(f^{(k)}, v_0) + c_{bw}(P(v_j, v_0)))$;
    $S \leftarrow S \cup \{I_i^{(k)}$ in cloudlet $v_0\}$; /* i.e., $\phi(k, i) = v_0$ */
    $A \leftarrow A \cup \{r_j$ is assigned to $I_i^{(k)}$ in cloudlet $v_0 \mid r_j \in R_v^{(k)}\}$; /* i.e., $\psi(k, j) = i$ and $\phi(k, \psi(k, j)) = \psi(k, i) = v_0$ */
return $cost$, S, and A.

4.3 An improved algorithm by deploying multiple VNF instances simultaneously

In `Algorithm` 1, only one VNF instance $I_i^{(k)}$ is deployed in a cloudlet $v \in V$ within iteration i, $1 \leq i \leq N^{(k)}$. Instead of deploying VNF instances one by one, we explore deploying multiple (e.g., α) VNF instances together, where α is a positive integer, i.e., $\alpha = 1, 2, 3$, etc. When $\alpha = 1$, one VNF instance is deployed in each iteration, and this is `Algorithm` 1. The rationale for deploying a group of α VNF instances is the fact that a single request can be admitted by multiple VNF instances, and its admission will occupy certain amounts of resources that will affect future request admissions. If multiple VNF instances are deployed at the same time, a better combination of requests can be admitted with the minimum admission cost.

Denote by $\mathcal{I}_i^{(k)} = \{I_{i,1}^{(k)}, I_{i,2}^{(k)}, \ldots, I_{i,\alpha}^{(k)}\}$ a group of α VNF instances of type k in iteration i. We treat each VNF instance $I_{i,a}^{(k)}$ as a bin with capacity $\lambda \cdot \mu^{(k)}$ and has been instantiated in a cloudlet $v \in V$ already, and each request $r_j \in U'_k$ as an element with size ρ_j, and a profit $profit(r_j) = \frac{1}{cost(r_j)}$ is collected if it is admitted. A solution to this generalized assignment problem (GAP) can be obtained by invoking an approximation algorithm due to Cohen *et al.* [6]. Then, we choose a group of α cloudlets to host the set $\mathcal{I}_i^{(k)}$ of VNF instances. The group of cloudlets are not necessarily distinct, which means multiple VNF instances in $\mathcal{I}_i^{(k)}$ can reside in the same cloudlet.

Denote by $\mathbb{S}^\alpha$ the collection of all subsets of cloudlets of size α, that is $\mathbb{S}^\alpha = \{\mathcal{S}\}$, and $\mathcal{S} = \{v \mid v \in V, |\mathcal{S}| = \alpha\}$. Thus, $|\mathbb{S}^\alpha| = |V|^\alpha$. A set $\mathcal{S}_0$ of cloudlets (not necessarily distinct) of size α is selected if the ratio of the number of newly admitted requests $R_{\mathcal{S}}^{(k)}$ to their admission cost is maximized, i.e.,

$$\mathcal{S}_0 = \underset{\mathcal{S} \in \mathbb{S}^\alpha}{\operatorname{argmax}} \frac{|R_{\mathcal{S}}^{(k)}|}{\sum_{v \in \mathcal{S}} c_{ins}(f^{(k)}, v) + \sum_{r_j \in R_{\mathcal{S}}^{(k)}} cost(r_j)}. \quad (13)$$

This procedure continues until all requests can be admitted. The detailed algorithm is given in `Algorithm` 2.

4.4 Analysis on the proposed algorithms

In the following we analyze the time complexities of `Algorithm` 1 and `Algorithm` 2.

Theorem 3. *Given an MEC network $G = (V, E)$, and a set U of users with each having a user request $r_j = \langle v_j, f_j^{(k)}, \rho_j \rangle$, assuming that there is sufficient computing resource in cloudlets to accommodate all VNF instances of different network functions, there are algorithms,* `Algorithm` *1 and* `Algorithm` *2, for the cost minimization problem, which take $O(|V| \cdot \frac{|U| \log |U|}{\epsilon} + |V|^3)$ time and $O(|V|^\alpha(|U| \log \frac{1}{\epsilon} + \frac{1}{\epsilon^4}) + |V|^3)$ time respectively, and deliver feasible solutions to the problem, where $\alpha \geq 1$ is a positive integer and ϵ is a constant with $0 < \epsilon \leq 1$.*

Proof. As there are K iterations of `Algorithm` 1 and `Algorithm` 2. 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 solutions delivered by `Algorithm` 1 and `Algorithm` 2 thus are feasible.

We then analyze the time complexity of `Algorithm` 1. The algorithm contains K iterations. Within iteration k with

Algorithm 2 Deploying VNF instances with a look-ahead strategy for the cost minimization problem

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

Output: A solution to minimize the admission cost of all requests, by deploying a group of VNF instances together in each iteration to admit all user requests.

1: $cost \leftarrow 0$; /* the cost of all request admissions */
2: $S \leftarrow \emptyset$; /* different VNF instance placements in cloudlets, i.e., function $\phi(\cdot)$ */
3: $A \leftarrow \emptyset$; /* different user request assignment to different VNF instances at different cloudlets, i.e., function $\psi(\cdot)$ */
4: **for** $k \leftarrow 1$ to K **do**
5: $i \leftarrow 0$; /* the ith set of VNF instances $\mathcal{I}_i^{(k)}$ of $f^{(k)}$ to be deployed in cloudlets */
6: $U_k' \leftarrow U_k$; /* the set of unassigned requests in U_k */
7: $\mathcal{I}^{(k)} \leftarrow \emptyset$; /* 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 (13) if the set $\mathcal{I}_i^{(k)}$ of VNF instances of $f^{(k)}$ deployed in cloudlets $\mathcal{S}$ by invoking an approximation algorithm for the GAP problem with α bins each with capacity $\lambda \cdot \mu^{(k)}$, and set of elements U_k', let cloudlets $\mathcal{S}_0$ be chosen to the host of the set of VNF instance $\mathcal{I}_i^{(k)}$;
11: $\mathcal{I}^{(k)} \leftarrow \mathcal{I}^{(k)} \cup \mathcal{I}_i^{(k)}$;
12: $U_k' \leftarrow U_k' \setminus R_{\mathcal{S}}^{(k)}$; /* all requests in $R_{\mathcal{S}}^{(k)}$ now have been admitted and assigned to VNF instances $\mathcal{I}_i^{(k)}$ to be deployed in a set $\mathcal{S}_0$ of cloudlets */
13: $cost \leftarrow cost + \sum_{v \in \mathcal{S}_0} c_{ins}(f^{(k)}, v) + \sum_{r_j \in R_{\mathcal{S}_0}^{(k)}} cost(r_j)$;
14: $S \leftarrow S \cup \{I_{i,a}^{(k)}$ in cloudlet $v \mid I_{i,a}^{(k)} \in \mathcal{I}_i^{(k)}, v \in \mathcal{S}_0\}$;
15: $A \leftarrow A \cup \{r_j$ is assigned to $I_{i,a}^{(k)}$ in cloudlet $v \mid r_j \in R_{\mathcal{S}_0}^{(k)}, I_{i,a}^{(k)} \in \mathcal{I}_i^{(k)}, v \in \mathcal{S}_0\}$;
16: **return** $cost$, S, and A.

$1 \leq k \leq K$, there are $N^{(k)}$ VNF instances of $f^{(k)}$ to be deployed and they are deployed one by one, where $N^{(k)} \leq |U_k|$. Determining the deployment of the ith VNF instance $I_i^{(k)}$ of $f^{(k)}$ on which cloudlet and the assignment of requests in U_k' to $I_i^{(k)}$ take $O(|V| \cdot \frac{|U_k'| \log |U_k'|}{\epsilon})$ time, by invoking an approximation algorithm for the knapsack problem, assuming that all pairs of shortest paths between the AP location of each request and the VNF instance assigned to it for its data packet traffic processing is pre-calculated, which takes $O(|V|^3)$ time. `Algorithm` 1 thus takes $O(|V|^3 + |V| \cdot \sum_{k=1}^{K} \frac{|U_k'| \log |U_k'|}{\epsilon}) = O(|V| \cdot \frac{|U| \log |U|}{\epsilon} + |V|^3)$ time, where $U_k' \subseteq U_k$ for each k with $1 \leq k \leq K$ and ϵ is constant with $0 < \epsilon \leq 1$.

The rest is to analyze the time complexity of `Algorithm` 2. In each iteration, determining the deployment of a group $\mathcal{I}_i^{(k)}$ of VNF instances of $f^{(k)}$ on which cloudlets and the assignment of requests in U_k' to $\mathcal{I}_i^{(k)}$ take $O(\alpha |V|^\alpha (|U_k'| \log \frac{1}{\epsilon} + \frac{1}{\epsilon^4}))$ time, by invoking an approximation algorithm for the GAP problem, since $|V|^\alpha$ combinations of VNF instances are examined. `Algorithm` 2 thus takes $O(|V|^\alpha (|U| \log \frac{1}{\epsilon} + \frac{1}{\epsilon^4}) + |V|^3)$ time and α is a positive integer and ϵ is constant with $0 < \epsilon \leq 1$. □

5 Algorithm for the Throughput Maximization Problem

The cost minimization problem assumed that computing resource at cloudlets is abundant. In practice, it is not uncommon that the computing resource in cloudlets is capacitated, i.e., the computing resource at each cloudlet $v \in V$ is capacitated by C_v. This implies that not all requests in U can be admitted due to lack of computing resource to meet their resource demands. In this section, we study *the throughput maximization problem* with the aim to maximize the throughput, by admitting as many as requests while minimizing their admission cost, subject to computing capacity on each cloudlet in MEC.

5.1 Integer linear program formulation

We formulate an ILP solution to the throughput maximization problem. The difference from the previous one is the objective function and the capacity constraint on each cloudlet.

$$maximize \sum_{k=1}^{K} \sum_{j \in U} \sum_{i=1}^{N^{(k)}} \sum_{l=1}^{|V|} a_j^k \cdot y_{ijl}, \tag{14}$$

subject to:

$$\sum_{k=1}^{K} \sum_{i=1}^{N^{(k)}} C(f^{(k)}) \cdot x_{ikl} \leq C_v, \forall 1 \leq l \leq |V| \tag{15}$$

$$\sum_{i=1}^{N^{(k)}} \sum_{l=1}^{|V|} a_j^k \cdot y_{ijl} \leq 1, \forall j \in U,\ 1 \leq k \leq K \tag{16}$$

$$0 \leq N^{(k)} \leq \min\{\sum_{l=1}^{|V|} u(k, v_l), |U_k|\}, \forall\ 1 \leq k \leq K \tag{17}$$

$$a_j^k \cdot y_{ijl} \leq x_{ikl}, \quad \forall 1 \leq l \leq |V|,\ j \in U,\ 1 \leq k \leq K,\ 1 \leq i \leq N^{(k)} \tag{18}$$

$$\sum_{j \in U} \rho_j \cdot a_j^k \cdot y_{ijl} \leq \mu^{(k)} \cdot x_{ikl}, \quad \forall 1 \leq l \leq |V|,\ 1 \leq k \leq K,\ 1 \leq i \leq N^{(k)} \tag{19}$$

$$x_{ikl} \in \{0, 1\}, \forall 1 \leq l \leq |V|,\ 1 \leq k \leq K,\ 1 \leq i \leq N^{(k)} \tag{20}$$

$$y_{ijl} \in \{0, 1\},\ \forall 1 \leq l \leq |V|,\ j \in U,\ 1 \leq i \leq N^{(k)} \tag{21}$$

Constraint (15) ensures that computing capacity constraints of all cloudlets are not violated.

5.2 Deploying VNF instances one by one

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_0 is chosen if its ratio 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_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_0 for the new

VNF instance deployment are determined by the following formula.

$$\langle k_0, v_0\rangle = \underset{1\le k\le K, v\in V}{\operatorname{argmax}} \eta_{k,v} = \underset{1\le k\le K, v\in V}{\operatorname{argmax}} \frac{|R_v^{(k)}|}{c_{ins}(f^{(k)}, v) + \sum_{r_j\in R_v^{(k)}} cost(r_j)}, \quad (22)$$

where $R_v^{(k)}$ 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 for each k with $1 \le k \le K$. Notice that identifying the set $R_v^{(k)}$ 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 size ρ_j and a profit $profit(r_j) = \frac{1}{cost(r_j)} = \frac{1}{\rho_j\cdot(c_{proc}(f^{(k)},v)+c_{bw}(P(v_j,v)))}$.

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_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` 3.

5.3 Deploying VNF instances with a look-ahead strategy

Similarly, we can apply a look-ahead strategy to deploy a group of VNF instances in each iteration. Specifically, in each iteration, a set of α VNF instances of type k are deployed in a group $\mathcal{S}$ of cloudlets (not necessarily distinct), where $\mathcal{S} = \{v \mid v \in V, |\mathcal{S}| = \alpha\}$, if each cloudlet $v \in \mathcal{S}$ has sufficient residual computing resource to accommodate the VNF instance of $f^{(k)}$ and the ratio of the number of newly admitted requests $R_{\mathcal{S}}^{(k)}$ that request for $f^{(k)}$ to their admission cost is the maximum one. The network function $f^{(k_0)}$ and cloudlet set $\mathcal{S}_0$ are determined by the following formula.

$$\langle k_0, \mathcal{S}_0\rangle = \underset{1\le k\le K, \mathcal{S}\in\mathbb{S}^\alpha}{\operatorname{argmax}} \frac{|R_{\mathcal{S}}^{(k)}|}{\sum_{v\in\mathcal{S}} c_{ins}(f^{(k)}, v) + \sum_{r_j\in R_{\mathcal{S}}^{(k)}} cost(r_j)}, \quad (23)$$

where $\mathbb{S}^\alpha$ denotes the combinations of all set of cloudlets of size α, $R_{\mathcal{S}}^{(k)}$ is the set of requests of $f^{(k)}$ that have not been admitted in previous iterations and to be admitted and processed by the set of α VNF instances in the following iteration. An approximation algorithm for the GAP problem due to Cohen *et al.* [6] is invoked to solve the GAP problem with a set of α bins each with bin capacity $\lambda \cdot \mu^{(k)}$ and a set of elements in $U'_k \subseteq U_k$, where each element $r_j \in U'_k$ has size ρ_j and a profit of $profit(r_j) = \frac{1}{cost(r_j)}$.

The above procedure continues until all requests in U are admitted or no more VNF instances can be instantiated in any cloudlet without violating its computing capacity. The algorithm is termed as `Algorithm` 4, and can be devised through performing modifications to `Algorithm` 3. Specifically, a group of α VNF instances of network function

Algorithm 3 Deploying VNF instances one by one for the throughput maximization problem

Input: Given an MEC $G = (V, E)$, there is a cloudlet $v \in V$ with capacity C_v co-located with each AP $v \in V$, a set of users U with each having a user request $r_j = \langle v_j, f_j^{(k)}, \rho_j\rangle$, and a given load factor λ with $0 < \lambda \le 1$.

Output: A solution that maximizes the throughput while minimizing the admission cost of admitted requests, by deploying VNF instances one by one in cloudlets and assigning the requests to the deployed VNF instances.

$flag \leftarrow true$;
$cost \leftarrow 0$; /* the admission cost of all admitted requests */
$S \leftarrow \emptyset$; /* different VNF instance placements in cloudlets, i.e., function $\phi(\cdot)$ */
$A \leftarrow \emptyset$; /* all admitted requests so far, and different user request assignment to different VNF instances at different cloudlets, i.e., function $\psi(\cdot)$ */
while $flag$ **do**
  /* when there are not admitted requests and there is computing resource in cloudlets for their VNF instance instantiation */
  $\eta_{max} \leftarrow 0$;
  $sign \leftarrow' No'$; /* terminate the nested loop */
  for $k \leftarrow 1$ to K **do**
    for each cloudlet $v \in V$ **do**
      $U'_k \leftarrow U_k - (U_k \cap A)$; /* unassigned requests in U_k */
      if (cloudlet v has residual computing capacity to accommodate a VNF instance of $f^{(k)}$) and ($U'_k \neq \emptyset$) **then**
        Compute $R_v^{(k)}$, by invoking an approximation algorithm for the knapsack problem with capacity $\lambda \cdot \mu^{(k)}$ and set U'_k;
        Compute the ratio $\eta_{k,v}$ in Eq.(22);
        if $\eta_{k,v} > \eta_{max}$ **then**
          $\eta_{max} \leftarrow \eta_{k,v}$;
          $k_0 \leftarrow k$;
          $v_0 \leftarrow v$;
          $sign \leftarrow' Yes'$; /* whether there is any update */
  if $sign =' Yes'$ **then**
    $C_{v_0} \leftarrow C_{v_0} - C(f^{(k_0)})$; /* update the residual computing resource at cloudlet v_0 */
    $U_{k_0} \leftarrow U_{k_0} \setminus R_{v_0}^{(k_0)}$;
    $cost \leftarrow cost + c_{ins}(f^{(k_0)}, v_0) + \sum_{r_j\in R_{v_0}^{(k_0)}} \rho_j \cdot (c_{proc}(f^{(k_0)}, v_0) + c_{bw}(P(v_j, v_0)))$;
    $S \leftarrow S \cup \{$a VNF instance of $f^{(k_0)}$ created in cloudlet $v_0\}$; /* i.e., its ith instance, $\phi(k_0, i) = v_0$ */
    $A \leftarrow A \cup \{r_j$ is assigned to $I_i^{(k_0)}$ in cloudlet $v_0 \mid r_j \in R_{v_0}^{(k_0)}\}$; /* i.e., $\psi(k_0, j) = i$ and $\phi(k_0, \psi(k_0, j)) = v_0$ */
  else
    $flag \leftarrow false$;
return $cost$, S, and A.

$f^{(k)}$ are explored and an approximation algorithm for the GAP problem is invoked to assign requests into α VNF instances, and Formula (23) is used to select the network function $f^{(k)}$ and set $\mathcal{S}$ of cloudlets to accommodate the VNF instances. Other steps are the same in the `Algorithm` 3, thus its detailed description is omitted.

5.4 Algorithm analysis

The rest is to analyze the time complexity of the proposed algorithms, `Algorithm` 3 and `Algorithm` 4.

Theorem 4. *Given an MEC network $G = (V, E)$, there is a cloudlet $v \in V$ with computing capacity C_v co-located with an AP $v \in V$, a set U of users with each j having a user request $r_j = \langle v_j, f_j^{(k)}, \rho_j \rangle$, there are algorithms,* `Algorithm` *3 and* `Algorithm` *4, for the throughput maximization problem in G, which take $O(K|V||U|^2 \log |U|/\epsilon + |V|^3)$ time and $O(K|V|^{\alpha}|U|(|U| \log \frac{1}{\epsilon} + \frac{1}{\epsilon^4}) + |V|^3)$ time respectively, where ϵ is constant with $0 < \epsilon \leq 1$.*

Proof. The analysis of time complexity of `Algorithm` 3 is similar to the one in the proof body of Theorem 3 for `Algorithm` 1. The only difference between them is whether the computing resource at each cloudlet is capacitated. Specifically, assuming that all pairs shortest paths in G have been calculated, which takes $O(|V|^3)$ time. `Algorithm` 3 proceeds iteratively. Within each iteration, one instance of a specific type of network function will be deployed and some of unassigned requests requested for that network function will be assigned. The VNF instances are deployed one by one until either all requests are admitted or there is no sufficient computing resource at any cloudlet to admit any request any more.

Consider the algorithm now is in its iteration i, a VNF instance of $f^{(k_0)}$ is identified and deployed at a cloudlet v_{l_0}, and a subset of requests in U'_{k_0} is determined and all requests in it will be assigned to the VNF instance. This takes $O(K \cdot |V| \cdot \frac{|U'_{k_0}| \log |U'_{k_0}|}{\epsilon})$ time. As for each type-k network function with $1 \leq k \leq K$, there are no more than $N^{(k)}$ VNF instances of $f^{(k)}$ to be deployed and $N^{(k)} \leq |U_k|$, the number of iterations in `Algorithm` 3 is no more than $\sum_{k=1}^{K} |U_k| = |U|$. The time complexity of `Algorithm` 3 thus is $\sum_{k=1}^{K} O(|U_k| \cdot K \cdot |V| \cdot \frac{|U'_k| \log |U'_k|}{\epsilon} + |V|^3) = O(K \cdot |V| \cdot |U|^2 \log |U|/\epsilon + |V|^3)$ time.

The rest is to analyze the time complexity of `Algorithm` 4. In each iteration, determining the deployment of a group of VNF instance of $f^{(k)}$ on which cloudlets and the assignment of requests in U'_k to α VNF instances take $O(\alpha K|V|^{\alpha}(|U'_k| \log \frac{1}{\epsilon} + \frac{1}{\epsilon^4}))$ time, by invoking an approximation algorithm for the GAP problem, since $|V|^{\alpha}$ combinations of VNF instances are examined. `Algorithm` 2 thus takes $O(K|V|^{\alpha}|U|(|U| \log \frac{1}{\epsilon} + \frac{1}{\epsilon^4}) + |V|^3)$ time and ϵ is constant with $0 < \epsilon \leq 1$. □

6 Performance Evaluation

In this section, we evaluate the performance of the proposed algorithms through experimental simulations. We also investigate the impact of important parameters on the performance of the proposed algorithms.

6.1 Experimental environment settings

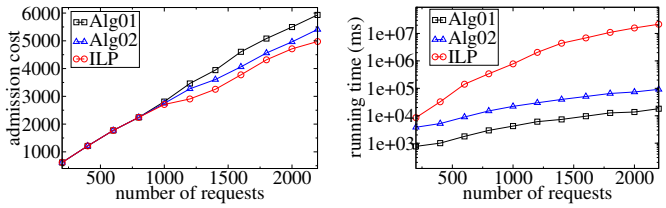
We consider an MEC network $G = (V, E)$ consisting of from 10 to 250 APs (cloudlets). We generate network topologies through the tool GT-ITM [11]. We assume that the computing capacity of each cloudlet varies from 5,000 MHz to 10,000 MHz [15]. Given a cloudlet, the instantiation cost of a VNF instance in it is randomly drawn from $[0.50, 2.0]$, while the processing cost 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 30, and computing demand of different types of VNFs is set from 200 MHz to 800 MHz [15], while the processing rate (capacity) of one type of VNF instance is randomly drawn from 50 to 100 data packets per millisecond [22]. Each user request r_j is randomly generated as follows. A node $v_j \in V$ in G is randomly chosen as its AP of r_j, and its data packet rate ρ_j is randomly drawn from 2 to 10 packets per millisecond [17], 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. Unless otherwise specified, these parameters will be adopted in the default setting.

In the following, we first evaluate the performance of `Algorithm` 1 and `Algorithm` 2 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 VNF instance of its type with sufficient residual processing capacity. If there are multiple such VNF instances, we then choose the one with the minimum request 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 a cloudlet. This process continues until all user requests being admitted. Another heuristic `GreedyCap` is also proposed as a benchmark for the throughput maximization problem, which follows similar idea with the one for algorithm `GreedyNonCap`. Specifically, `GreedyCap` examines requests one by one, and a request is assigned to a VNF instance with sufficient residual processing capacity and the minimum admission cost. Otherwise, if there is no VNF instance of its type having sufficient residual processing capacity for the request, a new VNF instance with the minimum instantiation cost will be instantiated in some cloudlet with sufficient residual computing capacity. This process continues until all user requests being admitted or no more requests can be admitted without avoiding processing capacity of each VNF instance or computing capacity of each cloudlet.

6.2 Performance evaluation of the proposed algorithms for the cost minimization problem

We first investigate the performance of `Algorithm` 1 and `Algorithm` 2 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, and fixing the look-ahead parameter α at 2. Fig. 2

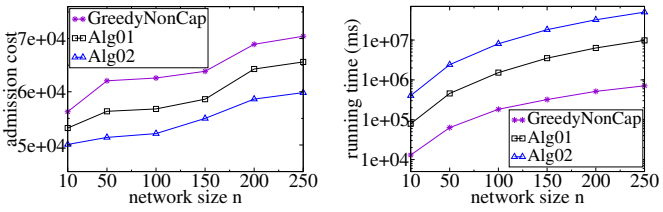
illustrates the admission cost of requests and running time of the three mentioned algorithms. It can be seen from Fig. 2 (a) that both Algorithm 1 and Algorithm 2 can achieve a near optimal admission cost, i.e., the admission cost delivered by Algorithm 1 is no more than 19.27% of the optimal one while the admission cost delivered by Algorithm 2 is no more than 8.75% of the optimal one. Fig. 2 (b) demonstrates the running times of these three algorithms. It can be seen that the running times of both Algorithm 1 and Algorithm 2 are only a small fraction of that of algorithm ILP, while their solutions are comparable with the optimal one. In particular, Algorithm 1 takes less than 20 seconds and Algorithm 2 takes less than 90 seconds, while algorithm ILP solution takes more than 4 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.



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

Fig. 2. Performance of Algorithm 1, Algorithm 2, 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 and Algorithm 2 against the benchmark heuristic GreedyNonCap, by varying network size from 10 to 250 for 20,000 user requests, and fixing the look-ahead parameter α at 2. Fig. 3 depicts the performance curves of the two mentioned algorithms. From Fig. 3 (a) we can see that both Algorithm 1 and Algorithm 2 outperform their benchmark counterpart GreedyNonCap. Specifically, with the increase on network size, the admission costs of all algorithms grow too. However, the performance gap between them becomes larger and larger. As shown in the figure, Algorithm 1 has 9.29% less admission cost than that by algorithm GreedyNonCap when the network size is 100, and 6.86% less admission cost when the network size is 250. And Algorithm 2 has 16.69% less admission cost than that by algorithm GreedyNonCap when the network size is 100, and 15.04% less admission cost when the network size is 250. It is noticeable that Algorithm 2 delivers a better solution than Algorithm 1 in all cases, which justifies the effectiveness of the look-ahead strategy. Fig. 3 (b) shows the running time curves of Algorithm 1, Algorithm 2, and algorithm GreedyNonCap. It can be seen that algorithm GreedyNonCap takes the least running time, while Algorithm 2 takes the longest running time in all network sizes. This is due to the fact that algorithm GreedyNonCap just places each request greedily to a VNF instance with the minimum admission cost, while Algorithm 2 strives for finding a good assignment of requests to share a set of VNF instances.

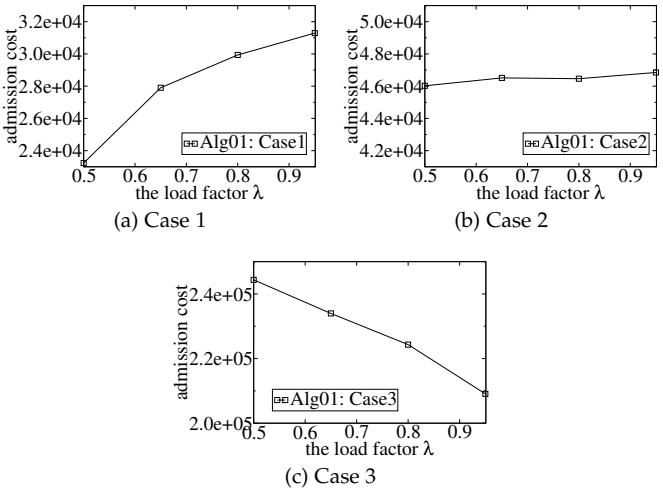


(a) The admission cost (b) The running time

Fig. 3. Performance of Algorithm 1, Algorithm 2, and algorithm GreedyNonCap when admitting 20,000 requests, by varying the number of APs from 10 to 250.

6.3 Impact of load factor λ on the performance of resource usage tradeoffs for cost minimization

We also investigate 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 performance of resource usage tradeoffs for admission cost minimization is shown in Fig. 4.



(a) Case 1 (b) Case 2

(c) Case 3

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

We investigate the impact of λ on the algorithm performance by the following three cases.

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 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 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.

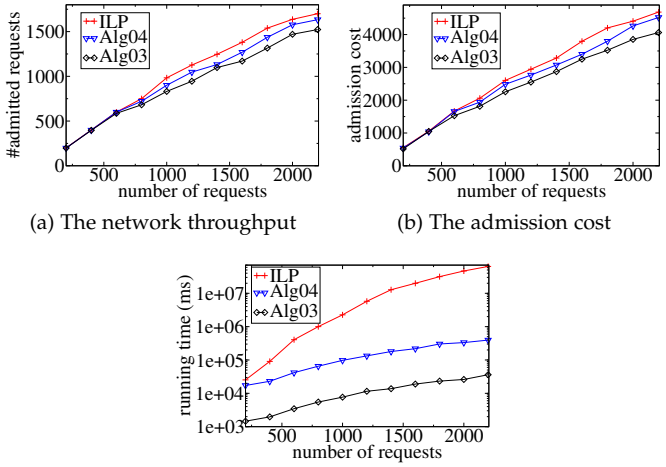
Case 3. The instantiation cost of a VNF instance is much more expensive than the bandwidth usage cost. In this case, the cost per unit of computing resource consumed in a VNF instance instantiation is drawn from $[5.0, 20.0]$. As can be

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, the admission cost is 244,357.76. However, when the load factor λ increases to 0.95, the admission cost reduces to 209,091.10. In other words, around 14.43% of the admission cost can be saved by fine tuning on the value of the load factor λ.

The rationale behinds is that the load factor λ can be used to adjust the workload of cloudlets depends on the costs of different resources in a network. When a certain resource becomes more expensive, the load factor λ can be tuned to achieve a much cheaper admission cost.

6.4 Performance evaluation of the proposed algorithms for the throughput maximization problem

We thirdly evaluate the performance of `Algorithm 3` and `Algorithm 4` against the optimal solution the ILP for the throughput maximization problem, by varying the number of requests from 200 to 2,200 while fixing the number $|V|$ of APs at 10, and the look-ahead parameter α at 2. Fig. 5 depicts the network throughput, admission cost and running time of each of the three comparison algorithms. We can see from Fig. 5 (a) that both `Algorithm 3` and `Algorithm 4` can achieve a near optimal throughput, i.e., `Algorithm 3` and `Algorithm 4` achieve at least 89.05% and 93.30% of the performance compared with the optimal solution delivered by the ILP, respectively. Specifically, `Algorithm 3` can achieve as much network throughput as `ILP` solution does, when the number of user requests is less than 600, and it can achieve 96.24% of the optimal network throughput delivered by the ILP when the number of requests is no more than 1,800. The similar performance behaviors can be observed between `Algorithm 4` and the ILP too. Fig. 5 (b) demonstrates the running times of these three algorithms. It can be seen that the ILP takes a prohibitively long time, while both algorithms `Algorithm 3` and `Algorithm 4` take a small fraction of the running time of the ILP. Notice that with the increase on the number of requests, the ILP fails to deliver a solution within a reasonable amount of time when the number of requests reaches 2,400.

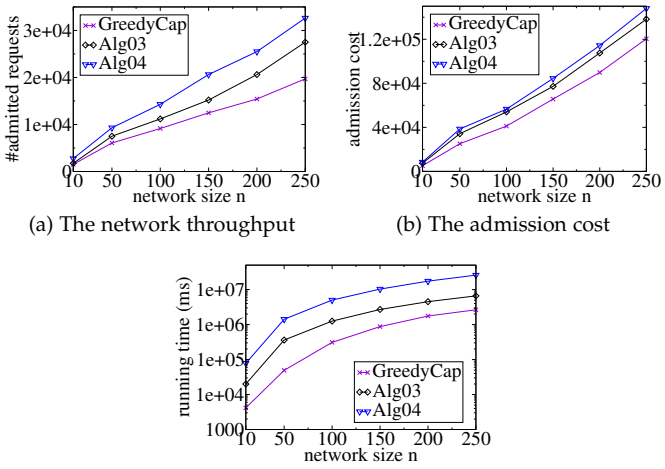


(a) The network throughput (b) The admission cost

(c) The running time

Fig. 5. Performance of `Algorithm 3`, `Algorithm 4`, and algorithm `ILP` for the throughput maximization problem by varying the number of requests from 200 to 2,200.

In the following we study the performance of `Algorithm 3` and `Algorithm 4` for the throughput maximization problem against a baseline heuristic `GreedyCap`, by varying the network size from 10 to 250 for 40,000 requests. The results are shown in Fig. 6. We can see from Fig. 6 that all algorithms deliver solutions with increasing throughput, along with the increase on the network size. This is due to more cloudlets are available for request admissions. However, `Algorithm 3` and `Algorithm 4` outperform algorithm `GreedyCap` significantly. Specifically, `Algorithm 3` can admit on average 27,514.26 requests with the admission cost 138,238.13, while algorithm `GreedyCap` can only admit on average 19,711.15 requests with admission cost 120,471.47. `Algorithm 3` can admit 25.9% more requests than that by algorithm `GreedyCap` with only 14.7% more admission cost, when the network size is 250. The performance of `Algorithm 4` is even better than that of `Algorithm 3`. Specifically, `Algorithm 4` can admit on average 20,621.70 requests with admission cost 84,447.25, while `Algorithm 3` can only admit on average 15,203.76 requests with admission cost 65,683.50 for a network with size 150. Fig. 6 (b) plots the running time curves of the mentioned three algorithms, where algorithm `GreedyCap` takes the least running time, while `Algorithm 4` takes the longest running time in all network sizes.



(a) The network throughput (b) The admission cost

(c) The running time

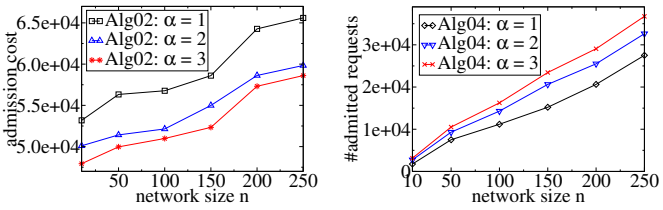
Fig. 6. Performance of `Algorithm 3`, `Algorithm 4`, and algorithm `GreedyCap` by varying the number of APs from 10 to 250.

6.5 Impact of important parameter α on the algorithm performance

We finally study the impact of the look-ahead parameter α on the performance of `Algorithm 2` and `Algorithm 4` for a set of 20,000 requests in an MEC with 10 APs, and varying α from 1 to 3. From Fig. 7 (a) it can be seen that the larger the look-ahead parameter α, the less admission cost delivered by `Algorithm 2`. Similarly, from Fig. 7 (b) it can be seen that the larger the look-ahead parameter, the more network throughput achieved by `Algorithm 4`.

7 Conclusions

In this paper we studied the provisioning of virtualized network function services for mobile users by admitting their requests in an MEC network, where each user requested a



(a) The admission cost (b) The network throughput

Fig. 7. Impact of the look-ahead parameter α on the performance of `Algorithm` 2 and `Algorithm` 4.

specific network function service. We aim to maximize the number of user requests admitted while minimizing their admission cost. We first formulated the cost minimization problem by formulating an ILP solution and two efficient heuristic algorithms assuming that there is no limitation of computing resource in cloudlets. We then dealt with the throughput maximization problem by admitting as many as requests while minimizing their admission cost, subject to computing capacity on each cloudlet, for which we provided an ILP solution, followed by two efficient heuristics. We finally evaluated the performance of the proposed algorithms through experimental simulations. Experimental results demonstrate that the proposed algorithms are promising.

ACKNOWLEDGEMENT

We appreciate the two anonymous referees for their constructive comments and invaluable suggestions, which help us improve the quality and presentation of the paper greatly. The work by Yu Ma, Weifa Liang and Meitian Huang was supported by Australian Research Council under its Discovery Project Scheme with Grant No. DP200101985, and the work of Wenzheng Xu was supported by the National Natural Science Foundation of China (NSFC) with grant number 61602330, Sichuan Science and Technology Program (Grant No. 2018GZDZX0010, 2018GZ0094, 2018GZ0093, 2017GZDZX0003).

REFERENCES

[1] H. Alameddine, S. Sharafeddine, S. Sebbah, S. Ayoubi, and C. Assi. Dynamic task offloading and scheduling for low-latency IoT services in multi-access edge computing. *IEEE Journal on Selected Areas in Communications*, vol.37, no.3, pp.668–682, IEEE, 2019.

[2] M. Charikar, S. Guha, *É*. Tardos, and D. B. Shmoys. A constant-factor approximation algorithm for the k-median problem. *Proc of STOC'99*, pp.1–10, ACM, 1999.

[3] M. Chen and Y. Hao. Task offloading for mobile edge computing in software defined ultra-dense network. *IEEE Journal on Selected Areas in Communications*, vol.36, no.3, pp.587–597, IEEE, 2018.

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

[5] 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.

[6] R. Cohen, L. Katzir, and D. Raz. An efficient approximation for the generalized assignment problem. *Info. Proc. Lett.*, vol.100, pp.162–166, Elsevier, 2006.

[7] T. H. Cormen, C. E. Leiserson, R. L. Riverst, and C. Stein. *Introduction to Algorithms*. 3rd edition, MIT Press, 2009.

[8] Q. Fan and N. Ansari. Application aware workload allocation for edge computing-based IoT. *IEEE Internet of Things Journal*, vol.5, no.3, pp.2146–2153, 2018.

[9] Q. Fan and N. Ansari. On cost aware cloudlet placement for mobile edge computing. *IEEE/CAA Journal of Automatica Sinica*, vol.6, no.4, pp.926–937, 2019.

[10] H. Feng, J. Llorca, A. M. Tulino, D. Raz, and A. F. Molish. Approximation algorithms for the nfv service distribution problem. *Proc of INFOCOM'17*, IEEE, 2017.

[11] GT-ITM. http://www.cc.gatech.edu/projects/gtitm/.

[12] 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'18*, IEEE, 2018.

[13] S. Hu and G. Li. Dynamic request scheduling optimization in mobile edge computing for IoT applications. *IEEE IoT journal*, vol.7, no.2, pp.1426–1437, 2020.

[14] M. Jia, J. Cao, and W. Liang. Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks. *IEEE Transactions on Cloud Computing*, vol.5, no.4, pp.725–737, 2017.

[15] M. Jia, W. Liang, and Z. Xu. QoS-aware task offloading in distributed cloudlets with virtual network function services. *Proc. of MSWiM'17*, ACM, 2017.

[16] M. Jia, W. Liang, Z. Xu, and M. Huang. Cloudlet load balancing in wireless metropolitan area networks. *Proc of INFOCOM*, IEEE, 2016.

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

[18] Y. Ma, W. Liang, and J. Wu. Online NFV-enabled multicasting in mobile edge cloud networks. *Proc. of ICDCS'19*, IEEE, July, 2019.

[19] Y. Ma, W. Liang, J. Wu, and Z. Xu. Throughput maximization of NFV-enabled multicasting in mobile edge cloud networks. *IEEE Transactions on Parallel and Distributed Systems*, vol.31, no.2, pp.393–407, 2020.

[20] P. Mach, and Z. Becvar. Mobile edge computing: a survey on architecture and computation offloading. *IEEE Commu. Surveys & Tutorials*, vol.19, no.3, pp.1628–1656, Jun. 2017.

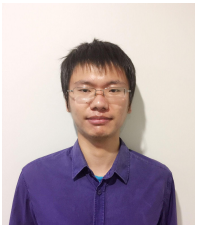
[21] Y. Mao, C. You, J. Zhang, K, Huang and K. Letaief. A survey on mobile edge computing: the communication perspective. *IEEE Commun. Surv. Tutor.*, vol.19, pp.2322–2358, 2017.

[22] J. Martins, M. Ahmed, C. Raiciu, V. Olteanu, M. Honda, R. Bifulco, and F. Huici. ClickOS and the art of network function virtualization. *Proc. of NSDI*, USENIX, 2014.

[23] Y. Ren, Z. Weng, Y. Li, Z. Xie, K. Song, and X. Sun. Distributed task splitting and offloading in mobile edge computing. *Proc. of ChinaCom*, 2019.

[24] Q. Xia, W. Liang, and W. Xu. Throughput maximization for online request admissions in mobile cloudlets. *Proc of LCN'13*, IEEE, 2013.

[25] 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.

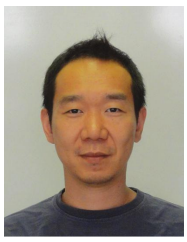


Yu Ma received his BSc degree with the first class Honours in Computer Science at the Australian National University in 2014. He is currently a PhD candidate in the Research School of Computer Science at the Australian National University. His research interests include Software Defined Networking, Internet of Things (IoT), and Social Networking.

Weifa Liang (M'99–SM'01) received the PhD degree from the Australian National University in 1998, the ME degree from the University of Science and Technology of China in 1989, and the BSc degree from Wuhan University, China in 1984, all in Computer Science. He is currently a Professor in the Research School of Computer Science at the Australian National University. His research interests include design and analysis of energy efficient routing protocols for wireless ad hoc and sensor networks, Mobile Edge Computing (MEC), Network Function Virtualization (NFV), Software-Defined Networking (SDN), design and analysis of parallel and distributed algorithms, approximation algorithms, combinatorial optimization, and graph theory. He is a senior member of the IEEE.

Meitian Huang received the BSc, and PhD degrees in Computer Science at the Australian National University in 2014, and 2020. His research interests include software-defined networking, virtualized network function services, algorithm design and analysis, and cloud computing.

Wenzheng Xu (M'15) received the BSc, ME, and PhD degrees in computer science from Sun Yat-Sen University, Guangzhou, P.R. China, in 2008, 2010, and 2015, respectively. He currently is an Associate Professor at Sichuan University. Also, he was a visitor at both the Australian National University and the Chinese University of Hong Kong. His research interests include wireless ad hoc and sensor networks, mobile computing, approximation algorithms, combinatorial optimization, online social networks, and graph theory. He is a member of the IEEE.



Song Guo (M'02-SM'11-F'19) is a Full Professor at Department of Computing, The Hong Kong Polytechnic University. He received his Ph.D. in computer science from University of Ottawa and was a professor with the University of Aizu. His research interests are mainly in the areas of big data, cloud computing and networking, and distributed systems with over 400 papers published in major conferences and journals. Prof. Guo was an Associate Editor of IEEE Transactions on Parallel and Distributed Systems and an IEEE ComSoc Distinguished Lecturer. He is now on the editorial board of IEEE Transactions on Emerging Topics in Computing, IEEE Transactions on Sustainable Computing, IEEE Transactions on Green Communications and Networking, and IEEE Communications.