Toward Latency-Optimal Placement and Autoscaling of Monitoring Functions in MEC

QUAN YUAN, XINSHEng JI, HONGBO TANG, AND WEI YOU
Information Technology Research Center, PLA Strategic Support Force Information Engineering University, Zhengzhou 450002, China
Corresponding author: Quan Yuan (b101180153@smail.nju.edu.cn)

This work was supported in part by the National Key Research and Development Program Cyberspace Security Special under Grant 2016YFB0801605, and in part by the National Natural Science Foundation Innovative Groups Project of China under Grant 61521003.

ABSTRACT
Multi-Access Edge Computing (MEC) promises to provide sufficient computing capacity close to users and realize smart management at the edge of mobile network. To achieve the aforementioned objectives, it is indispensable to implement real-time monitoring of the whole MEC network. However, the geo-distributed deployment of MEC infrastructure dramatically increases the communication latency of gathering the state information (e.g., current status and resource utilization) from servers at the edge. This paper addresses latency-optimal placement and autoscaling of monitoring functions in MEC. First, we formally formulate latency-optimal placement of monitoring functions as an integer linear programming problem and proposed a genetic algorithm-based meta-heuristic to obtain the optimal solution with fast convergence. Moreover, to serve the time-varying demand on resource capacity from diversified mobile services, an online VNF scaling scheme is designed for realizing on-demand resource allocation. The effectiveness of our heuristic algorithm is verified through both numerical simulation and experiments in real cloud environment. Experimental results demonstrate performance superiority of the proposed approach over the state-of-art researches, in terms of algorithm CPU time, total network latency and long-term scaling cost.

INDEX TERMS
Network function virtualization, multi-access edge computing, virtualized network function, latency-optimal placement, genetic algorithm.

I. INTRODUCTION
In deploying the fifth generation of mobile networks (5G), mobile network operators (MNOs) endeavor in implementing three main types of use cases - enhanced mobile broadband (eMBB), ultra-reliable low latency communications (URLLC), and massive machine type communications (mMTC) [1]. Featured with exponential growth of connectivity density and data traffic as well as millisecond network latency, the communication defined in these use cases entails higher processing and transmitting capability of mobile cloud infrastructure. To achieve the visions of 5G communications, recent years have seen a paradigm shift in mobile computing, from centralized mobile cloud computing towards multi-access edge computing (MEC) [2].

Emerging as a key enabling technology for a set of next-generation mobile services (e.g., internet of things, content distribution network and vehicular communication), MEC promises to move sufficient computing capacity close to diverse users by deploying thousands of high-performance servers (HPS) and network devices at multi-access edges across a city district [3]. Such a dense and geo-distributed deployment will sharply increase operating expenditure (OPEX) for managing the infrastructure [4]. To provide flexible and efficient cloud management, MNOs tend to abstract software-defined network function from dedicated middleboxes by introducing software-defined networking (SDN) and network function virtualization (NFV) in MEC architecture [5].

NFV instantiates network middleboxes and entities over virtual machine (VM) or container as virtualized network functions (VNF) and connects the VNFs as a service function chain (SFC) to provide network services. By nature, NFV changes the way network services are constituted and the SFC technology enhances the elasticity, scalability and availability of mobile services. To fulfill these features, a centralized NFV management and network orchestration (MANO) module is designed in NFV architecture, which integrates management functions from infrastructure layer to application layer [6]. Serving as a requirement for dynamic scheduling,
advanced security measures and disaster recovery mechanism in NFV MANO, monitoring function deployment is a task of paramount importance for collecting real-time state information of every single HPS (e.g. running state and resource utilization, etc.) and furthermore building global view of the whole NFV infrastructure [7]. However, in the geo-distributed MEC infrastructure, frequent remote communication between edge HPSes and NFV MANO dramatically increases the total latency of network monitoring, giving rise to a new challenge on how to maintain the real-time character of monitoring system. Focusing on solving this problem, we design a latency-aware monitoring function placement and autoscaling approach in MEC, aiming at reducing both total latency of distributed monitoring system and scaling cost incurred by creating and removing monitoring function instances.

Despite that VNF placement has been well-studied in virtualized cloud data center but to the best of our knowledge, only a few works have been done under the context of NFV-based MEC infrastructure [8], [9] or in specific use case of monitoring function placement [10]. In general, recent researches on latency-aware VNF placement can be roughly divided into two classes-those that rely on offline algorithm to realize optimal resource allocation and process service requests in batches [8]–[10] and those that resort to online algorithm to enable real-time scaling and processing [11], [12]. But in terms of distributed monitoring system deployment, both of the aforementioned approaches have their own limitations: i) the offline approaches periodically service request in batches, and thus can hardly guarantee that the detected events are available on time for their intended use. ii) The online approaches generally incur frequent creation and migration of VMs, which consumes large amount of computing resources for VNF management and routing rules update, leading to high scaling cost and low service availability.

To overcome the limitation of the existing approaches, this research work proposes a composite approach to deploy monitoring functions in distributed edge cloud, in which two sub-problems are included. In problem 1, we try to find which servers are adequate to host monitoring function towards the minimum total latency of the whole monitoring system. In problem 2, the key work is to determine how many monitoring function instances are supposed to be instantiated on each monitoring server according to the real-time service requests. In particular, the twofold contributions are explained as follows.

- We formally use integer linear programming to formulate the latency-optimal monitoring function placement problem in the context of MEC infrastructure. In our system model, we design a hierarchical monitoring system topology for better scalability in NFV-based MEC environment. To get the fast convergence of the proposed problem, a genetic algorithm-based meta-heuristic is proposed, which can adaptively adjust the scale of monitoring resource pool according to the scale of MEC infrastructure.

- It is a key challenge for existing online and offline placement approaches to reduce scaling cost and increase service availability simultaneously. Towards this objective, we design an online scaling scheme to realize real-time and on-demand resource allocation in our monitoring system. A dynamic lifecycle management approach is presented via Ski-Rental model and then an online algorithm is designed to find the optimal tradeoff between scaling cost and service availability. The performance of proposed scheme has been verified in real NFV-based cloud environment. The result suggests that our proposed approach successfully reduces the scaling cost and outperforms the compared approaches significantly across all performance metrics.

The rest of the paper is organized as follows. In Section 2, a brief background on the related works is presented. Section 3 gives the system model and problem formulation of monitoring function placement. In section 4, we propose an online scaling scheme based on Ski-Rental model. Simulation and experimental results are presented in section 5 and conclusions in Section 6.

II. RELATED WORK

During the past several years, a large number of researches have been carried out in the area of VNF placement and scaling strategies in central cloud data center. Among these recent works, we put an emphasis on the portion aiming at reducing network latency and improving service availability. Cziva et al. used mixed integer programming to formulate latency-aware VNF placement problem and presented a heuristic to dynamically re-schedule the optimal placement policy via optimal stopping theory [8]. Jiang et al. formulated the placement of monitoring functions in cloud data center as a 0-1 programming problem and designed an advanced quantum genetic algorithm to find the optimal mappings for reducing overall network latency [10]. Tang et al. implement a dynamic VNF instance scaling system in real operator network. By introducing a traffic forecast function, such system could proactively scale in VNF instances towards high service availability [13]. Alawe et al. presented an RNN-based solution that could dynamically schedule the virtualized elements of the 5G generation mobile core network in order to reduce provision latency of control plane functions [14]. All the above-mentioned research works used an offline formulation to describe the latency-optimal VNF placement and scaling problem. In their system model, they assumed that NFV MANO could obtain complete system knowledge in a time instance, during which VNF placement could be reduced to an offline optimization problem. Simulation results showed that performance of offline approaches reached closely to theoretical optimality but they could not make immediate response to the burst traffic emerging from unpredicted service requests in real operator network environment. Contrary to the offline placement algorithm, online approaches focus on performance optimization with incomplete information. Ferdaus et al. proposed a greedy heuristic for the on-demand
resource allocation of multi-layer cloud application in the data center interconnect [11]. Cho et al. designed an online algorithm to find the optimal VM that is available for hosting required network functions within a predefined network latency threshold [12]. Jia et al. implemented an online scaling solution of NFV service chains across geo-distributed datacenters, where a regularization-based approach from online learning literature was used to convert the offline optimal deployment problem into a sequence of one-shot regularized problems and then an online dependent rounding scheme was proposed to derive optimal integer solutions [15]. Experimental results indicated that online algorithm involved frequent creation and migration of VM, which incurred burdensome scaling cost and reduced the service availability.

Several other recent VNF placement and scaling works have been proposed focusing on QoS improvement in MEC. Yala et al. proposed a genetic algorithm-based VNF placement scheme that is applicable to NFV-based MEC environment with the purpose of minimizing access latency at mobile edge cloud [9]. Through simulation evaluation, the authors showed that the proposed scheme achieved faster convergence compared to traditional MIP solver CPLEX. Subramanya et al. addressed the issue of dynamic VNF scaling and placement at the network edges with the focus on minimizing end-to-end latency [16]. The latency-optimal placement was realized through an integer linear programming solution with the assistance of a multilayer perceptron-driven classifier. Gouareb et al. studied the problem of VNF placement and routing across the distributed physical hosts to minimize the queuing delay within the edge clouds [17]. A non-linear integer mathematical program was first used to formulate such problem along with a linearization solution. The superiority of performance has been validated in small and medium scale networks over some scale-free heuristics.

As far as our knowledge goes, placement of monitoring functions in distributed edge NFV infrastructure has not been studied before. Towards solving such problem, the aforementioned researches have following limitations. First, the existing monitoring function placement approach designed for cloud data center [10] applied a centralized virtual network topology of monitoring system, ignoring the optimization of distant communication latency in the context of distributed MEC architecture. Moreover, for other works in the context of cloud data center, the offline approaches [8], [10], [13], [14] can hardly handle burst service requests from massive edge servers and the online methods [11], [12], [15] involve frequent VNF migration and creation across different servers, which incurs burdensome scaling cost in mobile edges. In addition, given the monitoring service demand in terms of timeliness and availability [7], recent research works mainly focused on reducing the transmission and access latency in distributed edges [9], [16], [17], while none of them can realize latency-optimal placement and low-cost real-time scaling simultaneously. To provide a comprehensive solution to this problem, this paper proposes a composite approach for deploying distributed monitoring system in MEC, comprising both latency-optimal monitoring function placement and resource-efficient online scaling.

III. LATENCY-OPTIMAL EDGE MONITORING FUNCTION PLACEMENT

Our steps of this section go as followings. Firstly, we describe the system model of edge monitoring function placement and then formulize the latency-optimal placement problem as integer linear programming (ILP). Finally, a genetic algorithm-based approach is presented to find the optimal edge servers for placing the monitoring functions.

A. SYSTEM MODEL

Based on the CONCERT architecture [18], our system model of monitoring function placement in MEC is presented in Figure 1. In the system, data plane consists of wireless access point, SDN switches, and computing resources (including both regional and central servers in the distributed cloud). Control plane is basically composed of a conductor, which is a cluster of control functions managing both physical and virtualized infrastructure of the CONCERT architecture. Within the conductor, monitoring system is deployed in a hierarchical manner for better scalability. In terms of management domain and system role, components of the monitoring system can be hierarchically classified as central monitoring function (CMF) and edge monitoring function (EMF). Specifically, CMF is in charge of gathering, aggregating, formatting and presenting global information such as running status and resource utilization of physical and virtualized infrastructure and provides uniform and open service interface to other control plane functions (e.g. VNF scheduler, fault diagnostics and dashboard in Openstack). EMF is responsible for collecting information of proximal regional servers and then aggregating and reporting it to CMF. Instead of using the centralized monitoring system architecture in current data center, the rationale behind the hierarchical design in the context of MEC can be reduced to two points. On one hand, in the hierarchical monitoring system, regional servers tend to report state information to proximal EMF, thus avoiding direct communication between regional and central servers. On the other hand, given the
massive amount of regional servers, hierarchical placement approach gains better efficiency in terms of load balance and scalability [7]. As has been shown in Figure 1, the CMF is placed on a central server and EMFs are instantiated on regional servers. Every single regional server first reports their state information to a proximal EMF and then these information will be aggregated and forwarded to the CMF periodically. The main objective is to find adequate edge servers to host EMFs with the purpose of minimizing overall latency of the proposed monitoring system.

B. PROBLEM FORMULATION AND ALGORITHMIC SOLUTIONS

Table 1 lists all the key notations in our placement and scaling model. Formally, notations used in placement problem formulation are introduced as follows. We denote the physical network as an undirected graph \( G = (S, E) \), where \( S \) and \( E \) represent the set of physical servers and links between them respectively. The number of regional servers is assumed to be \( n \). We also assumed that all regional servers can provide computing, storage and IO resources for the placement of EMFs respectively. The number of regional servers is assumed to represent the set of physical servers and links between them. Formally, notations used in placement problem formulation are introduced as follows. We denote the physical network as an undirected graph \( G \), the element \( l_{ij} \) represents the shortest path latency between regional server \( i \) and \( j \). The element \( d_{ij} \) represents latency between regional server \( i \) and the central server that hosts the CMF. If the EMF is placed on regional server \( i \), then \( x_i = 1 \), otherwise \( x_i = 0 \).

**Table 1. Notations and descriptions of basic variables in placement and scaling model.**

| Parameters in Problem 1 | Descriptions |
|-------------------------|--------------|
| \( S \)                 | the set of regional servers |
| \( E \)                 | the set of physical links |
| \( n \)                 | the number of regional servers |
| \( L_{eil} \)           | the element \( l_{ij} \) represents the shortest path latency between regional server \( i \) and \( j \) |
| \( D_{eil} \)           | the element \( d_{ij} \) represents latency between regional server \( i \) and the central server that hosts the CMF, if the EMF is placed on regional server \( i \), then \( x_i = 1 \), otherwise \( x_i = 0 \).
| \( m(i) \)              | the EMF that is in charge of server \( i \) |

Parameters in Problem 2:

| Parameters in Problem 2 | Descriptions |
|-------------------------|--------------|
| \( M \)                 | the set of monitoring servers |
| \( \tau \)              | the whole-time span |
| \( m \)                 | the number of monitoring servers |
| \( \eta \)              | processing capability of an EMF |
| \( b \)                 | the number of physical resource types |
| \( H_{eil} \)           | the element \( h_i \) represents the required capability of resource \( i \) for creating an EMF |
| \( Y_{eil} \)           | the element \( y_i \) represents the number of running EMFs on monitoring server \( i \) at time instance \( t \) |
| \( P_{eil} \)           | the element \( p_i \) represents the number of suspended EMFs |

**Problem 1.** Given the set of regional servers, the latency matrix \( L_{n \times n} \) and latency column vector \( D_{n \times 1} \), try to find the optimal locations (regional servers) to place EMFs with the purpose of minimizing the overall latency of monitoring system.

Minimize:

\[
\varepsilon = \sum_{i=1}^{n} x_i \cdot d_i + \sum_{i=1}^{n} (1 - x_i) \cdot l_{i,m(i)}
\]

Subject to:

\[
m(i) = \{ k | l_{i,k} = \min_{j \in S, x_j = 1} l_{i,j} \} \quad \forall i \in S
\]

Equation (1) describes the objective of monitoring function placement problem where \( \varepsilon \) denotes the overall latency of the monitoring system. The first term represents the latency when all EMFs report aggregated state information to the CMFs. The second term denotes the latency when EMFs gathering the individual state information from every regional server, in which \( m(i) \) represents the EMF that is in charge of server \( i \). Equation (2) ensures that the EMF \( m(i) \) is the one with shortest end-to-end latency to server \( i \) among all EMFs. Equation (3) defines the domain of decision variable \( x_i \).

To obtain fast convergence of problem 1, we present a genetic algorithm meta-heuristic tailored to our model. The detailed flow is specified in Algorithm I. Based on existing researches [20], we make some improvements to classical genetic algorithm in terms of the two key operations during iterative evolution. A ranking-crossover scheme is used in our algorithm, which selects the genes with a higher fitness ranking to produce higher quality offspring and realize faster convergence than classical method. Moreover, a population cataclysm operation is launched to get rid of falling fast into local optimal trap. When the optimal solution keeps unchanged for a certain generations, we will kill all the chromosomes except for the optimal one in current generation and use the newly generated population for further iteration. Algorithm I outputs the optimal decision variable \( X^* \), representing a set of regional servers that are capable to host EMFs in latency-optimal solution. In the following part of this paper, we name these servers as monitoring servers for simplification.

In this section, we have solved the problem of where to place the EMFs towards minimum overall latency of the distributed monitoring system. In the next section, we will focus on how to adjust the number of EMF instances on each monitoring server, according to time-varying resource capacity requests.

IV. ONLINE EMF SCALING SCHEME

In this section, we propose an online EMF scaling scheme to dynamically adjust the number of EMF instances on each monitoring server. The following of this section first
Algorithm 1 Genetic Algorithm-Based Meta-Heuristic for Problem 1

Input: the set of edge servers \( \mathcal{S} \), latency matrix \( L_{n \times n} \) and latency column vector \( D_{n \times 1} \)
Output: decision variable \( X_{n \times 1} \)

1. Define: chromosomes of initial population, selection probability \( p_s \), mutation probability \( p_m \), maximum generation \( maxgen \) and maximum indicator \( maxind \)
2. Randomly initialize each chromosome of initial population as a feasible binary decision variable \( \{x_1, x_2, \ldots, x_n\}, x_i \in \{0, 1\} \)
3. Calculate the fitness \( \varepsilon \) defined in equation (1) among all chromosomes of initial population
4. \( gen = 0 \), cataclysm_indicator = 0
5. while \( gen < maxgen \)
6. Rank chromosomes of current population according to their fitness
7. Select chromosomes with higher fitness using roulette method according to \( p_s \)
8. Execute position-based crossover to obtain next generation chromosomes
9. Execute individual mutation to generate new chromosomes
10. Calculate the fitness of newly generated chromosomes
11. Record the optimal chromosome in current generation and population cataclysm
12. if current optimal chromosome \( \equiv \) the previous optimal chromosome
13. cataclysm_indicator = + 1
14. end if
15. if cataclysm_indicator > maxind and \( gen \leq maxgen/2 \)
16. Execute population cataclysm operation to avoid falling into the local optimal solution
17. cataclysm_indicator = 0
18. Recombine higher-fitness chromosomes to get next-generation population
19. \( gen + 1 \)
20. end if
21. Return the optimal chromosome \( X^* = [x^*_1, x^*_2, \ldots, x^*_n] \)

of the final generation

describes problem statement and mathematical formulation of EMF scaling and then specifies the detailed design of scaling scheme.

A. PROBLEM STATEMENT AND FORMULATION

It is inevitable to see bursty traffic arriving at or leaving regional servers, given the frequent and unpredictable location shift of mobile users, e.g. mass gathering for commercial conference and sports event as well as tidal effect of population mobility in cities. In these cases, regional servers should be either immediately put into operation to serve the bursty traffic or shut down to improve service availability and energy efficiency. Accordingly, the number of EMFs should be adjusted adaptively. Due to that no complete knowledge of user requests can be provided to the NFV MANO in advance, the existing offline placement approaches is inapplicable. For online VNF placement, the NFV MANO is supposed to dynamically adjust the number of VNFs hosted on monitoring servers to provide on-demand service capacity. When the number of running VNFs on a server falls down a predefined threshold value, the NFV MANO may choose to migrate the remaining VNFs and shut down the server for energy saving [21]. To provide on-demand resource capacity for EMFs, we propose an online scaling scheme based on the aforementioned placement approach. The system model and formulation are presented as follows.

In our model, we assume that all EMFs are instantiated by one image that stores in a centralized VNF repository and the MANO allocates the same CPU, RAM and disk resources to every single EMF instance. Hence, every single EMF can be deemed as possessing the same process capability. The scaling of monitoring function is realized by adjusting the number of running EMFs on each monitoring server. Generally, with the objective of providing high services availability, current online scaling approaches choose to shut down idle VM that are temporarily not processing any requests and recreate a new VM when necessary [11], [12]. However, these researches ignore the scaling cost incurred by frequent creation of VM, which involves transferring a VM image to target regional server, configuring network and route parameter and booting the VM. Our scaling approach focuses on finding the tradeoff between high service availability and low scaling cost by presenting a dynamical lifecycle management scheme. Instead of shutting down an idle EMF instance immediately, we tend to suspend the EMF and generate an optimal suspension lifecycle based on our online scaling algorithm. During the lifecycle, the EMF will be reused in priority to reduce scaling cost. Otherwise, it will be shut down to improve service availability.

The basic notations in our scaling model are explained as follows. We define the set of monitoring servers as \( \mathcal{M} \), \( \mathcal{M} = \{i | i \in \mathcal{S}, x_i = 1 \} \) and denote the whole-time span as \( T \). The number of monitoring servers are assumed to be \( m \). The processing capability of every single EMF is defined as \( \eta \), representing that one EMF can process individual state information from \( \eta \) regional servers at most. The number of physical resource types is defined as \( \beta \). The element \( h_i \) of resource vector \( H_{\beta \times 1} \) represents the required capability of resource \( i \) (e.g., CPU, RAM and disk) for creating an EMF. Similarly, we use a \( b \times 1 \) column vector \( C_i \) to denote the resource capability of monitoring server \( i \). Based on the assumption, we define the following two decision variables-running status vector \( V_{m \times 1} \) and suspended status vector \( P_{m \times 1} \). To be specific, the element \( y^*_i \) and \( p^*_i \) represent the number of running and suspended EMFs on monitoring server \( i \) at time instance \( t \), respectively. Finally, we define the EMF scaling problem as follows.
Problem 2. Given the set of monitoring servers $M$, the current status of every single regional server and EMF’s processing capability $\eta_i$, find the optimal time instance when each EMF should be suspended or shut down to realize the tradeoff between service availability and scaling cost.

Minimize:

$$\gamma = \sum_{i} \sum_{t} \varphi_i \left[ y_i^t - y_i^{-1} \right]^+ + \sum_{i} \sum_{t} \phi_i p_i^t, \quad i \in M, \quad t \in T$$

Object to:

$$y_i^t \cdot \eta \geq N[r(i)], \quad \forall i \in M, \quad \forall t \in T$$

$$(y_i^t + p_i^t) \cdot H \leq C_i, \quad \forall i \in M, \quad \forall t \in T$$

$$r(i) = \{ j \mid i = \arg \min_{j \in S} l_{i,j} \}$$

Equation (4) defines the objective of our scaling model, where $\left[ y_i^t - y_i^{-1} \right]^+ = \max\{y_i^t - y_i^{-1}, 0\}$ denotes the number of newly created EMFs on server $i$ in time instance $t$. Cost factor $\varphi_i$ and $\phi_i$ represent the cost of creating a new EMF instance and keeping an EMF suspended on server $i$, respectively. By strategically setting the weight of the two cost factors, we can reach the optimal tradeoff between service availability and scaling cost toward a given policy. For example, when the given policy puts extra emphasis on the scaling cost, we should increase the weight of $\varphi_i$, and accordingly the system will reduce the number of EMF creation in order to minimize the objective $\gamma$. Constraint (5) describes that the total process capability of running EMFs on server $i$ must exceed the number of regional servers monitored by EMFs on it, where $r(i)$ represents the set of the regional servers that are monitored by EMFs on server $i$ and $N[r(i)]$ denotes the number of running regional servers in $r(i)$. Constraint (6) implies that the resource that allocated to all running and suspended EMFs on server $i$ must be no more than the server’s resource capability. Constraint (7) indicates that among all monitoring servers, server $i$ has the shortest latency when communicating with regional servers in $r(i)$.

B. ONLINE SCALING SCHEME

To solve problem 2, we propose an online scaling scheme based on classical Ski-Rental algorithm. The objective of classical Ski-Rental problem is to find the trade-off between buying and renting a pair of skis under the condition that the skier has no knowledge of how long he is going to be skiing for in advance [22]. Every day from his starting skiing, a skier needs to determine whether to buy a pair of skis at a lump-sum payment of B dollars or to continuously spend 1 dollar for renting it per day. Obviously, if he skis for less than B days, it is better to rent the skis. Otherwise, to buy the skis at the outset becomes a better option. The main challenge lies in the fact that the skier has no idea ahead of time how many days he is going to ski for. In our scaling model, EMFs on each monitor servers are either marked as “suspended” or “shut down” from the time instance when they become idle. We regard the “suspended” operation as renting and “shutdown” operation as buying in our proposed Ski-Rental model variation. The period during which an EMF keeps idle is deemed as the ski duration. Accordingly, the cost factor $\varphi_i$ and $\phi_i$ defined in equation (4) represents the cost of buying and renting, respectively. Based on such adaptation, we proposed a dynamical lifecycle management scheme. When a running EMF turns to idle status, we first suspend it and then start a counter to record how many time instances the EMF keeps “suspended” for. As soon as the counter exceeds the lifecycle, the EMF will be shut down to release occupied resources. The key of our scaling scheme is to find the optimal lifecycle of each EMF in our proposed model. Based on the competitive analyze result in [22], [23], we give the optimal distribution of lifecycle in equation (8), aiming at minimizing the objective function $\gamma$ defined in equation (4).

$$P[\text{lifecycle} = j] = \begin{cases} 
(1 - \frac{1}{\Delta_i}) \cdot \frac{1}{\Delta_i} \cdot (1 - (1 - \frac{1}{\Delta_i})^{A_i})^{-1}, & 0 \leq j \leq A_i \\
0, & j > A_i
\end{cases}$$

In algorithm II, we give the detailed description of a composite approach comprising EMF placement and scaling simultaneously. It begins with the execution of algorithm I to get the set of monitoring server [line 1], followed by necessary initializations [line 2-8]. Afterwards, the monitoring system iteratively receives the individual state information of every regional servers as input and adjust the number of running EMFs adaptively [line 12-23]. At time instance $t$, we first calculate the minimum number of running EMF on each monitoring server $y_i^t = \text{Ceil}(N[r(i)]/\eta_i)$, where $\text{Ceil}(.)$ rounds the elements to the nearest integer towards infinite [line 12]. Then, we calculate the number of EMFs that need to be newly created for further decision [line 13]. If $[y_i^t - y_i^{-1}]^+ > 0$, we preferentially turn as many “suspended” EMFs to running as required and then, if necessary, create the rest EMFs to provide adequate processing capability. On the contrary, if $[y_i^t - y_i^{-1}]^+ = 0$, we will suspend $Y_i^t - Y_i^{-1}$ EMFs for the sake of resource and energy efficient management. At last, a counter update process is introduced [line 24-32], during which the counter of “suspended” EMFs will be accumulated automatically and the EMF whose counter reach its lifecycle will be shut down to release occupied resources, therefore improving the service availability.

V. SIMULATION AND EXPERIMENTAL RESULTS

In this section, we design two parts of experiments to evaluate the properties of aforementioned approach. We first carry out the performance simulation of our proposed EMF placement and autoscaling approach. In comparison, we consider a recent research on latency-optimal VNF placement at the network edge (latency-optimal solution) [8] and a network and data location-aware application environment placement (NDAP) scheme [11]. In the second part,
we implement a prototype system based on Openstack python API and verify the feasibility on real cloud environment, the realization of some code work has been uploaded to Github https://github.com/Tony910517/VNF_scaling.

**A. SETTINGS**

1) **SIMULATION ENVIRONMENT AND PARAMETER CONFIGURATION**

A computer with 4GB internal storage memory and Intel i7 4790 2.8GHZ CPU is employed for simulation experiments. We use GT-ITM to generate physical network topology [24] and assume that all regional servers have the same resource capability. Moreover, we manually set the following parameters. The latency between regional servers is initialized according to a uniform distribution in the area of [1,100]ms. For parameters of algorithm I, the size of population is set to 40 and the max generation is 50. The selection and mutation possibility are tuned to $p_s = 0.7$ and $p_m = 0.3$ respectively. The maximum indicator is defined as $\text{maxind} = 5$. As for algorithm II, we define the cost factor as $\phi_i = 1$ and $\varphi_i = 7$. The process ability of each EMF $\eta$ is set to 5 and $\text{max_timeslot} = 50$. In sum, Table 2 lists the detailed configuration of simulation parameter.

2) **EXPERIMENTAL ENVIRONMENT**

Our online scaling scheme has been implemented in an Openstack-based cloud platform, consisting of 6 Huawei RH2288H v3 servers and 3 Centec 10G SDN switches. In this experiment, we deem our small cloud platform as a regional server group at the edge and select one server to deploy EMFs. In particular, Openstack Python API is used to realize creating, deleting, suspending and resuming VMs automatically.

**B. NUMERICAL RESULTS**

In figure 2, we compare the CPU time of the three approaches in different network scale. Our proposed approach achieves the lowest CPU time, followed by latency-optimal solution, while NDAP algorithm becomes the most time-consuming one. The reason why NDAP algorithm takes the longest CPU time lies in that it applies an ergodic search strategy among all feasible links between regional servers. Despite that a subgraph isomorphism scheme is introduced to reduce the search space, its CPU time consumption still grows as a polynomial function of the input regional server amount. For latency-optimal solution, it builds an MIP model for latency optimal placement and use a traditional tool, CPLEX to get the optimal solution. The CPLEX toolbox shows higher optimizing efficiency, nevertheless it cannot approach the optimum with a particular gradient, which is precisely how our proposed genetic algorithm-based

**Algorithm 2 Online Scaling Scheme**

1. execute algorithm I to get the set of monitoring servers $M$ in real cloud environment
2. initialize the basic scaling parameter $\phi_i$, $\varphi_i$, $\eta$, $\text{max_timeslot}$, $t = 1$, $y^0_i = 0$ and $p^{0}_i = 0$
3. for $i \in M$
4. obtain the set $r(i)$ according to constraint (7)
5. calculate the upper bound for the number of EMFs that can be created on server $i$ according to constraint (6)
6. end for
7. generate a lifecycle sequence according to the distribution presented in equation (8) and assign a specific lifecycle to all potential EMFs
8. initialize the counter for all potential EMFs
9. while $t \leq \text{max_timeslot}$
10. gather the current status of each regional server through generic hypervisor API
11. for $i \in M$
12. count $N[r(i)]$ and calculate the minum $y^t_i$ according to constraint (5)
13. if $[y^t_i - y^{t-1}_i]^+ > 0$
14. if $p^t_i > [y^t_i - y^{t-1}_i]^+$
15. $p^t_i \leftarrow [y^t_i - y^{t-1}_i]^+$
16. turn on $[y^t_i - y^{t-1}_i]^+$ suspended EMFs
17. else
18. $p^t_i \leftarrow 0$
19. turn on $p^t_i$-suspended EMFs and create $[y^t_i - y^{t-1}_i]^+ - p^t_i$ new EMFs
20. end if
21. else
22. suspend $y^{t-1}_i - y^t_i$ EMFs
23. end if
24. for $j$ in all EMFs
25. if $j$.status == $\text{suspended}$
26. $j$.counter += 1
27. else
28. $j$.counter = 0
29. end if
30. if $j$.counter >= $j$.lifecycle
31. shut down EMF $j$
32. $j$.counter = 0
33. end if
34. end for
35. end for
36. end while

**Algorithm 2 Online Scaling Scheme**

| Parameter | Description | Value |
|-----------|-------------|-------|
| Algorithm I | the number of chromosomes in the population | 40 |
| $\text{maxgen}$ | the maximum evolution generations | 50 |
| $p_s$ | selection possibility | 0.7 |
| $p_m$ | mutation possibility | 0.3 |
| $\text{maxind}$ | maximum indicator | 5 |
| Algorithm II | $\phi_i$ | cost factor of suspending an EMF | 1 |
| | $\varphi_i$ | cost factor of shutting down an EMF | 7 |
| | $\eta$ | process ability of an EMF | 5 |
| | $\text{max_timeslot}$ | the maximum time instance | 50 |

1.Settings
2. Experimental environment
3. Numerical results
meta-heuristic works. Through the comparison, we can draw a conclusion that our genetic algorithm-based meta-heuristic proposed in algorithm I can get fast convergence and dramatically reduce CPU time.

Figure 3 gives the comparison of total latency. As an online algorithm, NDAP algorithm assumes that the NFV MANO cannot get the complete knowledge of the input network size in advance. Towards minimizing the objective function defined in equation (1), it uses a greedy local search algorithm to find the adequate regional servers for each EMF iteratively, and therefore cannot achieve global optimum. In contrast, the latency-optimal solution builds an offline MIP model to solve the proposed placement problem. Taking advantages of the solvers in CPLEX, it can obtain the global optimal solution in the proposed placement problem. In conclusion, the performance of our algorithm can reach close to the global optimal solution, when the network size is relatively small (less than 300 regional servers). As the network size growing larger, our proposed algorithm may be trapped in local optimum.

Figure 4 shows the comparison of scaling cost defined in equation (4) across 50 time instances. In this experiment, we use a large-scale topology of 300 regional servers as input and randomly change the status (running or shut down) of regional servers at each time instance. At the beginning, all the three approaches need to initially place the monitoring system onto the physical network. The process involves a large amount of VM creation, incurring a relatively high scaling cost. From then on, the offline latency-optimal solution keeps a high scaling cost, for the reason that it considers the scaling problem at each time instance as an independent optimization problem. As a result, the latency optimal solution may totally change the placement strategy at each time instance, leading to frequent VM creation and deletion and thus increasing the scaling cost. As for the two online approaches, our proposed scaling scheme has to keep some idle EMF suspended, so the scaling cost is slightly higher than NDAP algorithm at the beginning several time instances. But in the longer term, it is clear that our proposed approach achieves better performance.

Figure 5 gives the information about the proportion of monitoring servers among all regional servers in different network scale. Given the high-level requirement of stability, reliability and timeliness, the available location of monitoring servers is exclusively selected in current cloud data center [7]. Exactly, our proposed placement algorithm provides a way to find the available locations for latency-optimal placement. The simulation result shows that the proportion of the monitoring servers changes from 0.1 to 0.04 when the number of regional servers increase from 50 to 500. Moreover, the fitting curve indicates that the proportion decreases linearly at first and then turns to be stable when the network scale approaches to a certain size.

To verify the feasibility and availability of our proposed scaling scheme, we implement it on our experimental cloud environment. In this experiment, we mainly focus on the
VI. CONCLUSION

In this paper, we studied the important problem about latency-aware placement and autoscaling of monitoring functions in MEC. The main focus of our research lies on two problems. For the first problem, we try to find the strategy that generates a lifecycle randomly and the no policy scheme shuts down an EMF immediately after it has become idle. Figure 6 compares the average execution time across 50 time instances among the three schemes. It is clear that our proposed scheme outperforms the other two, which indicates that our scaling model can reduce the frequency of VM creation.

TABLE 3. Time consumption of some basic operations.

| Operations          | Time consumption |
|---------------------|------------------|
| Create a new EMF    | About 35s        |
| Suspend an EMF      | About 3s         |
| Resume an EMF       | About 3s         |
| Shut down an EMF    | About 5s         |

A comparison of execution time defined as a period that starts from the moment a monitoring server receives a new scaling request at a time instance and ends at the moment when all required EMFs turn to “running” status. Table 3 lists the approximate time consumption of some basic operations. We select one in all six servers in the cloud as the monitoring server and test the performance under different request intensity. Without loss of generality, we randomly generate the number of required EMFs at every time instance according to uniform distribution. By contrast, two internal scaling schemes in Openstack, scaling with random lifecycle and no policy, are involved in the experiment. The random scheme generates a lifecycle randomly and the no policy scheme shuts down an EMF immediately after it has become idle. Figure 6 compares the average execution time across 50 time instances among the three schemes. It is clear that our proposed scheme outperforms the other two, which indicates that our scaling model can reduce the frequency of VM creation.

REFERENCES

[1] 5G White Paper: Next Generation Mobile Networks, North Alliance, Oslo, Norway, Jan. 2015, pp. 1–125.

[2] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, “A survey on edge mobile computing: The communication perspective,” IEEE Commun. Surveys Tuts., vol. 19, no. 4, pp. 2322–2358, 4th Quart., 2017.

[3] F. Giust, X. Costa-Perez, and A. Reznik, “Multi-access edge computing: An overview of ETSI MEC ISG,” IEEE 5G Tech Focus, vol. 1, no. 4, Apr. 2017, p. 4.

[4] A. Ceselli, M. Premoli, and S. Secci, “Mobile edge cloud network design optimization,” IEEE/ACM Trans. Netw., vol. 25, no. 3, pp. 1818–1831, Jun. 2017.

[5] J. Gil Herrera and J. F. Botero, “Resource allocation in NFV: A comprehensive survey,” IEEE Trans. Netw. Service Manag., vol. 13, no. 3, pp. 518–532, Sep. 2016.

[6] J. Ordonez-Lucena, P. Ameigeiras, D. Lopez, J. J. Ramos-Munoz, J. Lorca, and J. Folgueira, “Network slicing for 5G with SDN/NFV: Concepts, architectures, and challenges,” IEEE Commun. Mag., vol. 55, no. 5, pp. 80–87, May 2017.

[7] G. Aceto, A. Botta, W. de Donato, and A. Pescapê, “Cloud: Monitoring: A survey,” Comput. Netw., vol. 57, no. 9, pp. 2093–2115, Jun. 2013.

[8] R. Cziva, C. Anagnostopoulos, and D. P. Pezaros, “Dynamic, latency-optimal VNF placement at the network edge,” in Proc. IEEE Conf. Comput. Commun., Apr. 2018, pp. 693–701.

[9] L. Yala, P. A. Frangoudis, and A. Ksentini, “Latency and availability driven VNF placement in a MEC-NFV environment,” in Proc. IEEE Global Commun. Conf. (GLOBECOM), Dec. 2018, pp. 1–7.

[10] Y. M. Jiang, J. L. Lan, and H. Q. Zhou, “Resource monitoring policy for network virtualization environment,” J. Electron. Inf. Technol., vol. 36, no. 3, pp. 708–714, Mar. 2014.

[11] M. H. Ferdaus, M. Murshed, R. N. Calheiros, and R. Buyya, “An algorithm for network and data-aware placement of multi-tier applications in cloud data centers,” J. Netw. Comput. Appl., vol. 98, pp. 65–83, Nov. 2017.

[12] D. Cho, J. Taheri, A. Y. Zomaya, and L. Wang, “Virtual network function placement: Towards minimizing network latency and lead time,” in Proc. IEEE Int. Conf. Cloud Comput. Technol. Sci. (CloudCom), Dec. 2017, pp. 90–97.

[13] H. Tang, D. Zhou, and D. Chen, “Dynamic network function instance scaling based on traffic forecasting and VNF placement in operator data centers,” IEEE Trans. Parallel Distrib. Syst., vol. 30, no. 3, pp. 530–543, Mar. 2019.

[14] I. Alawe, Y. Hadjadji-Aoul, A. Ksentini, P. Bertin, C. Viho, and D. Darche, “Smart scaling of the 5G core network: An RNN-based approach,” in Proc. IEEE Global Commun. Conf. (GLOBECOM), Dec. 2018, pp. 1–6.

[15] Y. Jia, C. Wu, Z. Li, F. Le, and A. Liu, “Online scaling of NFV service chains across geo-distributed datacenters,” IEEE/ACM Trans. Netw., vol. 26, no. 2, pp. 699–710, Apr. 2018.

[16] T. Subramanya and R. Riggio, “Machine learning-driven scaling and placement of virtual network functions at the network edges,” in Proc. IEEE Conf. Netw. Softwarization (NetSoft), Jun. 2019, pp. 414–422.

[17] R. Gouareb, V. Friderikos, and A.-H. Aghvami, “Virtual network functions routing and placement for edge cloud latency minimization,” IEEE J. Sel. Areas Commun., vol. 36, no. 10, pp. 2346–2357, Oct. 2018.

[18] J. Liu, T. Zhao, S. Zhou, Y. Cheng, and Z. Niu, “CONCERT: A cloud-based architecture for next-generation cellular systems,” IEEE Wireless Commun., vol. 21, no. 6, pp. 14–22, Dec. 2014.

[19] R. Bellman, “Dynamic programming,” Science, vol. 153, no. 3731, pp. 34–37, Feb. 1966.

[20] J. Xu and J. A. B. Fortes, “Multi-objective virtual machine placement in virtualized data center environments,” in Proc. IEEE/ACM Int. Conf. Green Comput. Commun. Int. Conf. Cyber, Phys. Social Comput., Dec. 2010, pp. 179–188.

[21] C. Pham, N. H. Tran, and S. Ren, “Traffic-aware and energy-efficient vnf placement for service chaining: Joint sampling and matching approach,” IEEE Trans. Services Comput., vol. 13, no. 1, pp. 172–185, Jan./Feb. 2017.

[22] A. Khanufer, M. Kodialam, and K. P. N. Puttassamy, “The constrained ski-rental problem and its application to online cloud cost optimization,” in Proc. Proc. IEEE INFOCOM, Apr. 2013, pp. 1492–1500.
[23] X. Wang, C. Wu, F. Le, A. Liu, Z. Li, and F. Lau, “Online VNF scaling in datacenters,” in Proc. IEEE 9th Int. Conf. Cloud Comput. (CLOUD), Jun. 2016, pp. 140–147.

[24] E. W. Zegura, K. L. Calvert, and S. Bhattacharjee, “How to model an internetwork,” in Proc. IEEE Conf. Comput. Commun., Apr. 1996, pp. 594–602.

QUAN YUAN was born in Qingdao, Shandong, China, in 1991. He received the B.S. degree from Nanjing University, in 2014, and the M.E. degree from PLA Strategic Support Force Information and Engineering University, China, in 2017, where he is currently pursuing the Ph.D. degree. His major research interests are in the field of network function virtualization and next-generation mobile communications.

XINSHENG JI was born in Zhumadian, Henan, China, in 1968. He received the B.S. degree from Fudan University, in 1991, and the degree from PLA Strategic Support Force Information and Engineering University, in 1994. He has been a Professor at PLA Strategic Support Force Information and Engineering University, since 2005. His current research interests include next-generation mobile communication and cyber space security.

HONGBO TANG was born in Xiaogan, Hubei, China. He received the B.E. degree from the Huazhong University of Science and Technology, in 1989, and the M.E. degree in engineering from the National University of Defense Technology, in 1995. He has been a Professor at PLA Strategic Support Force Information and Engineering University, China, since 2006. His current interests include next-generation mobile communication and cyber space security.

WEI YOU received the B.S. and M.S. degrees from PLA Strategic Support Force Information and Engineering University, in 2007 and 2010, respectively. Since 2010, he has been a Research Assistant at PLA Strategic Support Force Information and Engineering University. His current interests include network function virtualization and vEPC core networks.