# Enabling Dynamic Autoscaling for NFV in a Non-Standalone Virtual EPC: Design and Analysis

Yi Ren, Tuan Phung-Duc, Jyh-Cheng Chen, *Fellow, IEEE*, and Frank Y. Li

***Abstract*—Network function virtualization (NFV) is a novel concept that enables an architectural transition from dedicated hardware to orchestrated resource and function management. As an integral part of the core network, NFV offers a fine-grained network capability to cellular operators by scaling out or scaling in network resources in an on-demand manner to meet the performance requirements. However, designing an autoscaling algorithm with low operation cost and low latency in non-standalone networks, where legacy network equipment coexists with a virtual evolved packet core (EPC), is a challenging task. In this paper, we propose a dynamic NFV instance autoscaling algorithm that considers the tradeoff between performance and operation cost. Furthermore, we develop an analytical framework to assess the performance of the scheme by modeling the hybrid network as a queueing system that includes both legacy network equipment and NFV instances. The virtualized network function (VNF) instances are powered on or off according to the number of job requests. Numerical results based on extensive simulations validate the correctness of the model and the effectiveness of the algorithm.**

***Index Terms*—EPC, network function virtualization, NFV instance resource allocation, dynamic autoscaling algorithm, modeling and analysis**

## I. Introduction

In parallel with the recent progress in fifth generation (5G) new radio (NR) technologies and worldwide 5G deployment, the 5G system and core network architecture is experiencing a software- and service-centric transformation. The evolution of the mobile core network features a transition from *dedicated hardware*, which is a common practice in early generations, to *orchestrated resources and functions*, which are expected in 5G [1]. As one of the key technologies in the 5G core network, network function virtualization (NFV), which enables decoupling of logical functions from hardware, is transitioning from concept to reality.

Today, services provided by mobile and wireless networks are experiencing exponential growth in terms of both traffic volume and number of devices as well as the diversity of service categories. To satisfy these ever-growing requirements, a traditional solution for service providers is to deploy a sufficient amount of *physical network infrastructure*, based on, e.g., peak-hour traffic. However, this solution suffers from various shortcomings such as increased capital and operational expenditure, inflexibility for service provisioning, and underutilization of resources during non-busy periods. As a novel approach to solving these problems, the concept of NFV emerged [2]. NFV enables operators to virtualize hardware resources and implement network functions through software. It also allows them to make special-purpose network equipment toward software solutions, i.e., virtualized network function (VNF) instances. A VNF instance can run on several virtual machines (VMs), which can be *scaled out* or *scaled in*[1] to adjust the VNF's computing and networking capabilities. The flexibility provided by NFV allows operators to quickly roll out new services and manage their network equipment in a fine-grained and efficient way, minimizing the consumption of both energy and hardware resources.

Following its successful deployment in various sectors, for instance, in cloud computing, NFV, which virtualizes the core network from an evolved packet core (EPC) to a virtual EPC (vEPC), is expected to be deployed in 5G [3]–[5]. However, the transition from EPC to vEPC is an evolutionary phase, featured by the coexistence of equipment from both earlier and next-generation systems. This is because operators often intend to keep legacy cellular network equipment in operation as long as possible to maximize the return on investment (ROI). For 5G NR deployment, indeed both *standalone* (in which 5G NR next generation node B (gNB) is directly connected to the 5G core network) and *non-standalone* (in which both long term evolution (LTE)/evolved node B (eNB) and NR/gNB are connected to the EPC/vEPC) architectures are envisaged. Although standalone 5G is expected to be deployed, most operators currently provide both 4G EPC and non-standalone 5G simultaneously, namely 5G deployment option 3 [6]. They face an immediate need to virtualize EPC. As such, how to configure and utilize network resources automatically and efficiently is an imperative task.

### A. Motivation

For resource allocation in these networks, there is always a tradeoff between the number of allocated resources and the achieved quality of service. With respect to NFV, the operation

Manuscript received January 23, 2022; revised July 16, 2022 and October 30, 2022; accepted December 17, 2022. The associate editor coordinating the review of this article and approving it for publication was P. Lorenz. (Corresponding author: J.-C. Chen.)



Y. Ren is with the School of Computing Science, University of East Anglia (UEA), Norwich, U.K. E-mail: e.ren@uea.ac.uk.

T. Phung-Duc is with the Institute of Systems and Information Engineering, University of Tsukuba, Ibaraki, Japan. E-mail: tuan@sk.tsukuba.ac.jp.

J.-C. Chen is with the Department of Computer Science, National Yang Ming Chiao Tung University (NYCU), Hsinchu, Taiwan. E-mail: jcc@nycu.edu.tw.

F. Y. Li is with the Department of Information and Communication Technology, University of Agder (UiA), 4898 Grimstad, Norway. E-mail: frank.li@uia.no.

[1]VNF scale-out refers to increasing the number of VNF instances, whereas VNF scale-in is an action to remove existing VNF instances such that virtualized hardware resources are released and no longer needed.

cost is diminished by reducing the number of powered-on VNF instances. On the other hand, resource underprovisioning may cause service level agreement (SLA) violations [7]. Accordingly, VNF instance *autoscaling*, which can decrease operation costs while satisfying the demand for VNF services, has emerged as a promising approach. In principle, the NFV performance is improved by *scaling out* VNF instances, while the operation cost is reduced by *scaling in* VNF instances. However, it is not a trivial task to design suitable schemes to allocate VNF instances adaptively while fulfilling the demands of service requirements.

To facilitate VNF instance autoscaling, it is important to predict the workload of VMs. In the literature, various methods for VM load prediction and autoscaling exist, from the moving average to machine learning-based methods [8]. However, these studies either ignore the VM setup time or consider merely the virtualized resource itself without integrating it with legacy resources. We argue, therefore, that both the *VM setup time* and *legacy equipment capacity* should be considered for VNF instance autoscaling in 5G networks when coexisting with 4G systems.

More specifically, although a scale-out request can be initiated immediately after prediction, a VNF instance may not be readily available. To start an instance in Microsoft Azure, for example, the lag time could be 10 minutes or longer, with this delay varying from time to time [9]. If the lag time is not taken into consideration, the just-started instance may be too late to serve the VNF. On the other hand, the capacity of legacy network equipment may have a significant impact on autoscaling in these hybrid networks. Assuming that the capacity of a piece of legacy network equipment is equal to that of one VNF, scaling out from one VNF instance to two VNF instances would increase the capacity by $100\%$. However, if the capacity of a piece of legacy network equipment is equal to that of 100 VNFs, its capacity grows by less than $1\%$ when adding one more VNF instance. Cloud-based autoscaling schemes usually ignore this problem, which is referred to as a *non-constant* issue [10].

### B. Contributions

In this paper, we propose a dynamic autoscaling scheme to resolve these issues. In the proposed scheme, we consider that legacy 4G network equipment is always powered on as a block, while virtualized resources (VNF instances) are added to or removed from the system dynamically in an on-demand manner. The approach proposed in this paper offers network operators guidelines for optimal VNF autoscaling design systematically based on their management policies. Briefly, the main contributions of this paper are as follows.

- A dynamic autoscaling scheme for VNF instance allocation is proposed, in which the cost-performance tradeoff is quantified as an *operation cost metric* and a *performance metric*. According to our scheme, a number of VNF instances are scaled in and out depending on the number of jobs in the system. The scheme also deals with how to configure a suitable $k$, which is the number of VNF instances, while considering the cost-performance tradeoff.

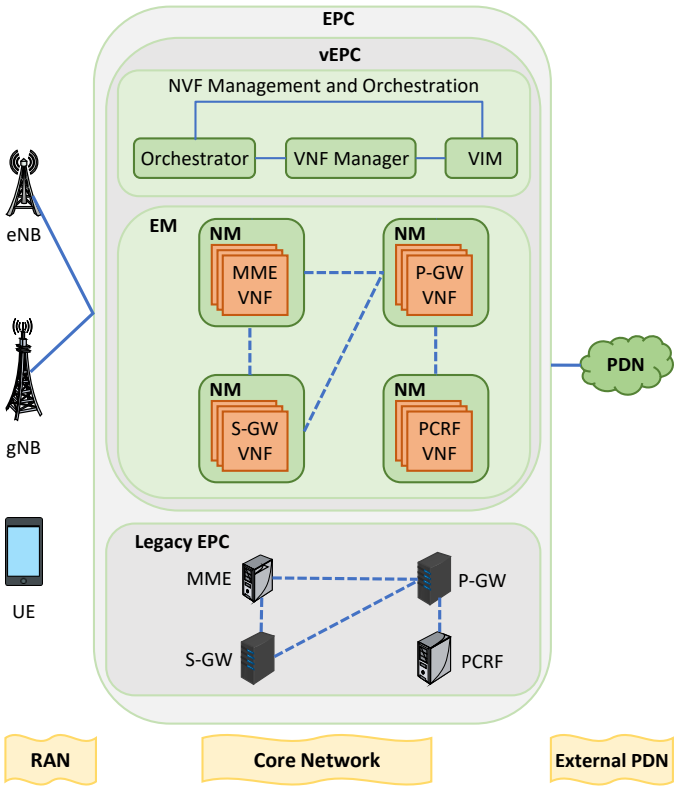



Fig. 1: Illustration of an NFV-enabled architecture in 4G/5G.

- To evaluate the performance of the proposed scheme, we develop an analytical model and derive closed-form expressions for the performance metrics. The performance is further validated based on extensive discrete-event simulations.
- Moreover, we develop a recursive algorithm that reduces the computational complexity from $O(k^3 \times K^3)$ to $O(k \times K)$, where $K$ is the total capacity of the system.

The remainder of this paper is organized as follows. In Section II, we briefly introduce the concept of EPC/vEPC and NFV. Section III reviews the related work and points out the uniqueness of this work. In Sections IV and V, we first present the proposed VNF instance autoscaling scheme and then develop an analytical model to evaluate the performance of the scheme. Afterwards, the numerical results are illustrated in Section VI. Finally, Section VII concludes the paper.

## II. Preliminaries on the EPC/vEPC and NFV

A cellular network is typically composed of one or multiple radio access networks (RANs) and a core network (CN), as shown in Fig. 1 [11]. In this section, we explain briefly the EPC and vEPC when NFV is deployed.

### A. Legacy EPC

In LTE, user equipment (UE) connects to an EPC through an eNB. The basic functions of an EPC include the serving gateway (S-GW), packet data network (PDN) gateway (P-GW), mobility management entity (MME), and policy and charging rules function (PCRF). For more details about these functions, refer to [12].

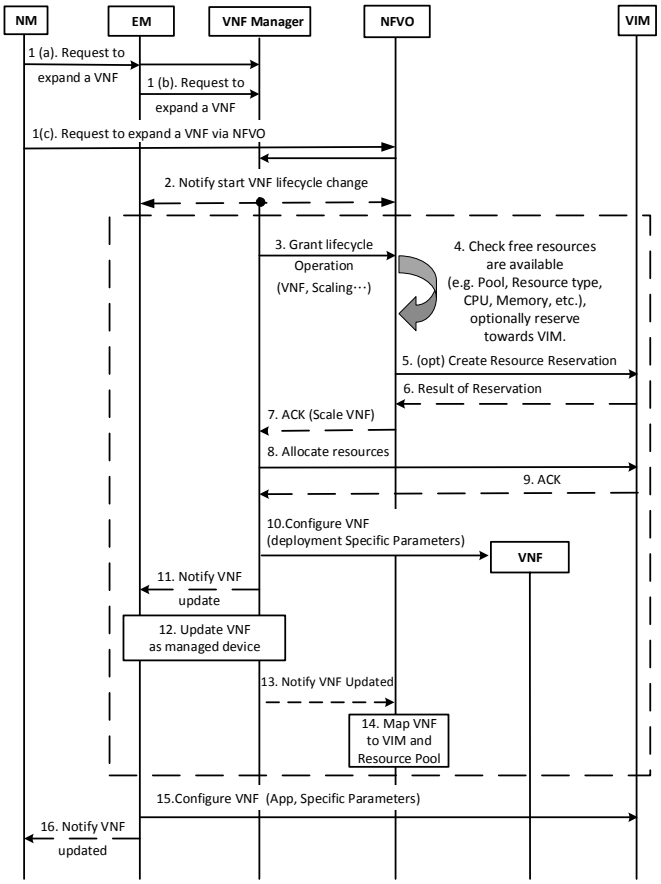



Fig. 2: Illustration of a VNF instance expansion procedure triggered by an NM/EM [13].

### B. vEPC

Conceptually, the vEPC can be divided into two main components: NFV management and orchestration [3], and an element manager (EM). Both components are parts of the 3rd generation partnership project (3GPP) management reference model, which is based on the European telecommunications standardization institute (ETSI) NFV specification [13].

As shown in Fig. 1, a NFV management and orchestration component consists of an orchestrator, a VNF manager, and a virtualized infrastructure manager (VIM). It controls the life cycles of VNFs and decides whether a VNF should be scaled in or scaled out. Additionally, it manages both hardware and software resources to support VNFs. A VNF may have multiple VNF instances, inside which there are many VMs.

In the vEPC, each network element (NE) in the legacy EPC, such as the S-GW, P-GW, MME, and PCRF, is virtualized as a VNF. As shown in Fig. 1, a network manager (NM) provides end-user functions for the network management of each NE. An EM is responsible for the management of a set of NMs.

### C. VNF Instance Scaling Procedures

A VNF manager allocates resources via two scaling procedures: VNF instance expansion, which is a scale-out procedure to add resources to a VNF, and VNF instance contraction, which is a scale-in procedure to release resources from a VNF.

Fig. 2 illustrates the VNF instance expansion procedure, with its steps briefly explained below. A similar procedure applies to VNF instance contraction. Refer to [13] for more details regarding these two procedures.

- Step 1: An NM/EM (via an NFV orchestrator, NFVO) sends a capability expansion request to the VNF manager (see 1(a), 1(b), and 1(c)).
- Step 2: The VNF manager sends a life cycle change notification to the EM and NFVO, indicating the start of the scaling operation.
- Steps 3-14: The VNF manager sends a request to the NFVO for VNF expansion. The NFVO then checks whether or not enough idle resources are available and, if yes, sends acknowledgement (ACK)/negative ACK (NACK) to the VNF manager for VNF expansion.
- Step 15: The EM configures the VNF with application-specific parameters.
- Step 16: The EM notifies the NM of the newly updated and configured capacity.

## III. Related Work

As presented above, autoscaling enables NFV to scale-out or scale-in VM resources in an on-demand manner. As such, the knowledge on the VM load is of vital importance. In the literature, various approaches have been proposed to predict the VM load to boot VMs before these VMs, which are under operation, become overloaded. These approaches can be categorized into those based on the moving average, including the exponential weighted moving average (EMA) [8], [14], autoregressive moving average (ARMA) [15], [16], and autoregressive integrated moving average (ARIMA) [17], [18], those based on machine learning [19], those based on Markov models [20], [21], and those based on queueing models [22]–[30]. In the following, we summarize each category briefly.

**Moving average based:** The basic idea of this category is that the most recent input data within a moving window are used to predict the next set of input data. In [8], the authors proposed an EMA-based scheme to predict CPU load. Their scheme was implemented in a domain name system (DNS) server, and the evaluation results showed that the capacities of the servers were well utilized. [14] introduced a novel prediction-based dynamic resource allocation algorithm to scale video transcoding services in a cloud. A two-step procedure was adopted to predict the load, resulting in a reduced number of required VMs.

ARMA adds autoregression (AR) to the moving average. A resource allocation algorithm based on the ARMA model was reported in [15], where the empirical results showed significant benefits both to cloud users and service providers. In [16], the authors developed a load forecasting model based on the ARMA, which achieved an approximately $6\%$ prediction error rate and saved up to $44\%$ of hardware resources compared with a random content-based distribution policy.

Unlike the ARMA and ARIMA, which differentiate the input data, [17] proposed a predictive and elastic cloud bandwidth autoscaling scheme considering multiple data centers. The work of [31] took VM migration overhead into account

TABLE I: Comparison of related work and ours

| Factors | [22] | [23] | [24] | [25] | [26] | [27] | [28] | [29] | [30] | Ours |
|---|---|---|---|---|---|---|---|---|---|---|
| Setup time | ✓ | ✓ | ✓ | | ✓ | ✓ | | ✓ | ✓ | ✓ |
| Independent setup time | | | ✓ | ✓ | ✓ | ✓ | | ✓ | ✓ | ✓ |
| Multiple sub-block | | | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Legacy network capacity | ✓ | ✓ | ✓ | ✓ | | | ✓ | | | ✓ |

when designing their autoscaling scheme. Furthermore, [32] dealt with the dynamic workload fluctuation of each VM and resource conflict handling. Another ARIMA-based workload prediction scheme was proposed in [18], which uses real traces of requests to web servers. The results showed that the model could achieve up to $91\%$ accuracy.

**Machine learning-based** approaches are becoming popular for the design of cloud autoscaling algorithms [19], [33]. [33] implemented a Bayesian network-based cloud autoscaling algorithm. In [19], the authors evaluated three machine learning approaches, i.e., linear regression, a neural network, and the support vector machine (SVM). Their results showed that the SVM-based scheme outperforms the other two.

**Markov model-based** approaches have also been widely used in cloud autoscaling algorithms [20], [21]. The authors of [20] developed CloudScale, an automatic elastic resource scaling system for multiple cloud service providers, which reduces the total energy consumption and the workload energy consumption by $8\% - 10\%$ and $39\% - 71\%$, respectively, with little effect on the application performance. Moreover, [21] proposed a novel multiple time series approach based on a hidden Markov model (HMM). The technique well characterizes the temporal correlations in the discovered VM clusters to predict variations in workload patterns.

**Queueing model-based** approaches were studied in [22]–[30]. In their developed models, the authors considered the setup time both with [22] and without [23] defections. After the setup time, all of the cloud servers in a block are assumed to be active concurrently. However, this is not realistic considering the inhomogeneity of servers. In [24], this assumption was relaxed such that, although all servers in a block are activated at the same time, each server has an independent setup time. In these three schemes, no individual and dynamic scale-out or scale-in is enabled. In [25], an enhanced scheme allowing sub-block activation was developed. This relaxation was significant, bringing these schemes closer to reality. Nevertheless, the setup time was ignored in [25] due to the modeling and analysis complexity. In [26], [29], a queueing model with setup time was proposed, where each server is shut down if it has no job to do and is started up if a job awaits. However, the legacy network capacity was not considered therein.

In summary, the mechanisms proposed in the aforementioned studies were targeted at *NFV applications in cloud computing*. These mechanisms either ignore the VM setup time or deal only with virtualized resources without considering resources from legacy systems. Although [27], [28], [30] derived optimal properties and presented a precise analysis for a queueing model, [28] ignores setup time and [27], [30] did not consider legacy servers. Therefore, the existing mechanisms and models are not applicable to non-standalone 5G networks, where eNBs and gNBs coexist. In contrast, a salient feature of this work is that we regard the buffer size to be finite, allow a certain number of always-on servers, which correspond to the legacy network capacity of non-standalone 5G networks, and consider the VM setup time in our model. In Table I, we give a qualitative comparison regarding various aspects that have been considered in our model versus the ones that were considered in a few reference studies.

## IV. Enabling Dynamic VNF Instance Autoscaling in the vEPC: The Proposed Algorithm

In this section, we first present the network scenario and autoscaling principle. Then, we propose an autoscaling algorithm based on the considered network scenario.

### A. Network Scenario and Autoscaling Principle

Consider a 5G EPC composed of a vEPC and legacy EPC, as shown in Fig. 1. The legacy EPC contains network entities such as the MME and PCRF, which lie on-site from early 4G deployment. With the support of the EM and NFV management and orchestration components, the vEPC, which consists of a number of VNF instances, offers fine-grained network capabilities to its collocated legacy network entity in an on-demand manner.

In our study, the legacy network equipment is considered to always be on, whereas VNF instances are added (or removed) according to the number of jobs awaiting in the system. It is worth mentioning that a VNF instance needs some time to start its operation upon receiving a request. During this setup time, the VNF instance consumes a certain amount of power but cannot provide services.

### B. Cost Function-based Autoscaling Design

As mentioned in Subsection I-A, there is a tradeoff between the number of allocated resources (i.e., VNF instances, which reflect the operation cost) and network performance. Accordingly, a design goal when developing an appropriate autoscaling algorithm for VNF instance allocation is to minimize operation costs while providing a satisfactory level of network performance.

For VNF instance resource allocation, an essential question to answer is how many VNF instances should be scaled out or scaled in such that the cost is minimized while the required level of performance is satisfied. In this study, the

cost function, denoted by $C$, is evaluated by two metrics, i.e., the average response time in the queue per request, $W_q$, and the average cost of VNF instances, $S$. The cost function-based resource allocation[2] can be mathematically formulated as follows:

$$\begin{aligned} \text{minimize} \quad & C = w_1 W_q + w_2 S \ , \\ \text{subject to} \quad & 0 < W_q < W_q' \ , \end{aligned} \tag{1}$$

where $W_q'$ is the upper bound of $W_q$, which can be determined by a cellular operator according to the service requirements. The two coefficients $w_1$ and $w_2$ are the weighting scalars for $W_q$ and $S$, respectively. Increasing $w_1$ (or $w_2$) indicates that an operator pays more attention to $W_q$ (or $S$). In this study, we do not suggest any concrete values for $w_1$ or $w_2$ since such a value should be determined by the cellular operator when taking its service and management policies into consideration. Instead, an algorithm for finding the optimal solution when $w_1$ and $w_2$ are specified is introduced in Subsection V-B. The algorithm returns an optimal number of VNF instances $k_{op}$. Then, the number of VNF instances can be adjusted accordingly.

### C. VNF Instance Autoscaling Algorithm

Consider a non-standalone 5G EPC in which there are $k$ VNF instances that can be switched on or off. We introduce two actions, *up* and *down*, denoted by $U_i$ and $D_i$, where $i = 1, 2, \cdots, k$, respectively, to represent the status change of VNF instances (the process of selecting an optimal $k$ is detailed in Section IV-C).

- $U_i-$ *Switch-on the $i$-th VNF instance:* If the $i$-th VNF instance is turned off and the number of requests in the system increases from $U_i - 1$ to $U_i$, the VNF instance needs to be powered on to meet the service requirement. During the setup time, a VNF instance cannot serve any user request, but it still consumes power (and leads to a monetary cost).
- $D_i-$ *Switch-off the $i$-th VNF instance:* If the $i$-th VNF instance is in operation and the number of requests in the system drops from $D_i + 1$ to $D_i$, the VNF instance is powered off instantaneously.

In **Algorithm 1**, we present a sketch of the proposed algorithm for dynamic VNF instance autoscaling.

## V. Performance Analysis and Cost Function Optimization

In this section, we first develop a Markov model to evaluate the performance of the proposed autoscaling scheme. Then, another algorithm to find an optimal number of VNF instances is presented. The notations adopted in our analysis are summarized in Table II.

### A. Modeling Autoscaling: An M/M/N/K/Setup Queue

As can be observed in Fig. 1, the total capacity of the studied system is the summed-up capacity from both the vEPC and legacy EPC. We consider a non-standalone 5G EPC in which there are $k$ VNF instances that can be switched on or off. Assume that the capacity of the legacy network entity is equivalent to $n_0$ VNF instances. The total capacity of the system, $N$, in terms of the number of VNF instances is then $N = n_0 + k$ since $k$ denotes the number of VNF instances in the vEPC. Based on our autoscaling algorithm, $N$ can be adjusted adaptively depending on $k$. Accordingly, we have $n_1 = n_0 + 1$ and $n_i = n_{i-1} + 1$, where $i = 1, 2, \cdots k$ and $n_k = N$. Furthermore, $U_i = n_i$, $D_i = n_{i-1}$, $i = 1, 2, \cdots, k$.

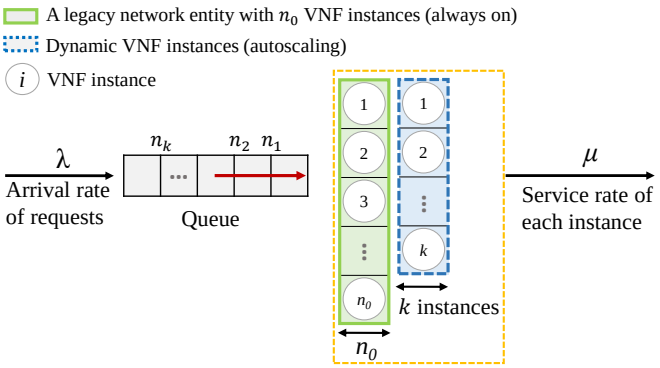



Fig. 3: A simplified queueing model for our system.

*1) Description of the model:* Fig. 3 illustrates a simplified queueing model developed in our system, which is an $M/M/N/K$/Setup queue. For user traffic or a job request injected into the system, we model traffic at the flow or service level instead of at the packet level to capture the dynamics related to the arrival and departure of services such as the flow duration or number of active flows. As such, it is reasonable to assume job request arrivals follow a Poisson process with a rate $\lambda$. Furthermore, a VNF instance accepts one job at a time with a service rate $\mu$. In the system, there is a finite first-come-first-served (FCFS) queue for those requests that have to wait for processing.

In our queueing model, there are $N = n_0 + k$ servers, which are divided into two blocks: a fixed block and a dynamic block. The $n_0$ servers in the fixed block, which represent the capacity of legacy equipment, are always on. The dynamic block denotes VNF instances in which the $k$ servers are in either a BUSY, OFF, or SETUP state. The queue has a total capacity $K$, i.e., the maximum number of jobs that can be accommodated in the system is $K$. In the dynamic block, a server is turned off immediately if it has no job to serve. Upon the arrival of a job, an OFF server is turned on if the job is placed in the buffer. However, a server needs some time to set up to serve a waiting job. We further assume that the setup time follows an exponential distribution with a mean value of $1/\alpha$. Let $j$ denote the number of jobs in the system and $i$ denote the number of active servers in the dynamic block. The number of servers in the SETUP state is $\min(j - n_i, N - n_i)$.

*2) Steady-state distribution calculation:* In the following, we present a recursive algorithm to calculate the joint stationary distribution. Let $C(t)$ and $L(t)$ denote the number of active servers in the dynamic block and the number of jobs in the whole system, respectively. It is clear that $\{X(t) =$

[2]Other types of cost function may apply. An instance of one such function that includes more parameters can be found in the Appendix.

TABLE II: List of Notations

| Notation | Description |
|---|---|
| $\lambda$ | job arrival rate |
| $\mu$ | service rate for each VNF instance |
| $k$ | the number of VNF instances |
| $k_{op}$ | the optimal number of VNF instances |
| $\alpha$ | setup rate for each VNF instance |
| $N$ | the total capacity of the system |
| $n_0$ | the capacity of the legacy network entity |
| $K$ | the maximum number of jobs that can be accommodated in the system |
| $W$ | average response time per job |
| $W_q$ | average response time in the queue per job |
| $W_q'$ | the upper bound of $W_q$ |
| $\overline{W}_q$ | the maximum value of $W_q$ in the system |
| $S$ | average VNF cost |
| $\overline{S}$ | the maximum value of $S$ in the system |
| $P_b$ | blocking probability |
| $L$ | mean number of jobs in the system |
| $\delta$ | the ratio of weighting factors, $w1/w2$ |
| $w_1$ | weighting factor for $W_q$ |
| $w_2$ | weighting factor for $S$ |
| $w_3$ | weighting factor for $P_b$ |
| $w_4$ | weighting factor for $W$ |
| $w_5$ | weighting factor for $L$ |

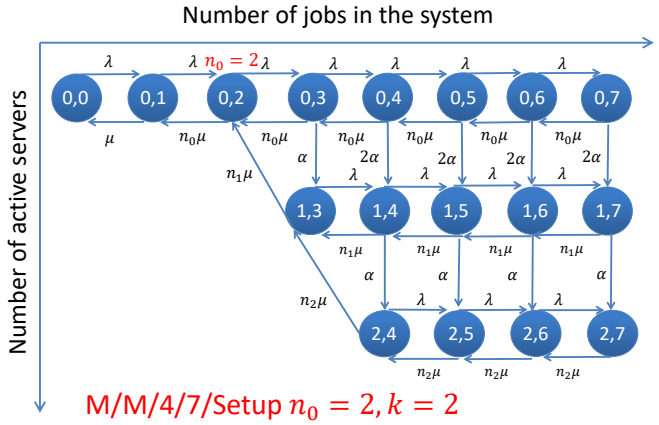



Fig. 4: Illustration of a state and state transition with $N = 4$, $n_0 = 2$, $k = 2$, and $K = 7$.

$(C(t), L(t)); t \geq 0\}$ forms a Markov chain in the state space:

$$\begin{aligned}\mathcal{S} =&\{(i,j); 1 \leq i \leq k, j = n_i, n_i+1, \ldots, K-1, K\} \\ &\cup \{(0,j); j = 0, 1, \ldots, K-1, K\}. \end{aligned} \tag{2}$$

As an example, we configure $N = 4$, $n_0 = 2$, $k = 2$, and $K = 7$. Fig. 4 depicts the states and transitions among the states under this configuration. Let

$$\pi_{i,j} = \lim_{t\to\infty} \mathrm{P}(C(t) = i, L(t) = j), (i,j) \in \mathcal{S} \tag{3}$$

denote the joint stationary distribution of $\{X(t)\}$. Here, we derive a recursion to calculate the joint stationary distribution $\pi_{i,j}$, $(i,j) \in \mathcal{S}$.

Let us first consider a recursion for $\pi_{0,j}$ $(j = 0, 1, \ldots, K)$. The balance equations for states with $i = 0$ are as follows:

$$\lambda\pi_{0,j-1} = j\mu\pi_{0,j}, \quad \text{for } j = 0, 1, \ldots, n_0, \tag{4}$$

$$\begin{aligned}\lambda\pi_{0,j-1} + n_0\mu\pi_{0,j+1} =& (\lambda + n_0\mu \\ &+ \min(j - n_0, N - n_0)\alpha)\pi_{0,j}, \\ &\text{for } j = n_0, n_0+1, \ldots, K-1, \end{aligned} \tag{5}$$

$$\lambda\pi_{0,K-1} = (n_0\mu + (N - n_0)\alpha)\pi_{0,K}, \tag{6}$$

leading to

$$\pi_{0,j} = b_j^{(0)}\pi_{0,j-1}, \qquad j = 1, 2, \ldots, K. \tag{7}$$

The sequence $\{b_j^{(0)}; j = 1, 2, \ldots, K\}$ is given as follows:

$$b_j^{(0)} = \frac{\lambda}{j\mu}, \qquad j = 1, 2, \ldots, n_0, \tag{8}$$

and

$$\begin{aligned} b_j^{(0)} = \frac{\lambda}{\lambda + n_0\mu + \min(j - n_0, N - n_0)\alpha - n_0\mu b_{j+1}^{(0)}}, \\ j = K-1, K-2, \ldots, n_0+1, \end{aligned} \tag{9}$$

where

$$b_K^{(0)} = \frac{\lambda}{n_0\mu + (N - n_0)\alpha}.$$

Furthermore, it should be noted that $\pi_{1,n_1}$ is calculated using the local balance equation in and out of the set $\{(0,j); j = 0, 1, \ldots, K\}$ as follows:

$$n_1\mu\pi_{1,n_1} = \sum_{j=n_1}^{K} \min(j, N - n_0)\alpha\pi_{0,j}. \tag{10}$$

*Remark*. Thus far, we have expressed $\pi_{0,j}$ $(j = 1, 2, \ldots, K)$ and $\pi_{1,n_1}$ in terms of $\pi_{0,0}$.

Now, let us consider the general case for $\pi_{i,j}$ where $1 \leq i \leq k-1$. Lemma 1 demonstrates that for a fixed $i = 1, 2, \ldots, k-1$, $\pi_{i,j}$ can be expressed in terms of $\pi_{i,j-1}$ $(j = n_i + 1, n_i + 2, \ldots, K)$. As a result, $\pi_{i,j}$ $(j = n_i + 1, n_i + 2, \ldots, K)$ is expressed in terms of $\pi_{i,n_i}$.

**Lemma 1.** *We have*

$$\pi_{i,j} = a_j^{(i)} + b_j^{(i)}\pi_{i,j-1}, \quad j = n_i + 1, n_i + 2, \ldots, K-1, K, \tag{11}$$

*where*

$$a_j^{(i)} = \frac{n_i\mu a_{j+1}^{(i)} + \min(N - n_{i-1}, j - n_{i-1})\alpha\pi_{i-1,j}}{\lambda + \min(N - n_i, j - n_i)\alpha + n_i\mu - n_i\mu b_{j+1}^{(i)}}, \tag{12}$$

$$b_j^{(i)} = \frac{\lambda}{\lambda + \min(N - n_i, j - n_i)\alpha + n_i\mu - n_i\mu b_{j+1}^{(i)}}, \tag{13}$$

*and*

$$a_K^{(i)} = \frac{(N - n_{i-1})\alpha\pi_{i-1,K}}{(N - n_i)\alpha + n_i\mu}, \quad b_K^{(i)} = \frac{\lambda}{(N - n_i)\alpha + n_i\mu}. \tag{14}$$

*Proof.* The balance equation for state $(i, K)$ is given as

follows:

$$((N-n_i)\alpha+n_i\mu)\pi_{i,K}=\lambda\pi_{i,K-1}+(N-n_{i-1})\alpha\pi_{i-1,K}. \tag{15}$$

On this basis, Lemma 1 is true for $j=K$. Assume that

$$\pi_{i,j+1}=a_{j+1}^{(i)}+b_{j+1}^{(i)}\pi_{i,j},\ \ j=n_i+1,n_i+2,\ldots,K-1. \tag{16}$$

Substituting this expression into the next balance equation,

$$\begin{aligned}(\lambda+\min(N-n_i,j-n_i)\alpha+n_i\mu)\pi_{i,j}&=\lambda\pi_{i,j-1}\\ +n_i\mu\pi_{i,j+1}+\min(N-n_{i-1},j-n_{i-1})\alpha\pi_{i-1,j},&\\ j=K-1,K-2,\ldots,n_i+1,&\end{aligned} \tag{17}$$

we obtain

$$\pi_{i,j}=a_j^{(i)}+b_j^{(i)}\pi_{i,j-1}. \tag{18}$$

□

*Remark.* In Corollary 1 below, we show that $a_j^{(i)}$ and $b_j^{(i)}$ are positive. Thus, the recursive algorithm is stable because it manipulates only positive numbers. Furthermore, we prove that $b_j^{(i)}$ is bounded from above. Although we cannot obtain an explicit upper bound for $a_j^{(i)}$, we observe that $a_j^{(i)}$ is not so large according to numerical experiments. One reason may be that the coefficient of $a_{j+1}^{(i)}$ in (19) is less than 1. These upper bounds are the rationale for the stability of our recursive algorithm since we deal with numbers that are not too large so that overflow is avoided.

**Corollary 1.** *The following bound holds.*

$$0<a_j^{(i)}<\frac{n_i\mu a_{j+1}^{(i)}+\min(N-n_{i-1},j-n_{i-1})\alpha\pi_{i-1,j}}{n_i\mu+\min(j-i,N-n_i)\alpha}, \tag{19}$$

$$0<b_j^{(i)}<\frac{\lambda}{n_i\mu+\min(j-i,N-n_i)\alpha}, \tag{20}$$

*for* $j=n_i+1,n_i+2,\ldots,K; i=1,2,\ldots,k-1$.

*Proof.* We prove this corollary through mathematical induction. It is clear that Corollary 1 is true for $j=K$. Assume that Corollary 1 is also true for $j+1$, i.e.,

$$a_{j+1}^{(i)}>0,\qquad 0<b_{j+1}^{(i)}<\frac{\lambda}{n_i\mu+\min(j+1-n_i,N-n_i)\alpha}, \tag{21}$$

for $j=n_i+1,n_i+2,\ldots,K-1, i=1,2,\ldots,k-1$. It then follows from the second inequality that $n_i\mu b_{j+1}^{(i)}<\lambda$. This result, together with (12) and (13), yields the claimed bound. □

It should be noted that $\pi_{i+1,n_{i+1}}$ is calculated using the following local balance equation in and out of the set of states

$$\{(k,j)\in\mathcal{S};k=0,1,\ldots,i\} \tag{22}$$

as follows:

$$n_{i+1}\mu\pi_{i+1,n_{i+1}}=\sum_{j=n_i+1}^{K}\min(j-n_i,N-n_i)\alpha\pi_{i,j}. \tag{23}$$

*Remark.* Until now, we have expressed $\pi_{i,j}$ $(i=0,1,\ldots,k-1, j=n_i,n_i+1,\ldots,K)$ and $\pi_{k,n_k}$ in terms of $\pi_{0,0}$.

Next, let us continue with the case in which $i=k$. The balance equation for state $(k,j)$ $(j=k,k+1,\ldots,K)$ leads to Lemma 2.

**Lemma 2.** *The following equation holds.*

$$\pi_{k,j}=a_j^{(k)}+b_j^{(k)}\pi_{k,j-1},\ \ j=n_k+1,n_k+2,\ldots,K, \tag{24}$$

*where*

$$\begin{aligned}a_j^{(k)}&=\frac{n_k\mu a_{j+1}^{(k)}+(N-n_{k-1})\alpha\pi_{k-1,j}}{\lambda+n_k\mu-n_k\mu b_{j+1}^{(k)}},\\ &j=K-1,K-2,\ldots,n_k+1,\end{aligned} \tag{25}$$

$$\begin{aligned}b_j^{(k)}&=\frac{\lambda}{\lambda+n_k\mu-n_k\mu b_{j+1}^{(k)}},\\ &j=K-1,K-2,\ldots,n_k+1,\end{aligned} \tag{26}$$

*and*

$$a_K^{(k)}=\frac{\alpha\pi_{k-1,K}}{n_k\mu},\qquad b_K^{(k)}=\frac{\lambda}{n_k\mu}. \tag{27}$$

*Proof.* The global balance equation in state $(k,K)$ is given by

$$n_k\mu\pi_{k,K}=(N-n_{k-1})\alpha\pi_{k-1,K}+\lambda\pi_{k,K-1}, \tag{28}$$

leading to

$$\pi_{k,K}=a_K^{(k)}+b_K^{(k)}\pi_{k,K-1}. \tag{29}$$

Assume that $\pi_{k,j+1}=a_{j+1}^{(k)}+b_{j+1}^{(k)}\pi_{k,j}$. Following this formula, the global balance equation in state $(k,j)$ becomes

$$\begin{aligned}(\lambda+n_k\mu)\pi_{k,j}=&\lambda\pi_{k,j-1}+n_k\mu\pi_{n_k,j+1}\\ &+(N-n_{k-1})\alpha\pi_{k-1,j},\\ j=&n_k+1,n_k+2,\ldots,K-1,\end{aligned} \tag{30}$$

where $\pi_{k,j}=a_j^{(k)}+b_j^{(k)}\pi_{k,j-1}$ for $j=n_k+1,n_k+2,\ldots,K$. □

**Corollary 2.** *The following bound holds.*

$$\begin{aligned}a_j^{(k)}>0,\qquad 0<b_j^{(k)}<\frac{\lambda}{n_k\mu},&\\ j=n_k+1,n_k+2,\ldots,K-1.&\end{aligned} \tag{31}$$

*Proof.* This proof is also carried out by using mathematical induction. It is clear that Corollary 2 is true for $j=K$. Assume that Corollary 2 is true for $j+1$, i.e.,

$$\begin{aligned}a_{j+1}^{(k)}>0,\qquad 0<b_{j+1}^{(k)}<\frac{\lambda}{n_k\mu},&\\ j=n_k+1,n_k+2,\ldots,K-1.&\end{aligned} \tag{32}$$

It follows from the second inequality that $n_k\mu b_{j+1}^{(k)}<\lambda$. Together with (25) and (26), this inequality yields the claimed bound. □

As a result, we have expressed all the probabilities $\pi_{i,j}$ $((i,j)\in\mathcal{S})$ in terms of $\pi_{0,0}$, which are uniquely determined

by the normalization condition as follows:

$$\sum_{(i,j)\in\mathcal{S}} \pi_{i,j} = 1. \tag{33}$$

*Remark.* In summary, we are able to calculate all probabilities $\pi_{i,j}$ ($(i,j) \in \mathcal{S}$) in the following order. First, we set $\pi_{0,0} = 1$. We then calculate all the probabilities $\pi_{0,j}$ ($j = 1, 2, \ldots, K$) using (7). Next, $\pi_{1,n_1}$ is calculated using (10). After that, we apply Lemma 1 and (23) repeatedly for $i = 1, 2, \ldots, k-1$. At this point, $\pi_{i,j}$ (with $i = 0, 1, \ldots, k-1$) and $\pi_{k,n_k}$ are obtained. Furthermore, we use Lemma 2 to obtain $\pi_{k,j}$ for $j = n_k+1, n_k+2, \ldots, K$. Finally, we divide all $\pi_{i,j}$ ($(i,j) \in \mathcal{S}$) by $\sum_{(i,j)\in\mathcal{S}} \pi_{i,j}$ to obtain the stationary distribution.

*3) Performance analysis:* Based on the obtained steady-state probabilities, we perform the following performance analysis. Let $L$ denote the mean number of jobs in the system. We have:

$$L = \sum_{(i,j)\in\mathcal{S}} \pi_{i,j} j = \sum_{i=0}^{n_0-1} \pi_{0,j} j + \sum_{i=0}^{k} \sum_{j=n_i}^{K} \pi_{i,j} j. \tag{34}$$

Following Little's law, $W$ is expressed as follows:

$$W = \frac{\mathrm{E}[L]}{\lambda(1-P_b)} = \frac{\sum_{i=0}^{n_0-1} \pi_{0,j} j + \sum_{i=0}^{k} \sum_{j=n_i}^{K} \pi_{i,j} j}{\lambda(1-\sum_{i=0}^{k} \pi_{i,K})}. \tag{35}$$

Therefore, we obtain

$$W_q = W - \frac{1}{\mu}. \tag{36}$$

Denote by $P_b$ the blocking probability. It is clear that

$$P_b = \sum_{i=0}^{k} \pi_{i,K}. \tag{37}$$

Furthermore, the mean number of VNF instances is given by

$$S = \sum_{(i,j)\in\mathcal{S}} \pi_{i,j}(n_i - n_0) + \sum_{i=0}^{k} \sum_{j=n_i}^{K} \pi_{i,j} \min(j-n_i, N-n_i), \tag{38}$$

where the first term is the number of VNF instances that are already active and the second term is the mean number of VNF instances in the SETUP state.

*4) Summary of the derivation:* In this subsection, we have developed a mathematical model and derived closed-form expressions for the two metrics $W_q$ and $S$, which are the main components of the cost function (1). The expressions for $W_q$ and $S$ are shown in (36) and (38), respectively. Based on these closed-form expressions, one can further find an optimal $\tau \in \{k, n_0, \mu, K, \alpha\}$ to balance the cost function $C$, given that the other parameters and $\lambda$ are known since $W_q$ and $S$ are functions of those parameters. We formulate this optimization problem as follows:

$$\begin{aligned} \arg\min_{\tau} \quad & C = w_1 W_q + w_2 S\,, \\ \text{subject to} \quad & 0 < W_q < W_q'\,. \end{aligned} \tag{39}$$

*5) Cost function optimization:* Based on the analysis presented above, we are able to find the local maximum or minimum of the cost function at point $\tau$ when the derivative of $C$, $C' = 0$. In addition, $C$ at point $\tau$ has the local minimum if the second-order derivation $C'' > 0$. The optimal $\tau$ can then be obtained.

**Algorithm 1:** Selecting the optimal $k$

**input :** $\lambda$, $\mu$, $n_0$, $\alpha$, $K$
$\overline{S}$: The maximum values of $S$.
$\overline{W}_q$: The maximum values of $W_q$ per job.
$\delta$: The ratio of the weighting factors, $w1/w2$.
**output:** $k_{op}$: The optimal number of VNF instances.

Initialize $k$ as 0;
**for** $k \leftarrow 0$ **to** $K - n_0$ **do**
  /* Calculate all probabilities $\pi_{i,j}$, where $(i,j) \in \mathcal{S}$ */
  set $N = n_0 + k$;
  Initialize probability $\pi_{0,0} = 1$;
  **for** $j \leftarrow 1$ **to** $K$ **do**
    /* Get $\pi_{0,j}$ $(j = 1, 2, \ldots, K)$ */
    Calculate $\pi_{0,j}(\lambda, \mu, n_o, \alpha, N)$ using (7)
  **end**
  Calculate $\pi_{1,n_1}$ using (10);
  **for** $i \leftarrow 0$ **to** $k$ **do**
    /* Get $\pi_{i,j}$ $(i = 0, 1, \ldots, k)$ */
    Calculate $\pi_{i,j}$ using Lemma 1 and (23)
  **end**
  **for** $j \leftarrow n_k + 1$ **to** $K$ **do**
    /* Get $\pi_{k,j}$ $(j = n_k+1, n_k+2, \ldots, K)$ */
    Calculate $\pi_{k,j}$ using Lemma 2
  **end**
  Obtain $S = S(k, n_0, K, N, \pi_{i,j})$ using (38), where $(i,j) \in \mathcal{S}$;
  Update $S' = S/\overline{S}$;
  Obtain $W_q = W_q(\lambda, \mu, k, n_0, K, \pi_{i,j})$ using (35) and (36), where $(i,j) \in \mathcal{S}$;
  Update $W_q' = W_q/\overline{W}_q$;
  **if** $S'/W_q' < \delta$ **then**
    $k = k + 1$;
  **else**
    return $k_{op} = k$;
  **end**
**end**

In the next subsection, we apply $\tau = k$ as an example to demonstrate how the derived metrics can be used to determine the optimal $k$ and the number of VNF instances based on their weighting factors. Keep in mind however that we can also set $\tau$ as another parameter, such as $n_0, \mu, K$, or $\alpha$, and apply the same algorithm to be introduced in the next subsection to these parameters.

Moreover, other performance metrics, such as $P_b$, $W$, and $L$, which are expressed in (37), (35), and (34), may also be included to define another cost function, e.g., the variant of the cost function presented in the Appendix. In general, the model and derivations presented in this subsection and the Appendix are generic and can be extended to other types or numbers of metrics.

It is also worth mentioning that we have solved a problem for a system with $n_0 + \sum_{i=0}^{k}(K - n_i) = O(k \times K)$ unknown variables. In contrast to the computational complexity of a conventional method, which is $O(k^3 \times K^3)$, the computational complexity of our recursive algorithm is merely $O(k \times K)$.

Furthermore, our algorithm is numerically stable since it manipulates only positive numbers.

### B. Algorithm for Optimal VNF Instance Allocation

Based on the analytical framework presented above, one can quickly estimate the operation costs and find a suitable configuration for achieving optimal system performance. This approach saves time and reduces costs for an operator without the need for real-life deployment.

In this subsection, we propose an algorithm, Algorithm 1, which is designed to determine the optimal value of $k$ considering the weighting factors. Let us continue with the cost function defined in (1), which is the weighted sum of two metrics, $W_q$ and $S$. Instead of specifying concrete values for the two weighting factors $w_2$ and $w_1$, we define another scalar, $\delta = w_2/w_1$, which is the ratio between $w_2$ and $w_1$. With different values of $\delta$, the interest of a service provider is reflected. For instance, configuring $\delta < 1$ implies that achieving a short response time is more important than reducing the cost of VNF instances.

As shown in the algorithm diagram, there is a set of input parameters in Algorithm 1, i.e., $\overline{S}$, $\overline{W}_q$, $\delta$, $\lambda$, $\mu$, $n_0$, $\alpha$, $N$, and $K$, where $\overline{S}$ and $\overline{W}_q$ denote the maximum values of $S$ and $W_q$ in the system. The output of the algorithm is the obtained optimal value of $k$, i.e., $k_{op}$. To calculate $k_{op}$, we initially set $k$ to 0 and allow $k$ to grow dynamically, bounded by $K-n_0$. Note that $\overline{S}$ and $\overline{W}_q$ are the constraints given by the service provider. As $k$ starts from 0, the ratio of $S'/W'_q$, where $S' = S/\overline{S}$ and $W'_q = W_q/\overline{W}_q$, increases in every iteration accordingly. The algorithm continues the iteration procedure until it finds the lowest $k$ value for $k_{op}$.

Fig. 5 presents a graphical plot of $S$ and $W_q$ to demonstrate how $k_{op}$ is obtained based on different system configurations. The three dashed black lines represent different weighting factors, i.e., $\delta = 2/5, 1, 5/3$. The results based on two values of $n_0$ are reported in this figure, i.e., $n_0 = 60$ and $n_0 = 100$. Each point in the blue solid curve depicts the corresponding values for a pair $S$ and $W_q$, and the whole curve indicates the obtained values for $S$ and $W_q$ based on different $k$ values when $n_0 = 60$. Similarly, the brown solid curve represents the results when $n_0 = 100$. The intersections of these dotted lines and the solid curves are then the obtained optimal values of $k$ with respect to the configured parameters. Take the brown solid curve, i.e., $n_0 = 100$, as an example; the obtained optimal values of $k$ are 44, 24, and 11 for $\delta = 5/3$, $\delta = 1$, and $\delta = 2/5$, respectively. Note however that Fig. 5 is used for illustration purposes only. A service provider may apply Algorithm 1 to other system configurations and obtain an optimal value of $k$ based on the scenario of interest.

## VI. Numerical Results

To validate the mathematical model and evaluate the performance of the proposed autoscaling algorithm, we performed extensive simulations using ns-2, version 2.35. The obtained numerical results are illustrated in this section.

For the network configurations in our simulations, we adopt the results measured from real-life working systems, e.g., $\lambda$ based on Facebook data center traffic [34], $\mu$ based on the base service rate of an Amazon EC2 VM [35], and $\alpha$ based on the average VM startup time [36]. Unless otherwise stated, the default values for our parameter configurations are as follows: $n_0 = 110$, $\mu = 1$, $\alpha = 0.005$, $K = 250$, and $\lambda = 50 \sim 250$. For each simulation run, $15 \sim 750$ million job requests are generated during a simulation time of 300,000 seconds. The simulation results illustrated in the figures below represent the mean values from multiple simulation runs with a 95% confidence level.

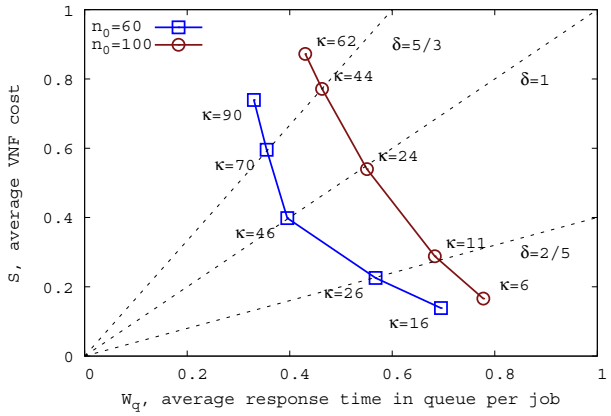



Fig. 5: Selection of the optimal $k$ for a given $\delta$.

### A. Model Validation

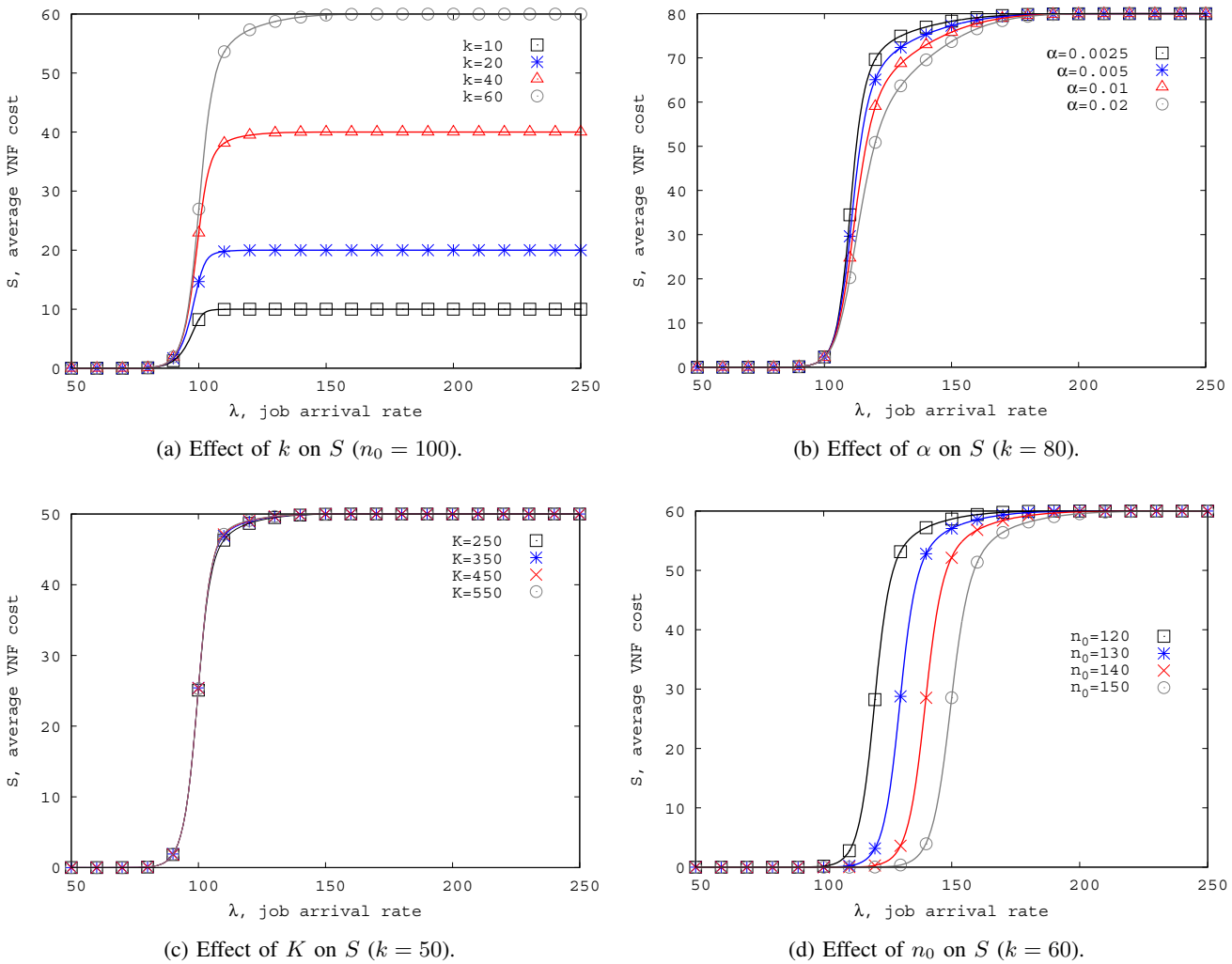
Figs. 6-7 illustrate both the simulation and analytical results with respect to the obtained average VNF cost, $S$, and the average response time in a queue per job, $W_q$, respectively. In these figures, the *lines* denote the analytical results, and the *points* represent the simulation results. It is evident that the analytical and simulation results coincide with each other. Thus, the preciseness of the analytical model is validated. In the rest of this section, we demonstrate the effects of $\lambda$, $k$, $K$, $n_0$, and $\alpha$ on system performance in terms of $S$ and $W_q$.

### B. Effect of the Arrival Rate, $\lambda$

Figs. 6(a)-6(d) illustrate the effect of the arrival rate $\lambda$ on the VNF instance cost $S$. In the beginning, when there are few jobs in the system, the cost, $S$, is 0 because when $\lambda < n_0\mu$, the incoming jobs are handled solely by the legacy equipment and no VNF instances are turned on. As traffic load $\lambda$ increases to $(n_0 + k)\mu$, much more VNF instances need to be turned on. Accordingly, the cost grows sharply. Later on, it increases smoothly, and when $\lambda > (n_0+k)\mu$, $S$ stops growing, as it has reached an upper bound despite the continuing growth of $\lambda$. This occurs because when all the available $k$ VNF instances are turned on, $S$ is bounded as the cost of the $k$ VNF instances.

Figs. 7(a)-7(d) reveal the effect of $\lambda$ on $W_q$. The trend of these curves can generally be divided into three phases[3]: an ascent phase, a descent phase, and a saturation phase. In the first phase, $W_q$ grows sharply because of the setup time of

[3]In Figs. 7(b) and 7(d), only two phases are displayed due to the range of $\lambda$. Given a larger $\lambda$, all three phases would be visible.




(a) Effect of $k$ on $S$ ($n_0 = 100$).

(b) Effect of $\alpha$ on $S$ ($k = 80$).

(c) Effect of $K$ on $S$ ($k = 50$).

(d) Effect of $n_0$ on $S$ ($k = 60$).

Fig. 6: Effects on the NFV instance cost, $S$, of four different parameters.

VNFs. Specifically, when $\lambda \ll n_0\mu$, $W_q$ is almost 0 since all jobs are handled by the legacy equipment. As $\lambda$ approaches $n_0\mu$, and when it is larger than $n_0\mu$, the autoscaling algorithm switches on VNFs. Accordingly, $W_q$ increases until the full capacity of the system has been reached. In the second phase, $W_q$ starts to descend since the switched-on VNFs are in full operation after the setup time. In the third phase, however, $W_q$ increases again and then saturates at another bound. The reason for this increase is that when $\lambda \geq (n_0+k)\mu$, the system is not able to handle incoming jobs. In the end, the curves become saturated because the system capacity is full, unable to handle more jobs. Consequently, the value of $W_q$ is constrained by $K$.

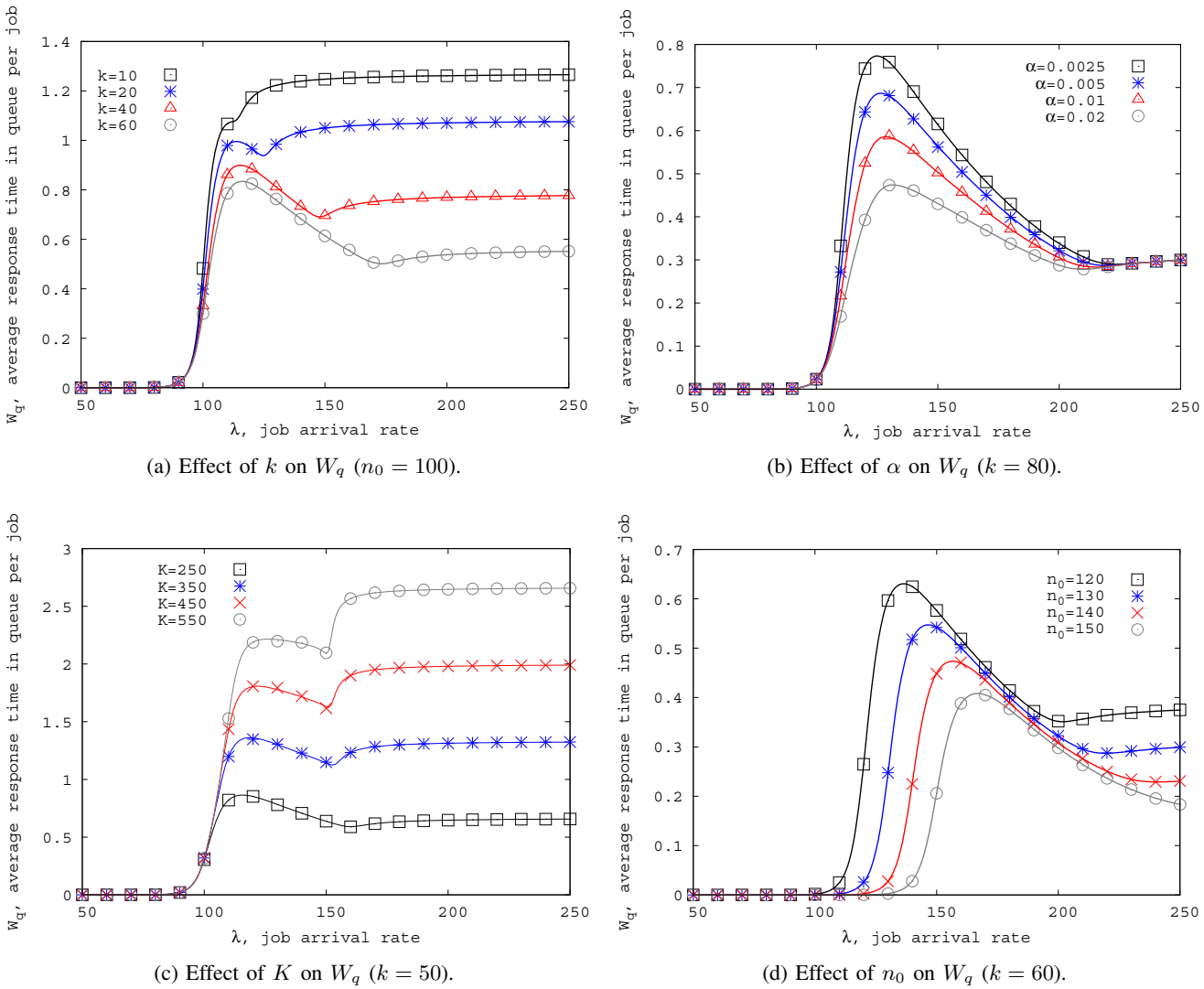
### C. Effect of the Number of VNF Instances, $k$

Fig. 6(a) reveals the effect of $k$ on $S$ with four different values of $k$. For all these four curves, the trend is the same as discussed in the above subsection. With a greater $k$, a larger gap between the initial cost and the cost upper bound is observed because a larger $k$ means that more VNF instances can be allocated to handle incoming job requests, leading to a higher cost. Based on its budget, an operator may configure a suitable value for $k$ based on (38).

Fig. 7(a) illustrates the effect of $k$ on $W_q$. With four different values of $k$, the curves show the same trend as discussed in Section VI-B. Furthermore, the length of the second phase increases with $k$. This is because configuring a larger value for $k$ gives the system higher capability to handle a larger volume of job requests. Accordingly, it increases the time required for the system capacity to reach its upper bound.

### D. Effect of the VNF Setup Rate, $\alpha$

Recall that $\alpha$ is the setup rate of VNFs. To obtain a different setup rate, one can adjust the number or/and volume of resources for VNFs (e.g., CPU, memory). Fig. 6(b) shows the effect of $\alpha$ on $S$, reflected by the slope of these curves. A larger $\alpha$ leads to a smoother slope, but $\alpha$ does not have an effect at the beginning and the end of the curves. The reasons are as follows. A larger $\alpha$ indicates a shorter VNF setup time. Moreover, a shorter VNF setup time allows the VNF to be switched on to full operation faster so that the system is more efficient than the one with a longer VNF setup time. However, when the system is only purely operated by




(a) Effect of $k$ on $W_q$ ($n_0 = 100$).

(b) Effect of $\alpha$ on $W_q$ ($k = 80$).

(c) Effect of $K$ on $W_q$ ($k = 50$).

(d) Effect of $n_0$ on $W_q$ ($k = 60$).

Fig. 7: Effects on the response time per request, $W_q$, of four different parameters.

the legacy equipment or is already saturated with all $k$ VNF instances in full operation, $\alpha$ does not play a role.

Fig. 7(b) illustrates the effect of $\alpha$ on $W_q$, which is also reflected by the slope of these curves. Similarly, a larger $\alpha$ leads to a smoother slope. Moreover, $\alpha$ determines the maximum value of $W_q$. The reason is that a shorter setup time enables the VNF to handle jobs faster.

### *E. Effect of the System Capacity, $K$*

Fig. 6(c) and Fig. 7(c) depict the effects of $K$ on $S$ and $W_q$, respectively. As can be observed in Fig. 6(c), $K$ has little effect on $S$. This is because $S$ is mainly determined by $k$. For $k = 50$, the change in $K$ from $K = 250$ to $K = 550$ is a result of an increased value of $n_0$, which does not contribute to $S$.

On the other hand, $K$ has a significant impact on $W_q$. As shown in Fig. 7(c), different $K$ values lead to very large gaps among these curves. These curves also have three phases, as explained in Subsection VI-B. Furthermore, a larger $K$ leads to a longer $W_q$. This is evident from our analysis expressed in (35) and (36).

### *F. Effect of the Legacy Equipment Capacity, $n_0$*

Fig. 6(d) and Fig. 7(d) illustrate the effects of $n_0$ on $S$ and $W_q$, respectively. As can be observed, all these curves have an initial value of 0 for $S$ and $W_q$. As $\lambda$ increases, this value continues for a period until $\lambda$ reaches a certain threshold. The value of $n_0$ decides the length of this period, after which the curves start to grow. This threshold indicates the system capacity limit within which the legacy equipment can handle ongoing jobs. When $\lambda$ exceeds the capacity of the legacy equipment, both $S$ and $W_q$ start to increase since VNF instances need to be switched on. With a larger amount of legacy equipment $n_0$, the cost caused by activating VNF instances is lower and the response time is shorter.

**Further Discussions** Overall, Figs. 6-7 demonstrate not only the correctness of our analytical model but also the effects of $\lambda$, $k$, $K$, $\mu$, $n_0$, and $\alpha$ on the performance metrics $S$ and $W_q$. Although a service time of $1/\mu$ is assumed to be exponentially distributed in the results presented above, the proposed analytical model can be easily extended to a service time with a deterministic, normal, uniform, Erlang, or Gamma distribution. In [37], we report more simulation results based

on various service time distributions. As such, our analytical framework enjoys wider applicability in various scenarios.

### G. Relationship between $C$ and $k$

Finally, we present how to obtain an optimal $k$ value under various configurations based on Algorithm 1. Figs. 8(a) and 8(b) illustrate the analytical and simulation results for two metrics, i.e., cost function $C$ and response time $W_q$, both shown along the y-axis. The curve and dots in blue or red correspond to $C$, as specified in (39), or the average response time $W_q$, respectively, for a given $\lambda$ value, whereas the x-axis indicates the number of VNF instances, $k$.

Let us first take a look at the red dotted line in Fig. 8(a). It is shown that $W_q$ decreases monotonically as $k$ increases. As discussed earlier, increasing $k$ will make arriving jobs experience a shorter waiting time and better quality of service. Specifically, we observe that $W_q$ first declines sharply, and then, at a point (around $k = 28$), it starts to descend gradually. At the same point, however, the cost function $C$ shown in blue starts to grow sharply, leading to a higher cost. Therefore, for this given arrival rate, i.e., $\lambda = 130$ (job/s), the optimal value of $k$ is obtained as 28. By applying this optimal $k$, the corresponding $W_q$ value is 1.17 s (see the green dotted straight line). If a cellular operator defines a latency level greater than 1.17 s as an SLA violation, i.e., $W_q'$ in (39) is set to be greater than 1.17 s, we can just set $k$ as 28 to balance $C$ and fulfill the latency requirement. Otherwise, we can use the green dotted line to find the $k$ value that corresponds to the latency level lower than the requirement defined by the operator. In this case, although the obtained $C$ value is not the minimum one, $k$ is still the optimal value that can satisfy the latency requirement. Similarly, one can find another example in Fig. 8(b) where an optimal $k$ value for $\lambda = 170$ can be found in the same way.

**Final comment:** In this subsection, we have demonstrated how a latency requirement can be met by identifying the optimal $k$ value through (39). Note, however, that the same method applies to other parameters as well, such as $n_o$, $\mu$, $K$, and $\alpha$ in $\tau$. To do so, one can simply consider the parameter of interest and apply it to (39) or (41).

## VII. Conclusions

In this paper, we proposed a dynamic autoscaling algorithm that addresses the tradeoff between performance and operation cost for VNF instance allocation for NFV in non-standalone 5G networks. We developed an analytical model to evaluate the performance of the proposed scheme and investigated the system performance in terms of the average job response time and operation cost. Through extensive simulations, we demonstrated the preciseness of the model as well as the effectiveness of the algorithm. The scheme developed in this paper fills a research gap by taking both the VNF setup time and legacy equipment capacity into consideration for VNF resource allocation. In addition, we proposed a novel recursive algorithm that reduces the computational complexity significantly. Our study provides a reference for 5G cellular operators to bound their operation cost with a manageable job response time in a systematic manner.

## Appendix
## A Variant of the Cost Function

As mentioned in Footnote 2, other types of cost functions may be defined to reflect the interests of an operator. For example, if the operator also cares about the job blocking probability $P_b$, mean waiting time in the system $W$, and mean number of jobs in the system $L$, the cost function $C$ in (1) can be redefined as:

$$C = w_1 W_q + w_2 S + w_3 P_b + w_4 W + w_5 L, \qquad (40)$$

where $w_3$, $w_4$, and $w_5$ are the weighting factors for $P_b$, $W$, and $L$, respectively. Similarly, since closed-form expressions of these metrics have already been derived in (36), (38), (37), (35), and (34), one can easily find the local minimum when $C' = 1$ and $C'' > 0$ are satisfied.

Based on (40), (39) can be expressed as

$$\begin{aligned} \arg\min_{\tau} \quad & C = w_1 W_q + w_2 S + w_3 P_b + w_4 W + w_5 L \,, \\ \text{subject to} \quad & 0 < W_q < W_q' \,, \end{aligned} \qquad (41)$$

where $\tau \in \{k, n_0, \mu, K, \alpha\}$.

Note again that this model can easily be extended to include a larger number of parameters.

## References


[1] S. Yang, F. Li, S. Trajanovski, R. Yahyapour, and X. Fu, "Recent advances of resource allocation in network function virtualization," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 2, pp. 295–314, Feb. 2021.

[2] B. Yi, X. Wang, K. Li, S. K. Das, and M. Huang, "A comprehensive survey of network function virtualization," *Elsevier Comput. Netw.*, vol. 133, pp. 212–262, Mar. 2018.

[3] B. Nogales, I. Vidal, D. R. Lopez, J. Rodriguez, J. Garcia-Reinoso, and A. Azcorra, "Design and deployment of an open management and orchestration platform for multi-site NFV experimentation," *IEEE Commun. Mag.*, vol. 57, no. 1, pp. 20–27, Jan. 2019.

[4] X. Cheng, Y. Wu, G. Min, A. Y. Zomaya, and X. Fang, "Safeguard network slicing in 5G: A learning augmented optimization approach," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 7, pp. 1600–1613, Jul. 2020.

[5] A. R. Hossain and N. Ansari, "Priority-based downlink wireless resource provisioning for radio access network slicing," *IEEE Trans. Veh. Technol.*, vol. 70, no. 9, pp. 9273–9281, Sep. 2021.

[6] 3GPP TR38.801 v14.0.0, *Study on new radio access technology: Radio access architecture and interfaces (Release 14)*, Std., Apr. 2017.

[7] W. Miao, G. Min, Y. Wu, H. Huang, Z. Zhao, H. Wang, and C. Luo, "Stochastic performance analysis of network function virtualization in future internet," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 3, pp. 613–626, Mar. 2019.

[8] Z. Xiao, W. Song, and Q. Chen, "Dynamic resource allocation using virtual machines for cloud computing environment," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 6, pp. 1107–1117, Jun. 2013.

[9] Z. Hill, J. Li, M. Mao, A. Ruiz-Alvarez, and M. Humphrey, "Early observations on the performance of Windows Azure," in *Proc. 19th ACM HPDC*, 2010, pp. 367–376.

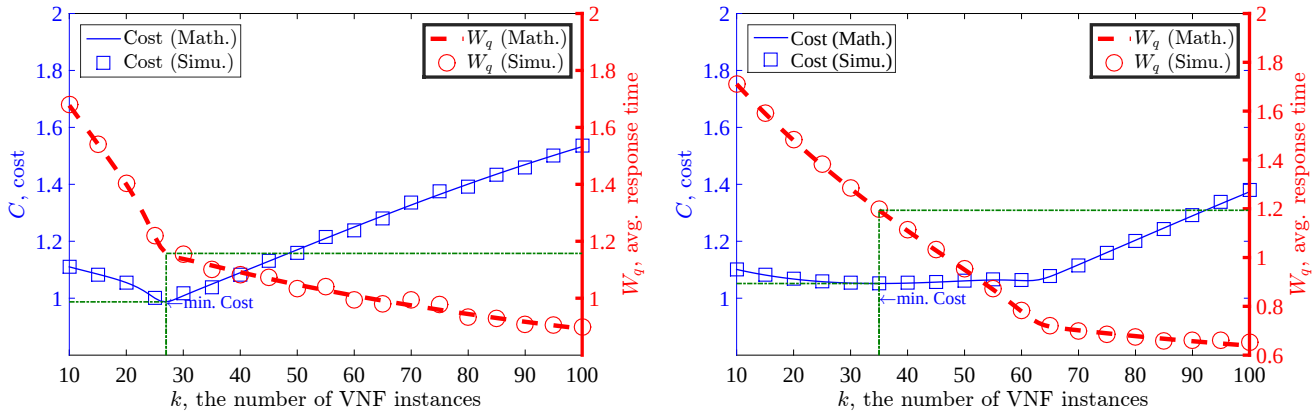
[10] M. Mao, J. Li, and M. Humphrey, "Cloud auto-scaling with deadline and budget constraints," in *Proc. IEEE/ACM Int. Conf. Grid Computing (Grid)*, 2010, pp. 41–48.

[11] J.-C. Chen and T. Zhang, *IP-based next-generation wireless networks: systems, architectures, and protocols*. John Wiley & Sons, 2004.

[12] 3GPP TS 23.002 V15.0.0, *Network architecture (Release 15)*, 3GPP Std., Mar. 2018.

[13] 3GPP TR 32.842 V13.1.0, "Telecommunication management; Study on network management of virtualized networks (Release 13)," Tech. Rep., Dec. 2015.

[14] F. Jokhio, A. Ashraf, S. Lafond, I. Porres, and J. Lilius, "Prediction-based dynamic resource allocation for video transcoding in cloud computing," in *Proc. 21st Euromicro Int. Conf. Parallel, Distrib. Netw.-based Process. (PDP)*, 2013, pp. 254–261.




(a) $\lambda = 130$ (b) $\lambda = 170$

Fig. 8: Optimal $k$ under various conditions.


[15] N. Roy, A. Dubey, and A. Gokhale, "Efficient autoscaling in the cloud using predictive models for workload forecasting," in *Proc. IEEE Int. Conf. Cloud Comput. (CLOUD)*, 2011, pp. 500–507.

[16] J. M. Tirado, D. Higuero, F. Isaila, and J. Carretero, "Predictive data grouping and placement for cloud-based elastic server infrastructures," in *Proc. 11th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput. (CCGrid)*, 2011, pp. 285–294.

[17] D. Niu, H. Xu, B. Li, and S. Zhao, "Quality-assured cloud bandwidth auto-scaling for video-on-demand applications," in *Proc. IEEE INFOCOM*, 2012, pp. 460–468.

[18] R. Calheiros, E. Masoumi, R. Ranjan, and R. Buyya, "Workload prediction using ARIMA model and its impact on cloud applications' QoS," *IEEE Trans. Cloud Comput.*, vol. 3, no. 4, pp. 449 – 458, Aug. 2014.

[19] A. A. Bankole and S. A. Ajila, "Cloud client prediction models for cloud resource provisioning in a multitier web application environment," in *Proc. IEEE 7th Int. Symp. Serv. Oriented Syst. Eng. (SOSE)*, 2013, pp. 156–161.

[20] Z. Shen, S. Subbiah, X. Gu, and J. Wilkes, "Cloudscale: elastic resource scaling for multi-tenant cloud systems," in *Proc. 2nd ACM Symp. Cloud Comput.*, 2011, pp. 1–14.

[21] A. Khan, X. Yan, S. Tao, and N. Anerousis, "Workload characterization and prediction in the cloud: A multiple time series approach," in *Proc. IEEE NOMS*, 2012, pp. 1287–1294.

[22] I. Mitrani, "Service center trade-offs between customer impatience and power consumption," *Elsevier Perform. Eval.*, vol. 68, no. 11, pp. 1222 – 1231, Nov. 2011.

[23] I. Mitrani, "Managing performance and power consumption in a server farm," *Ann. Oper. Res.*, Jul. 2013.

[24] J. Hu and T. Phung-Duc, "Power consumption analysis for data centers with independent setup times and threshold controls," in *Proc. Int. Conf. Numer. Anal. Appl. Math. (ICNAAM'14)*.

[25] I. Mitrani, "Trading power consumption against performance by reserving blocks of servers," *Springer Comput. Perform. Eng.*, vol. LNCS 7587, pp. 1–15, 2013.

[26] T. Phung-Duc, "Multiserver queues with finite capacity and setup time," in *Proc. 22nd. Int. Conf. Anal. Stocha. Model. Tech. Appl. (ASMTA'15)*, May 2015, pp. 173–187.

[27] V. Maccio and D. Down, "Structural properties and exact analysis of energy-aware multiserver queueing systems with setup times," *Elsevier Perform. Eval.*, vol. 121-122, pp. 48 – 66, May 2018.

[28] C.-J. Chang, F.-M. Chang, and J.-C. Ke, "Optimal power consumption analysis of a load-dependent server activation policy for a data service center," *Elsevier Comput. Ind. Eng.*, vol. 130, pp. 745–756, Apr. 2019.

[29] T. Phung-Duc and K. Kawanishi, "Delay performance of data-center queue with setup policy and abandonment," *Ann. Oper. Res.*, vol. 293, no. 1, pp. 269–293, May 2020.

[30] J. Ortin, P. Serrano, J. Garcia-Reinoso, and A. Banchs, "Analysis of scaling policies for NFV providing 5G/6G reliability levels with fallible servers," *IEEE Trans. Netw. Serv. Manag.*, vol. 19, no. 2, pp. 1287–1305, June 2022.

[31] Q. Huang, S. Su, S. Xu, J. Li, P. Xu, and K. Shuang, "Migration-based elastic consolidation scheduling in cloud data center," in *Proc. IEEE 33rd Int. Conf. Distrib. Comput. Syst. Workshops (ICDCSW)*, 2013, pp. 93–97.

[32] Q. Huang, K. Shuang, P. Xu, J. Li, X. Liu, and S. Su, "Prediction-based dynamic resource scheduling for virtualized cloud systems," *J. Netw.*, vol. 9, no. 2, pp. 375–383, Feb. 2014.

[33] A. Bashar, "Autonomic scaling of cloud computing resources using BN-based prediction models," in *Proc. IEEE 2nd Int. Conf. Cloud Netw. (CloudNet)*, 2013, pp. 200–204.

[34] A. Roy, H. Zeng, J. Bagga, G. Porter, and A. C. Snoeren, "Inside the social network's (datacenter) network," in *Proc. ACM SIGCOMM*, 2015.

[35] M. Gilani, C. Inibhunu, and Q. H. Mahmoud, "Application and network performance of Amazon elastic compute cloud instances," in *Proc. IEEE 4th Int. Conf. Cloud Netw. (CloudNet)*, 2015, pp. 315–318.

[36] M. Mao and M. Humphrey, "A performance study on the VM startup time in the cloud," in *Proc. IEEE 5th Int. Conf. Cloud Comput. (CLOUD)*, 2012, pp. 423–430.

[37] Y. Ren, T. Phung-Duc, and J.-C. Chen, "Design and analysis of dynamic auto scaling algorithm (DASA) for 5G mobile networks," National Chiao Tung University, Tech. Rep., 2019. [Online]. Available: https://arxiv.org/abs/1604.05803


## ACKNOWLEDGMENTS

We thank Zheng-Wei Yu and Yi-Hao Lin for their help with our simulations. The research of Yi Ren was supported in part by EPSRC EP/T022566/1, EP/T024593/1, and the Royal Society IEC\R3\213100. The research of Tuan Phung-Duc was supported in part by JSPS KAKENHI Grant Numbers 18K18006 and 21K11765. Jyh-Cheng Chen's work was supported in part by the National Science and Technology Council of Taiwan under grant numbers 111-2221-E-A49-093-MY3, 111-2218-E-A49-023, and 111-3114-E-A49-001. For the work of F. Y. Li, the research leading to these results has received funding from the European Economic Area (EEA) Norway (NO) Grants 2014-2021, under project contract no. 42/2021, RO-NO-2019-0499 - "A Massive MIMO Enabled IoT Platform with Networking Slicing for Beyond 5G IoV/V2X and Maritime Services (SOLID-B5G)". The authors are very grateful to the editor and all reviewers for their constructive comments and valuable suggestions that significantly improved the quality of this paper.

**Yi Ren** received a Ph.D. degree in information communication and technology from the University of Agder, Grimstad, Norway, in 2012. He was with the Department of Computer Science, National Chiao Tung University (NCTU), Hsinchu, Taiwan, R.O.C., as a Postdoctoral Fellow, an Assistant Research Fellow, and an Adjunct Assistant Professor from 2012 to 2017. He is currently a Lecturer with the School of Computing Science, University of East Anglia (UEA), Norwich, U.K. His current research interests include Internet of Things and 5G mobile technology: security, performance analysis, protocol design, radio resource allocation, mobile edge computing, WiFi and Bluetooth Technology, 3GPP, LTE, software-defined networking, network function virtualization, etc. He was a recipient of the Best Paper Award in IEEE MDM 2012.

**Tuan Phung-Duc** is an Associate Professor at Institute of Systems and Information Engineering, University of Tsukuba. He received a Ph.D. in Informatics from Kyoto University in 2011. He is on the Editorial Board of the Journal of the Operations Research Society of Japan and two other international journals. He served as Guest Editor of four special issues of Annals of Operations Research. He was the Chairman of the 10th International Workshop on Retrial Queues (WRQ'2014) and the TPC co-chair of the 23rd International Conference on Analytical, and Stochastic Modelling Techniques and Applications (ASMTA'16), TPC co-chair of The 13th and 14th International Conference on Queueing Theory and Network Applications (QTNA2018, QTNA2019), General co-chair of EAI VALUETOOLS 2020 - 13th EAI International Conference on Performance Evaluation Methodologies and Tools, and General chair of ASMTA/EPEW : The 26th International Conference on Analytical & Stochastic Modelling Techniques & Applications / 17th European Performance Engineering Workshop. Dr. Phung-Duc received the Research Encourage Award from The Operations Research Society of Japan in 2013. His research interests include Applied Probability, Stochastic Models and their Applications in Performance Analysis of Telecommunication and Service Systems.

**Jyh-Cheng Chen** (S'96-M'99-SM'04-F'12) received the Ph.D. degree from the State University of New York at Buffalo in 1998. He was a Research Scientist with Bellcore/Telcordia Technologies, Morristown, NJ, USA, from 1998 to 2001, and a Senior Scientist with Telcordia Technologies, Piscataway, NJ, USA, from 2008 to 2010. He was with the Department of Computer Science, National Tsing Hua University (NTHU), Hsinchu, Taiwan, as an Assistant Professor, an Associate Professor, and a Full Professor from 2001 to 2008. He has been a Faculty Member with National Yang Ming Chiao Tung University (NYCU), formerly National Chiao Tung University (NCTU), since 2010. He is currently the Chair Professor with the Department of Computer Science, and the Dean of the College of Computer Science, NYCU. Dr. Chen is a Distinguished Member of the Association for Computing Machinery (ACM). He was a member of the Fellows Evaluation Committee, IEEE Computer Society. He has received numerous awards, including the Outstanding Teaching Award from both NCTU and NTHU, the Outstanding I.T. Elite Award, Taiwan, the Mentor of Merit Award from NCTU, the Medal of Honor and the K. T. Li Breakthrough Award from the Institute of Information and Computing Machinery, the Outstanding Engineering Professor Award from the Chinese Institute of Engineers, the Outstanding Research Award from the Ministry of Science and Technology, the Best Paper Award for Young Scholars from the IEEE Communications Society Taipei and Tainan Chapters, and the IEEE Information Theory Society Taipei Chapter, and the Telcordia CEO Award.

**Frank Y. Li** received the Ph.D. degree from the Department of Telematics (currently, Department of Information Security and Communication Technology), Norwegian University of Science and Technology, Trondheim, Norway, in 2003. He was a Senior Researcher with the UniK-University Graduate Center (currently, Department of Technology Systems), University of Oslo, Norway, before joining the Department of Information and Communication Technology, University of Agder, Norway, in August 2007, as an Associate Professor and then a Full Professor. From August 2017 to July 2018, he was a Visiting Professor with the Department of Electrical and Computer Engineering, Rice University, Houston, TX, USA. During the past few years, he was an active participant in multiple Norwegian and EU Research Projects. His research interests include MAC mechanisms and routing protocols in 5G and beyond mobile systems and wireless networks, Internet of Things, mesh and ad hoc networks, wireless sensor networks, D2D communications, cooperative communications, cognitive radio networks, green wireless communications, dependability and reliability in wireless networks, QoS, resource management, traffic engineering in wired and wireless IP-based networks, and the analysis, simulation, and performance evaluation of communication protocols and networks. He was listed as a Lead Scientist by the European Commission DG RTD Unit A.03-Evaluation and Monitoring of Programmes in November 2007.