Placement and Chaining for Run-Time IoT Service Deployment in Edge-Cloud

Duong Tuan Nguyen, *Student Member, IEEE*, Chuan Pham, *Member, IEEE*, Kim Khoa Nguyen, *Member, IEEE*, and Mohamed Cheriet, *Senior Member, IEEE*

***Abstract*—This paper investigates an efficient placement and chaining of Virtual Network Functions (VNFs) to provide cloud based IoT services with minimal resource usage cost. We take into account bandwidth capacity and link delay of network connection between clouds where VNFs are allocated and underlying IoT networks where sensors and IoT gateways are deployed. Regarding the constantly changing network dynamics, input traffic of service components is considered at the lower granularity level of messages based on the communication between each VNF and corresponding sensors via IoT gateways. From the algorithm perspective, the specific topology of multiple edge clouds is leveraged to improve the solution. In this paper, we present an NFV-based high-level architecture for a system that enables the deployment of IoT services across multiple edges and clouds. We formulate the VNF placement problem using a non-convex Integer Programming model. Taking into account different IoT topologies, we devise two algorithms for small- and large-scale networks to find the near optimal solution: i) a customized Markov approximation with two techniques, i.e., multi-start and batching, and a node ranking-based heuristic. Simulation and experimental results show that the proposed solution improves the cost up to 21% compared to state-of-the-art schemes.**

***Index Terms*—VNF placement, service function chain, IoT services, edge/cloud computing, QoS.**

I. Introduction

With a dramatic growth in the volume of network traffic over the last decade, Network Function Virtualization (NFV) [1] has been considered as a promising solution whereby network services are provisioned in software-based network functions or elements, i.e., bridges, routers. Thanks to virtualization technology, heterogeneous virtual networks can coexist in the same physical (or substrate) network and share the resources efficiently. This paper focuses on a class of Internet of Things (IoT) services which are typically composed and deployed at run-time to respond to user's need in a specific context [2]. Adopting NFV paradigm allows high flexibility to adapt to the change of service demand, which is critical in the success of IoT application delivery with regard to service performance and reliability.

Along with NFV advantages is a key challenge related to the optimal allocation of resources of a substrate network to virtual network requests, or virtual network embedding (VNE). Despite being intensively investigated in literature [3], [4], deploying such VNE solutions in an arbitrary IoT environment with confidence is still challenging regarding delay sensitivity of IoT services and the constantly changing network dynamics. For the former aspect, previous VNE approaches has mainly focused on the communication between Virtual Network Functions (VNFs) at data center [4] in modeling service latency. In IoT context, modeling End-to-End (E2E) service delay for VNF placement problem requires to consider not only VNF-VNF connection but also between VNF and IoT sensors via IoT gateways which has not been considered in prior work. This is challenging given the complicate interaction of relevant IoT sensors and IoT gateways with VNFs at clouds, i.e., from IoT devices to the VNFs that need to collect sensing data, or in reverse direction to activate certain device's functions. Mathematically, the presence of such the IoT devices introduces additional elements which increase the inherent system complexity and thus creates new constraints to VNE problem.

Regarding the dynamic nature of network traffic, the bandwidth resource [5] should be considered in the VNF placement and chaining problem. Unlike previous studies that addressed this issue by assuming continuous bandwidth demand which is not completely appropriate for IoT devices [6], [7], [8], [9], we go a step further in this paper by investigating the impact of VNFs' input traffic at the lower granularity level of discrete messages via connections between IoT networks and clouds. In addition, we argue that placing VNFs based on resources allocated statically in advance might be not optimal in reality. For practical techniques such as statistical multiplexing of service requests to benefit system resource usage [10] which are appropriate for IoT applications, network resource should be taken into account at a higher dynamic level, i.e., discrete messages, rather than in a static manner as in existing approaches [3].

The paper contribution is three-fold. First, we design a system that enables the deployment of IoT applications in form of service chains across multiple edges and clouds. Second, we propose a model for the optimization problem of VNF placement and chaining with aggregated traffic from IoT gateways and formulate it as a non-convex Integer Programming

Manuscript received July 15, 2019; revised October 7, 2019; accepted October 12, 2019. The authors thank NSERC and Ericsson for funding the project CRDPJ 469977. This research also receives support from the Canada Research Chair, Tier 1, hold by Mohamed Cheriet. The associate editor coordinating the review of this article and approving it for publication was J. Sa Silva. *(Corresponding author: Duong Tuan Nguyen.)*

The authors are with the École de technologie supérieure, University of Quebec, Montreal, QC G1K 9H7, Canada (e-mail: duong-tuan.nguyen.1@etsmtl.net; chuan.pham.1@ens.etsmtl.ca; knguyen@synchromedia.ca; mohamed.cheriet@etsmtl.ca).

Digital Object Identifier 10.1109/TNSM.2019.2948137

(IP) problem. The novelty of our model lies in the consideration of input traffic of VNF and the presence of IoT devices, i.e., sensors, gateways in IoT services. Particularly, we model the latency for service chains while taking into account the specifications of connection between clouds where VNFs are deployed and IoT gateways, such as the distance to IoT devices and the connectivity to multiple edges and clouds. Third, regarding the NP-hardness of proposed problem, we introduce a Markov approximation based framework that adopts multistart and batching techniques (MBMAP) to solve the combinatorial network problem. The framework exploits underlying IoT infrastructure to perform algorithms in a distributed manner and consequently accelerate convergent rate which has been known as a limitation of Markov-based algorithms due to the large space of states. We also present another heuristic (NRP) that employs the concept of node rank in placing VNFs. The heuristic aims for large-scale networks and is considered as a baseline to demonstrate the advantage of MBMAP given a large number of possible states. Our source code is available online[1] for other researchers to use and modify. Simulations and experiments' results show the effectiveness of the proposed MBMAP over prior works that do not consider the IoT network.

The rest of this paper is organized as follows. Section II reviews prior works. System modeling and the formulation of the optimization problem are explained in Section III. Then Markov-based approximation algorithm MBMAP and node-ranking heuristic NRP are described in Section IV. Section V presents performance evaluation of the proposed methods with simulation and testbed settings. Finally, conclusions are drawn.

II. Related Work

With the rapid growth of virtualization technology, a large number of recent publications have studied VNF optimal resource provisioning and service chain routing. The VNE has been investigated in the literature from various aspects, such as system models [6], [7], objectives [3], [4] and solutions [7], [8]. In this section, we summarize the main results on VNE for IoT services and explain how our work is distinguished from the others.

In [11], the authors propose the solutions for the problem of VNF embedding for virtual 5G network infrastructure while dealing with the mobility features and service usage behavioral patterns of mobile users. The solutions address two conflicting objectives, which are the insurance of Quality of Experience (QoE) via the placement of VNFs of data anchor gateways closer to end users and the avoidance of the relocation of mobility anchor gateways via placing their corresponding VNFs far enough from users. Apart from user mobility, the constantly changing network dynamics as a well-known characteristic of IoT network is addressed in [3]. Another IoT service specific is the presence of micro-data centers, known as edge cloud, whose locations significantly affect the requirement of ultra-short latency and has been investigated in [12]. The study in [13] address the VNE problem regarding the constraints related to the location of substrate nodes. Reference [14] adopts network topology information including node location to conduct a node-ranking approach to solve the problem. In general, all of these prior works mainly focus on VNF location optimization for services between end-user and corresponding VNFs, not service function chain.

Focusing on the relation between link and server usage, the authors in [15] investigate the joint VNF placement and path selection problem. While the approach can be generalized to include the underlying IoT network, it requires an effort to adapt the model for distributed clouds as well as the formulation of constraints on service chain latency in the IoT context. A similar approach in [16] considered partial orders and anti-affinity rules which states that two VNFs cannot handle the same service chain on the same node

In [17], the authors propose an analytical model for the placement of service function chains in multi-cloud environments. However, they only consider inter-cloud traffic *w.r.t* the fact that inter-cloud links are more likely to be congested and more expensive compared against the links within a single datacenter. Another work for placing service chains across multiple clouds in [18] adopts machine learning technique for a predictive model combining with random cloud selections. Tackling the issue of deploying network services across multiple Points of Presence (PoPs), the framework in [19] provides an optimization model on various metrics, i.e., cost of assigning VNFs to PoPs, overall delay, and overall resource link usage.

Closer to our work is MaxZ [5] that proposes a model accounting for services involved in 5G networks such as IoT, Machine-to-Machine (M2M) applications. Adopting a queueing model for VNFs, the authors deal with traffic not only between VNFs but also from outside the system, which might be applied for the case of IoT devices. However, MaxZ neglects IoT nodes, in terms of their resource capacity and connection delay, and this, as confirmed by our numerical results, can yield sub-optimal performance.

In this work, we consider three system features, i.e., distributed clouds, multiple VNF instances, connections between clouds and underlying IoT networks, which are not taken into account in prior works.

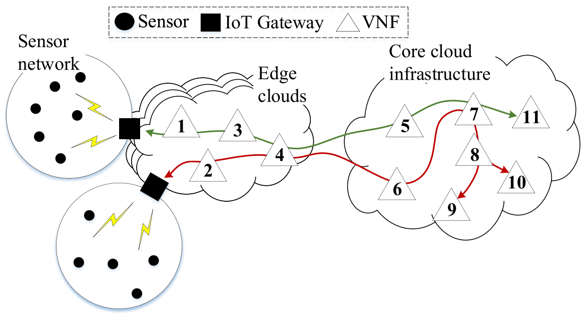
III. System Architecture

In this section, we describe the system that performs service function chaining on multiple clouds for IoT applications. The overall architecture as a reference for implementing and deploying proposed solution with an illustrative IoT-based use case is explained.

A. System Description

Fig. 1 depicts a system composed of multiple clouds where VNFs are deployed to implement service functions. Each VNF can be replicated on different places depending on the number of licenses that the provider has purchased [20]. A VNF can process network traffic from other VNFs or sensor devices (or nodes) scattered in a sensor field via IoT gateways. While a sensor may have multiple interfaces, i.e., Bluetooth, WiFi, LTE, due to its constrained resource, only one interface is

[1] https://github.com/hoangtuansu/smal



AQ2 Fig. 1. Multi-cloud service function chain for IoT applications.

activated at a given time and connects to one gateway within its coverage. Each node either collects data (i.e., temperature, noise) or performs a certain function (i.e., sprinkle, smart light). Toward sensor side, the VNF either receives and processes data from that sensor or sends a control message to activate its function.

The IoT gateways aggregate data from connected sensors and communicate with VNFs through the network link between the gateways and the clouds. A gateway might have various interfaces (i.e., wireline, cellular, LoRA) and thus connects to several clouds at the same time through the Internet. A user request for a service will be served by a chain of service functions performed by VNFs which interact with the IoT gateways to retrieve the input data or trigger the commands from or towards the sensor. In this work, we consider a common IoT case in which service functions are executed in sequential or branching manners [21].

B. Overall Architecture

From the aforementioned system, we design an implementation architecture that takes into account not only the presence of multiple clouds but also the service management for micro-services as service functions, and underlying IoT network as shown in Fig. 2.

At each edge, Optimization Agent (OpAg) and VNF Allocator (VAl) are two main components of Edge Orchestrator (EdOr). Specific components and functionalities of EdOr are similar to MANO reference architecture that can be found in [1]. The role of OpAg is to expose both resource and service function's information of the edge to the Global Optimizer (GlOp) at Core Orchestrator (CoOr) which is in turn deployed at Core Cloud. Placement scheme returned by OpAg is used by VAl component to perform cloud's resource allocation.

The Network Controller is responsible for controlling network resources and establishing connectivity between VNFs that implement service functions. It maintains a list of IoT network topologies including gateways and sensors, from which providing necessary information, i.e., data rate, latency, as input of OpAg.

At Core Cloud, Service Chain Manager component of the CoOr retrieves information from BSS/OSS system to construct a catalog of IoT applications. Similarly to the EdOr, the

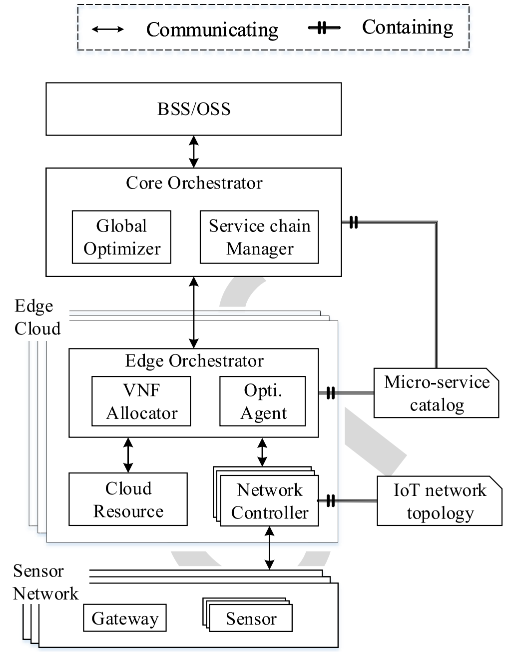


Fig. 2. Implementation architecture.

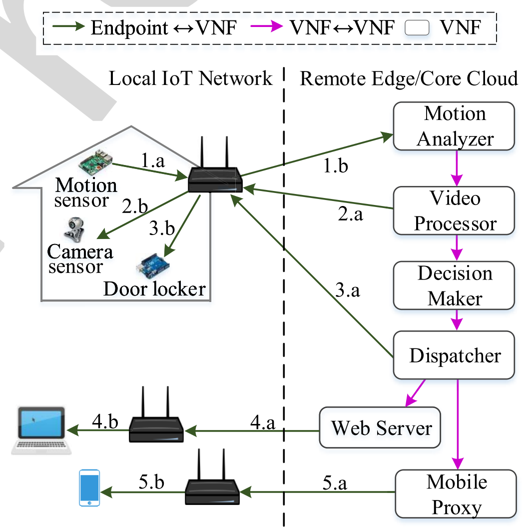


Fig. 3. An illustrative IoT service chain with multiple end-points.

CoOr's catalog is served as the input of GlOp and might be related to multiple edge clouds.

C. Illustrative Use Case

An illustrative example of the service chain that is formed in accordance with a security surveillance scenario as shown in Fig. 3. In this use case, a motion sensor (MS) is the initial source, a camera sensor (CS) provides supporting data to improve the decision making process, and destination nodes include Door Locker, a laptop representing surveillance service provider, and a mobile phone as a user. Service functions, e.g., Motion Analyzer, Video Processor (VP), Decision Maker (DM), Dispatcher (DP), Web Server (WS) and Mobile Proxy (MP), are implemented as VNFs running on edge-cloud. Upon

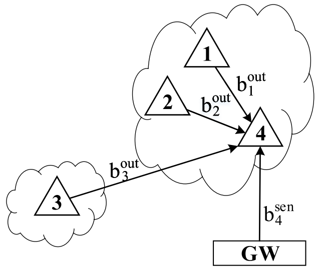


Fig. 4. Details of bandwidth required by a VNF.

detecting a motion (step 1a & 1b), Motion Analyzer (MA) checks whether or not it is a suspicious move, i.e., not via the main door. If it is the case, the MA will inform VP to trigger the camera sensor to perform at a higher resolution (step 2). Using face recognition, DM decides whether or not it is an intrusion if the person is not identified as a home user. DP receives decision result from DM and send an activation message to Door Locker (step 3) as well as notifies other two endpoints (step 4 & 5) which are also behind IoT gateways. Note that the DP can be configured to forward the message to more than one endpoints at step 4 & 5.

In this use case, placing service functions or VNFs with only consideration of data center's resource does not guarantee the performance of IoT services. IoT traffic in terms of sets of discrete messages, if ignored, may result in a sub-optimal placing solution as confirmed by our simulation and experimental results. Moreover, the communication delay between local IoT networks and remote edge/core clouds also plays an important role in E2E service latency, which is critical in many scenarios, e.g., security surveillance.

IV. System Modeling & Problem Formulation

We model the system described in Fig. 1 as a directed graph $\mathcal{G} = \{N \cup G, E\}$ where $N \cup G$ and E are the sets of nodes and links respectively. To facilitate the model, both edge and core cloud are referred to the set of N clouds and G is the set of IoT gateways. A link $(q, q') \in E$ connecting two entities either clouds or gateways or both represents a logical communication link between them. $\Phi^B_{q,q'}$ and $l_{q,q'}$ denote the capacity and delay of edge (q, q'), respectively. While $\Phi^B_{g,n}$ is estimated based on the communication technology of gateway's network attachment point, $\Phi^B_{n,n'}$ is usually determined by the contract between network infrastructure providers. We use m_n^{com} and $m_{q,q'}^{net}$ to define the cost of one computing resource unit at $n \in N$ and one network bandwidth unit over the link (q, q'), respectively. The mathematics notations are summarized in Table I.

We collect the set V of VNFs hosted at the clouds. For any VNF $v \in V$, let κ_v denote the number of v's replications (or instances), v_i where $i = 1, \dots \kappa_v$ is the i-th replication of v, b_v^{out} the required bandwidth for an IoT gateway to send v's aggregated messages to other VNFs, and $\tau_v \in \{0, 1\}$ indicates whether v is an IoT-based VNF ($\tau_v = 1$) or not ($\tau_v = 0$). The implementation of VNFs is realized via virtual machines

TABLE I
Notation List

General Inputs	
N, G	Set of clouds and connected IoT gateways
C, V	Set of VNFs and service chains
V_g	Set of VNFs associated with gateway g
$\alpha_{g,n}$	Indicator of the association between g and n
κ_v	Number of VNF v's replications
λ_c	Arrival rate of service chain c's request
λ_v^{sen}	Arrival rate of data from sensor to VNF v
λ_{v_i}	Arrival rate of network traffic at VNF instance v_i
τ_v	1 if v is an IoT-based VNF, 0 otherwise
$\beta_{u,v}^c$	1 if u links to v in service chain c, 0 otherwise
Service Latency	
Φ_c^L	Maximum tolerated delay of service chain c
Γ_{v_i}	Processing delay of VNF instance v_i
Γ_g	Aggregation delay at gateway g
L_{c,v_i}^{ctl}	Transmission delay between VNF instance v_i and sensor
$L_{u,v}^{com}$	Transmission delay between two VNFs u and v
L_c	Total delay of service chain c
System Resource	
b_v^{out}	Output network bandwidth of VNF v
b_v^{sen}	Network bandwidth between VNF v and sensor
Φ^B	Link capacity between any two nodes
$B_{q,q'}$	Network bandwidth between two nodes q, q'
r_n	Compute resource for n to process a bandwidth unit
R_n	Compute resource allocated at cloud n
System Cost	
m_n^{com}	Cost per compute resource unit at cloud n
$m_{k,q}^{net}$	Cost per bandwidth unit of the link between nodes k, q
M_{net}	Network resource cost of the whole system
M_{com}	Compute resource cost of the whole system
M_{sys}	Total system cost
Decision Variables	
$x_{v_i}^n$	1 if VNF instance v_i is at cloud n, 0 otherwise
$y_{v_i}^c$	1 if service chain c uses VNF instance v_i, 0 otherwise

which are typically shifted in different templates (or configurations) in terms of CPU, memory, storage, and so on, depending on the cloud they are provisioned. Having said that, we use r_n to denote units of resource allocated for a VNF instance at the cloud n to process a bandwidth unit.

Given a gateway $g \in G$, a VNF v is associated with the gateway g if it is an IoT-based and g has a connection to its corresponding sensor. A set of such the VNFs is presented after g, i.e., V_g. Additionally, it is assumed that the sensor of the IoT-based VNF v's sensor generates sensing data at the rate λ_v^{sen} which requires b_v^{sen} units of bandwidth. For the sensors controlled by VNFs rather than generating sensing data, λ_v^{sen} and b_v^{sen} are set to 0.

Given C as the set of independently and identically distributed (i.i.d) service chains, each $c \in C$ is characterized by λ_c the initial service rate, $o(c)$ the source VNF, $d(c)$ the destination VNFs, Φ_c^L the maximum tolerated delay and $\vec{V}_c$ the directed tree composed of related VNFs. Any VNF can be shared by different service chains. One use case for such the shared VNF's instance is that the firewall function can be employed to filter traffic of multiple chains. For the sake of simplifying the latency model of a service chain, the notation $\vec{V}_{c \dashv v}$ is used to present the sub-sequence (or a path) from the first VNF in *c* to *v*. From Fig. 3, *o*(*c*) is the VNF MA while *d*(*c*) is the set {*DP*, *WS*, *MP*}. An example of $\vec{V}_{c \dashv v}$ with *v* as VNF-WS is {*MA* → *VP* → *DM* → *DP* → *WS*} or {*MA* → *VP* → *DM* → *DP* → *MP*} with *v* as VNF-MP. Considering any two VNFs *v* and v', the notation $\beta^c_{v,v'} \in \{0,1\}$ with $\beta^c_{v,v'} = 0$ if $v \equiv v'$ indicates whether or not they are linked together regardless their instances and $\sum_{v' \in V_c} \beta^c_{v,v'} = 1, \forall v \in V_c$.

The output of our model is the optimal solution of the VNF placement problem for the given set of inputs and is represented by decision binary variables $\mathbf{x} = \{x^n_{v_i}\}^{n \in N}_{v \in V, 1 \le i \le \kappa_v}$ and $\mathbf{y} = \{y^c_{v_i}\}^{c \in C}_{v \in V, 1 \le i \le \kappa_v}$. Precisely, $x^n_{v_i} = 1$ if v_i is allocated at *n* and 0 otherwise whereas $y^c_{v_i} \in \{0,1\}$ indicates the assignment of the replica v_i to requested service chain *c*.

1) Resource Constraint: The bandwidth required for the communication channel between the VNFs at the same cloud and associated with the same gateway *g* should not exceed the link capacity between *g* and *n*. Hence, with $\mathbf{x}^n_v = \sum_{1 \le i \le \kappa_v} x^n_{v_i}$, we get

$$B_{g,n}(\mathbf{x}) = \sum_{v \in V_g} b^{sen}_v \mathbf{x}^n_v \le \Phi^B_{g,n} \tag{1}$$

Similarly, the total amount of bandwidth that any two consecutive VNFs in any service chain, that connects n' to *n* must be lower than $\Phi^B_{n',n}$. This value $B_{n',n}(\mathbf{x})$ is computed based on the data that a VNF instance generates towards its connected VNF of the same chain. Since each chain only has one pair of any two VNFs v', *v*, the value of $B_{n',n}(\mathbf{x})$ is obtained in terms of $\mathbf{x}^{n'}_{v'}$ and $\mathbf{x}^n_v$, that is

$$B_{n',n}(\mathbf{x}) = \sum_{c \in C} \sum_{v',v \in V} \beta^c_{v,v'} b^{out}_v \mathbf{x}^{n'}_{v'} \mathbf{x}^n_v \le \Phi^B_{n',n} \tag{2}$$

For a VNF instance, there are two input data sources from its precedent connected VNFs of service chains, and the sensors in case of an IoT-based VNF as shown in Fig. 4. The bandwidth for an instance of *v*, i.e., B_v, is

$$B_v = \sum_{c \in C} \sum_{v' \in V} \beta^c_{v',v} b^{out}_v + \tau_v b^{sen}_v \tag{3}$$

The total amount of resource $R_n(\mathbf{x})$, $\forall n \in N$ needed to deploy a VNF for the cloud *n* considering resource availability Φ^R_n is computed as

$$R_n(\mathbf{x}) = \sum_{v \in V} \mathbf{x}^n_v r_n B_v \le \Phi^R_n. \tag{4}$$

2) System Stability: We model a VNF replica as a *M*/*M*/1 queueing system with μ^n_v the service processing capacity and λ_{v_i} the arrival rate. Similar to the bandwidth, λ_{v_i} is also attributed to the traffic from two sources: precedent VNFs of v_i in all the chains of *C*, and its sensor through the gateway with $\tau_v = 1$ and hence

$$\lambda_{v_i}(\mathbf{y}) = \sum_{c \in C} \sum_{v' \in V} \sum_{1 \le j \le \kappa_{v'}} \beta^c_{v',v} \lambda_{v'_j} y^c_{v'_j} y^c_{v_i} + \tau_v \lambda^{sen}_v \tag{5}$$

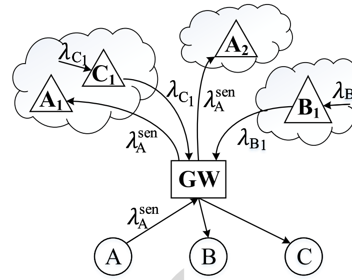


Fig. 5. Arrival rate at VNF in details.

As there is only one precedent VNF of *v* in *c*, the Equ. (5) can be written in terms of λ_c as follows

$$\lambda_{v_i}(\mathbf{y}) = \sum_{c \in C} \left(\lambda_c y^c_{v_i} + \sum_{v' \in \vec{V}_{c \dashv v'}} \tau_{v'} \lambda^{sen}_{v'} \right) \tag{6}$$

Note that although the sensors are typically configured to periodically sense ambient conditions, the sensing periods are different from a sensor to others. Thus, it can be assumed that the data generated by sensors follows the Poisson process. In other words, the arrival of traffic to the IoT gateway can be considered as a Poisson process. It is reasonable to model the gateway as a *M*/*M*/1 queueing system, with μ_g the service processing rate together with λ_g. From Fig. 5,the gateway receives data from *v*'s sensor if $\lambda^{sen}_v > 0$ and from its associated IoT-based VNFs if $\lambda^{sen}_v = 0$. Note that the IoT gateway's presence does not change the value of λ^{sen}_v as arrival rate of a *M*/*M*/1 system is equal to departure rate. This yields

$$\lambda_g(\mathbf{x}, \mathbf{y}) = \sum_{\substack{(n,g) \in E \\ v \in V_g}} \sum_{1 \le i \le \kappa_v} x^n_{v_i} (\lambda^{sen}_v + \lambda_{v_i}(\mathbf{y})) \tag{7}$$

To guarantee a VNF instance is not overloaded, the average time between two successive messages must be greater than the mean processing time by any server of *v* to a message. In other words, we require the stability condition for the system to be stable, that is

$$\lambda_{v_i}(\mathbf{y}) < \sum_{n \in N} x^n_{v_i} \mu^n_{v_i} \tag{8}$$

$$\lambda_g(\mathbf{x}, \mathbf{y}) < \mu_g. \tag{9}$$

3) Service Latency Constraint: In order to formulate the latency of a service function chain, it needs to retrieve the formulation for the processing time at each VNF instance v_i, i.e., Γ_{v_i} and the aggregation time at *g*, i.e., Γ_g. From (5) and (7), we have

$$\Gamma_g(\mathbf{x}, \mathbf{y}) = \left(\mu_g - \lambda_g\right)^{-1}, \forall g \in G \tag{10}$$

$$\Gamma_{v_i}(\mathbf{x}) = \left(\sum_{n \in N} x^n_{v_i} \mu^n_v - \lambda_{v_i}(\mathbf{y}) \right)^{-1}, \forall v \in V \tag{11}$$

Assuming that all the VNFs in the same cloud are incurred the same delay of communicating with external entities and the delay between a gateway and a sensor is negligible to be ignored. The delay of a c's control message from a IoT-based VNF instance v_i, if exists, to its sensor through g is

$$L^{ctl}_{c,v_i}(\mathbf{x},\mathbf{y}) = \sum_{\substack{(g,n)\in E\\ v\in V_g}} \tau_v x^n_{v_i}(\Gamma_g + l_{g,n}) \tag{12}$$

Next, given two VNFs v and v', the following is the formulation of the inter-network delay between their hosting clouds

$$L^{com}_{v,v'}(\mathbf{x}) = \sum_{(n,n')\in E} \mathbf{x}^n_v \mathbf{x}^{n'}_{v'} l_{n,n'}, \forall v, v' \in V \tag{13}$$

Given a source and multiple destinations, the total delay for a service chain is the maximum delay for transmitting a message to all the destination nodes which must not be greater than the maximum tolerated latency Φ^L_c. As a result

$$\begin{aligned} L_c = \max_{v\in d(c)} &\sum_{u\in \vec{V}_{c\dashv v}} \sum_{1\le i\le \kappa_u} \left(\sum_{w\in \vec{V}_{c\dashv v}} y^c_{u_i}\beta^c_{u,w} L^{com}_{u,w} + y^c_{u_i}\Gamma_{u_i} \right) \\ &+ \sum_{1\le j\le \kappa_v} y^c_{v_j} L^{ctl}_{c,v_j} \le \Phi^L_c, \forall c \in C \end{aligned} \tag{14}$$

In Equ. (14), L_c is composed of the transmission latency between every pair of VNFs, i.e., the first term inside the brackets, the time for each VNF to process the message, i.e., the second term at the next line, as well as the time for the last node to activate its corresponding sensor, i.e., the last term.

4) System Cost: In this paper, we also consider total system cost which is the weighted sum of the cost of allocated network bandwidth (M^{net}) and that of computing resource (M^{com}), that is

$$\begin{aligned} M^{sys} &= \omega M^{net} + (1-\omega) M^{com} \\ &= \omega \sum_{(q,q')\in E} B_{q,q'} m^{net}_{q,q'} + (1-\omega) \sum_{n\in N} R_n m^{com}_n . \end{aligned} \tag{15}$$

5) Problem Formulation: Let $\alpha_{g,n} \in \{0,1\}$ represent the connection between g and n. Based on above analysis, IoT VNF placement problem is formulated as the following constrained optimization, i.e., by i, j indicate the instances' indices of VNFs v and u, respectively:

$$\underset{\mathbf{x},\mathbf{y}}{\text{minimize}} \quad M^{sys} = \omega M^{net} + (1-\omega) M^{com} \tag{16}$$

$$\text{subject to} \quad (1),\ (2),\ (4),\ (8),\ (9),\ (14)$$

$$(\forall v \in V_g, 1 \le i \le \kappa_v){:} x^n_{v_i} \le \alpha_{g,n} \tag{17}$$

$$(\forall v \in V_g, 1 \le i \le \kappa_v){:} \quad \sum_{n\in N} x^n_{v_i} \le min(\kappa_v, \sum_{m\in N} \alpha_{g,m}) \tag{18}$$

$$(\forall c \in C, u \in V_c){:} \sum_{1\le j\le \kappa_u} y^c_{u_j} = 1 \tag{19}$$

$$(\forall (u,v) \in V, 1 \le i \le \kappa_v, 1 \le j \le \kappa_u){:} \quad x^n_{v_i} \in \{0,1\}, y^c_{u_j} \in \{0,1\} \tag{20}$$

Our objective is to find a placement scheme to minimize the total cost incurred in the system. Equ. (17) implies that a cloud does not provision the instance of a VNF if the gateway connecting to that instance is not associated with that VNF. In this case, both $x^n_{v_i}$ and $\alpha_{g,n}$ are set to zero. If the gateway is associated with n, $\alpha_{g,n}$ is set to 1 and $x^n_{v_i}$ can be a free variable. Moreover, the number of deployed VNF instances must not exceed the number of connections between its associated gateways and the clouds as specified by constraint (18). Equ. (19) stipulates a VNF cannot be involved more than one time by a service chain and so do its instances.

V. IoT Topology-Aware VNF Placemenet

The problem (16) is NP-hard and it is difficult to obtain an exact solution in the polynomial time. Hence, a Markov-based approximation (MA) framework [22] is adopted to find a near-optimal solution within an acceptable period of time. In this section, we present multistart and batching techniques that are implemented regarding IoT topology. We explain how these techniques are incorporated with MA framework, *a.k.a* MBMAP, to address slow convergence drawback.

A. Batching Markov Approximation Framework

1) Log-Sum-Exp Approximation: Let $f = \{\mathbf{x}, \mathbf{y}\}$ indicate a specific VNFs placing scheme and $\mathcal{F}$ be the set of feasible configuration defined by constraints of problem (16). A change of any VNF instance either allocated at a cloud or a service chain will lead to another configuration or new state in the context of Markov chain. Let M^{sys}_f denote system cost under a configuration f. The problem (16) is re-written as follows:

$$\underset{\mathbf{p}\ge 0}{\text{minimize}} \sum_{f\in\mathcal{F}} p_f M^{sys}_f \tag{21}$$

$$\text{s.t.} \sum_{f\in\mathcal{F}} p_f = 1 \tag{22}$$

where p_f is the probability of choosing configuration f. Adopting log-sum-exponential approximation approach in [22], the problem (21) is approximated as

$$\underset{\mathbf{p}\ge 0}{\text{minimize}} \sum_{f\in\mathcal{F}} p_f M^{sys}_f + \frac{1}{\delta} \sum_{f\in\mathcal{F}} p_f log(p_f) \tag{23}$$

$$\text{subject to} \sum_{f\in\mathcal{F}} p_f = 1 \tag{24}$$

where δ is a positive constant and a gap upper-bound by $\frac{1}{\delta} log|\mathcal{F}|$.

By solving the Karush-Kuhn-Tucker (KKT) conditions of the problem (23), we obtain the optimal and close-form probability solution, that is

$$p^*(\mathbf{M}^{sys}_f) = \frac{exp(-\delta M^{sys}_f)}{\sum_{f'\in\mathcal{F}} exp(-\delta M^{sys}_{f'})}, \forall f \in \mathcal{F} \tag{25}$$

Obviously, the more optimal a configuration is chosen for the whole system, the closer the system cost is to the optimal value with the aforementioned gap. However, in order to compute p^*_f for each configuration, it requires to take into account the whole feasible configuration space to compute (25), i.e., the sum at the denominator, which is inefficient due to the large solution space $\mathcal{F}$. Instead, a Markov chain is constructed in a way that the stationary distribution of each state is p^*_f.

While the existence of such the chain has been already proven in [22], the states and the transition mechanism respecting to a transition probability need to be defined.

2) Markov Chain Construction Procedure: Let two configurations f, f' in $\mathcal{F}$ represent two states of the time-reversible ergodic Markov chain with the stationary probability $p^*(\mathbf{M}_f^{sys})$. The transition probability between f and f', which are $t_{(f\to f')}$ and symmetrically defined $t_{(f'\to f)}$, must satisfy following balanced equation:

$$p^*(\mathbf{M}_f^{sys})t_{(f\to f')} = p^*(\mathbf{M}_{f'}^{sys})t_{(f'\to f)} \tag{26}$$

There are many values of $t_{(f\to f')}$ and $t_{(f'\to f)}$ in Equ. (26). We choose the following option with $t_{(f\to f')}$ defined symmetrically, which is:

$$t_{(f\to f')} = \rho\ exp\left(\frac{1}{2}\delta\Big(M_f^{sys} - M_{f'}^{sys}\Big)\right) \tag{27}$$

where ρ is a conditional non-negative constant. Intuitively, this can be understood that if a transition results in a lower system cost, i.e., $M_f^{sys} > M_{f'}^{sys}$, the value of $t_{(f\to f')}$ increases and makes the occurrence of f' more likely. A basic procedure to construct a Markov chain toward the stationary distribution is thus given as :

- **Step 1**: Initialize a feasible configure f_0, in terms of placing VNFs instances onto clouds, i.e., $\mathbf{x}_0$ and assigning them to service chains, i.e., $\mathbf{y}_0$. Compute the system cost $M_{f_0}^{sys}$.
- **Step 2**: From f, generate a new VNF placement scheme, in terms of $\mathbf{x}'$ and $\mathbf{y}'$ for a new configuration f' with a corresponding cost $M_{f'}^{sys}$.
- **Step 3**: Compute the transition probability based on Equ. (27) and set the best configuration to either the current one f or the newly generated one f'.
- **Step 4**: Go back Step 2 until stopping criteria is met.

3) Multistart and Batching Based Markov Approximation Placement Framework: Our MBMAP framework is designed following several observations. First, an inherent limitation of the Markov method is the slow convergence rate due to the large space of states. In the worst case, an algorithm might go through $O(2^{\sum_{v\in V}\kappa_v(|C|+|N|)})$ states to retrieve the optimal placing scheme of the problem (16). In practice, there is typically a stopping criteria to achieve a near-optimal solution within an acceptable time. Therefore, we argue that the more space's size and the number of computation steps are reduced, the "nearer" optimal a solution could be found. For space's size, it can be done by eliminating or fixing variables that do not satisfy constraints. Procedure SPACEREDUCE in Algorithm 1 is an example of assigning constant values to a subset of variables. It can be intuitively understood that a VNF instance should not be placed on a cloud that does not connect to the gateway associated with that VNF. By doing so, we ignore states with invalid placements and thus enhance algorithm's performance.

Similarly, procedure COSTDIFF illustrates an efficient method to compute cost difference term of transition probability in Equ. (27). The idea is to compute the cost associated with each transition and to have it added to the original cost of the current state to obtain the value associated with the newly formed state rather than manually calculating the cost of each state. Note that the loop at line 3 of COSTDIFF can be avoided by performing lines 4-6 upon changing the placement to any VNF instance.

Algorithm 1 Solution State Reduction Procedure

procedure SPACEREDUCE
    Set instances' number less than that of clouds
    for each $v \in V, n \in N$ **do**
        Set g as the gateway associated with v
        if g connects to n **then**
            $x_{v_i}^n \leftarrow 0, \forall 1 \le i \le \kappa_v$

Algorithm 2 Efficient Computation Support Procedures

1: **procedure** COSTDIFF
    ▷ f: previous configuration, f': set of configurations, V': newly replaced VNFs, Δ: cost difference
2:     Set $\Delta \leftarrow 0$
3:     **for each** v's instance $v_i \in V'$ **do**
4:         Set g as the gateway associated with v, and let n, n' be clouds connecting to v_i under f, f'
5:         $\Delta \leftarrow \Delta + (m_{n',g}^{net} - m_{n,g}^{net})(b_v^{sen} + b_v^{out})$
6:         $\Delta \leftarrow \Delta + B_v(m_{n'}^{com}r_{n'} - m_n^{com}r_n)$

Second, it may take time for a Markov approximation basic procedure to retrieve the feasible configuration at the beginning (Step 1) as well as from another (Step 2). To tackle this issue, we design a batching transition placement heuristic based on the observation that a VNF instance should be placed in a cloud which not only has the most amount of available resources but also is close to that VNF's associated gateway. In other words, the preference on a cloud $n \in N$ varies for different VNFs considering that cloud's residual resource and the delay with a corresponding IoT gateway. Given v-associated gateway g, we define $\mathcal{P}(v,n)$ as the preferential function on n of v as

$$\mathcal{P}(v,n) = l_{g,n}\Phi_n^R\Phi_{g,n}^B\sum_{n'\in H}\Phi_{n,n'}^B \tag{28}$$

Our strategy is illustrated in Algorithm 3 with $f = NULL$ to indicate the case of creating initial state and $f \neq NULL$ for the generation of new states from the current one. If it is the first case, all the VNFs in $V' \equiv V$ will be placed in its most preferential network with $nI = 1$. The randomness of the transition is guaranteed by line 4 where only one VNF v is randomly selected and a random number of most preferential networks (line 7-8) are used to place v whenever the procedure BTRANS is invoked. Each placement of the selected VNF on a chosen cloud, which is not done in the current configuration, is considered as a new state (line 10). Note that in the Markov framework, the procedure BTRANS should be repeatedly performed until all the constraints of the problem (16) are satisfied.

Third, the Markov approximation method can be accelerated by leveraging the presence of multiple edge clouds in IoT network to deploy a distributed implementation which can be done via several approaches. The most common one is based

Algorithm 3 Batching Transition Placement Algorithm

1: **procedure** BTRANS
   ▷ $\mathcal{G}$: network topology , *C*: service requests, *V*: set of VNFs, *f*: current configuration
2: Set $\mathcal{F}' \leftarrow \emptyset$, $V' \leftarrow V, nI \leftarrow 1$
3: **if** $f \neq NULL$ **then**
4: Select a random VNF $v \in V$ and set $V' \leftarrow \{v\}$
5: **for each** $v \in V'$ **do**
6: **if** V' has more than one VNF **then**
7: Set $nI \leftarrow rand\Big(0, min(\kappa_v, \sum_{n\in N} \alpha_{g,n})\Big)$
8: Let N' be the *nI* most preferential clouds of *v* using Equ. (28)
9: **for each** $n \in N'$ **and** *v* not placed on *n* **do**
10: Create a new state from *f* with the placement of *v* on *n* and add it to $\mathcal{F}'$
11: **return** $\mathcal{F}'$

Algorithm 4 Placement Procedure at Master Controller

1: **procedure** MASTERCTRL
   ▷ Δ: state distance threshold
2: $S \leftarrow \emptyset$
3: Generate a batch of $|N|$ feasible states using procedure BTRANS with $f = NULL$
4: **for each** newly generated state *f* **do**
5: Assign *f* to an idle controller
6: **while** listening slave controllers **do**
7: **if** all controllers complete **then**
8: **break**
9: Let S', f_{min} be the set of received states, the state with the lowest cost, respectively
10: $S \leftarrow S \cup \{f_{min}\}$
11: **for each** $f \in S'$ **and** $dist(f_{min}, f) \leq \Delta$ **do**
12: Assign *f* to a randomly idle controller
13: **if** there are still idle controllers **then**
14: Invoke BTRANS to generate new states from f_{min} and assign to idle controllers
15: **return** the minimum cost state in *S*

on partitioning the problem such that the partitions could be run in parallel and then merged. However, this approach is not generalized for the placement problem which may involve different parameters or constraints depending on the applications. Instead, we have controllers at clouds explore the entire solution space in parallel and periodically compare the results. Our basic idea is to extend the basic Markov search strategy using a multi-start and batching approach (MBMAP), instead of performing with only one initial state. The details are provided in Algorithm 4 with two procedures, MASTERCTRL for a master controller (MC) and SLAVECTRL for slave ones (SC). At the beginning of MASTERCTRL, the MC generates a list of feasible states (line 3) and assign them as initially starting states to each idle slave controller (line 4-5). After that, the MC moves to a listening state and waits for data from the SCs at line 6 until receiving a certain number of states. The loop exists if all the SCs complete their tasks (line 7). A set

Algorithm 5 Placement Procedure at Slave Controller

1: **procedure** SLAVECTRL
2: **while** listening master controller **do**
3: Set received state as current state *f*
4: Generate a batch of states from *f* and sort them in cost descending order
5: **for each** f' of the batch **do**
6: **if** *f* not transit to f' **then**
7: Send f' to the master controller
8: **else if** small cost improvement **then**
9: Send f' to the master controller and go back line (2)
10: **else**
11: Go back line (4)

of states which have cost difference less than a threshold Δ are then randomly re-assigned to idle controllers (line 11-12). The lowest cost state f_{min} is also used to generate a batch of new states to assign in case there are still idle SCs. Note that all the potentially "good" states, i.e., f_{min} are tracked by the MC (line 10) and only the one with the lowest cost will be returned at the end of the procedure (line 15). This makes sure that the output is always the best one among those generated by the SCs.

For the SCs in the procedure SLAVECTRL, upon receiving a state *f* from the MC, a batch of states will be created and sorted in cost descending order (line 4). By doing this, we ensure that the SC preferably takes the state with lower cost into account first to perform the transition. There are two cases occurred at the SC's side. If the transition from *f* to f' does not happen, then f' will be reported to the MC (line 7) for the tracking purpose. If the transition does not lead to any significant cost improvement after several times, then the SC will restart its operation with a new state by going back to the listening state (line 2).

B. Node Ranking-Based Placement Heuristic

In order to evaluate MBMAP performance, a node ranking-based placement heuristic (NRP) is proposed. The NRP is developed as a deterministic algorithm based on the BTRAN procedure. In particular, we define the VNF ranking function $\mathcal{R}(v)$ based on the number of VNFs that have connections to *v* regardless the service chain as follow:

$$\mathcal{R}(v) = \frac{1}{\kappa_v} \sum_{c\in C} \sum_{u\in V} (\beta^c_{u,v} + \beta^c_{v,u}) \tag{29}$$

The usage of $\mathcal{R}(v)$ allows the placement process to prioritize VNFs which are more important in terms of the popularity among service chains and the number of instances. NRP procedure is described in Algorithm 6 which starts by constructing an ordered VNF list by $\mathcal{R}$ using Equ. (29). Each VNF *v* is placed one by one (line 4) onto preferable clouds as long as that cloud has enough bandwidth, i.e., $\Phi^R_n > r_n B_v$, $\Phi^B_{g,n} > b^{sen}_v$, $\Phi^B_{n,n'} > b^{out}_v$ as realized by the condition at line 6. The preference $\mathcal{P}(v, n)$ is updated (line 7) after placing a certain VNF. If none of the preferable clouds has enough

Algorithm 6 Node Ranking-Based Placement Algorithm

1: **procedure** NRPLACEMENT
2:   Set $V' \leftarrow \emptyset$
3:   Sort VNFs of V in $\mathcal{R}(V)$-descending order
4:   **for each** $v \in \text{sorted}\,V$ **do**
5:     Set $nI \leftarrow min(\kappa_v, \sum_{n\in N}\alpha_{g,n})$ and let N' be the *nI* most preferential clouds of *v* using Equ. (28)
6:     **for each** $n \in N'$, *n* has enough resource **do**
7:       Place *v* on *n* and update $\mathcal{P}(v, N)$ with *n*'s residual resource
8:     **if** *v* is not placed yet **then**
9:       Set $N' \leftarrow \{N \backslash N'\}$
10:       Go back line 9 if N' is empty
11:       Set $V' \leftarrow V' \cup \{v\}$
12:   **while** stopping criteria is not met **do**
13:     **if** constraints are satisfied **then**
14:       Store current scheme with its cost
15:     **for each** $v \in V'$ **do**
16:       Increase $\mathcal{R}(v)$ by a pre-defined parameter
17:     Go back line 3
18:   **return** scheme with lowest cost stored at line 14

resource to host the VNF, the algorithm continues with other clouds (line 9) as an effort to deploy VNFs. After iterating through all the VNFs, the ranking values of unplaced VNFs, if any, are increased by a pre-defined amount (Section V-C). As a result, such the unplaced VNFs will be more likely placed on suitable clouds. The feasible solution of the problem (16) with its cost is stored at line 13, and the one with the lowest cost will be returned upon meeting the stopping criteria (line 18).

C. Discussion

In general, NRP is simpler to implement than MBMAP as it mainly relies on ranking functions and sorting procedure. The complexity of each iteration in NRP (line 3-11) is contributed by sorting VNFs at line 3, i.e., $O(|C||V|log(|V|))$, the loop at line 4, i.e., $O(|N|\sum_{v\in V}\kappa_v)$. In the worst case, the condition at line 9 is always reached and the complexity of the loop at line 6 is $O(|N|)$. Note that the advantage of NRP lies in its fast convergence speed with the much lower number of feasible states. Its limitation is to easily get stuck in local optimum due to greedily place VNFs until all the constraints are satisfied. A trigger at line in Alg. 6 is not enough to make a significant "jump" regarding the ranking difference between nodes.

From the implementation perspective, several options can be considered for MBMAP, i.e., MC/SC selection, batch's size. In Alg. 4, the MC's operations include, i) to keep track of states generated from the SCs and assign them to other idle SCs and ii) to generate new states only if there are not enough states to assign. The more states a SC generates, i.e., the more processing resource the SC requires, the less possibility the MC invokes BTRANS procedure. In other words, the larger batch of states the BTRANS procedure generates, the less resource the MC requires to manipulate states, the more powerful the SCs are and consequently more resource in total is allocated for controllers since the number of SCs is typically higher than that of MCs. However, regarding the convergence speed, a large batch's size enables MBMAP to explore more candidate solutions and thus faster at discovering the optimal solution. Similarly, there is also a trade-off in setting the number of controllers between the allocated resource and the purpose of driving the algorithm into new regions of the solution space. One way to deal with the parameter is to start with several controllers to encourage the exploration of solutions near a local optimum and add more to push the search out of that local region based on some stopping criteria.

In the worst case, MBMAP might go through the entire space of up to $|\mathcal{E}| = O(2^{\sum_{v\in V}\kappa_v(|C|+|N|)})$ states. Every BTRANS invocation requires $O(1)$ step to transit between two states and $O(|C||N|^2|V|^2)$ steps to validate the new state. As a Markov-based approach, MBMAP is approximated by an entropy term $\frac{1}{\delta}\sum_{e\in\mathcal{E}} p_e log(p_e)$. The gap is therefore computed as $\frac{1}{\delta}log|\mathcal{E}|$, or $O(|V|logM)/\delta$. As pointed out in [22], besides the batch's size and the number of controllers, δ is another parameter that can be adjusted as a trade-off between the requirement of fast convergence as well as small optimality gap and the system performance.

VI. PERFORMANCE EVALUATION

This section presents the performance analysis of proposed model. We assess the applicability of our VNF placement solution by comparing it with other solutions that do not consider IoT network characteristics, i.e., the term $l_{g,n}$ is ignored in the Equ. (28), or the multistart and batching techniques, are not adopted in MBMAP.

A. Simulation Analysis

1) Simulation Settings: We build the simulation with 100 VNFs that are placed onto a fully meshed network topology of 8 clouds and 15 IoT gateways. A VNF can be replicated from 4 to 8 instances. Each gateway is configured to connect to a cloud with a probability 0.8 and is uniformly assigned to handle several sensors of 40 IoT-enabled VNFs. The simulation is performed on service chains with 6 VNFs as illustrated in Section III-C. Each chain consists of a single source node and from 1 to 6 destination nodes. Maximum tolerable service latency is set to 45ms.

From the deployment perspective, input data in terms of traffic rates from sensors is periodically generated at the rate λ_v^{sen} while service requests arrives according to a Poisson distribution with mean λ_c. The NRP algorithm is deployed at only one cloud and its output is applied to other clouds. For MBMAP algorithm, the connections between MCs and SCs are pre-established and maintained during the performance of the algorithm. From such input data, a MC script implements Alg. 1 to initialize a state composing of adjacency matrices that represent the placement of VNFs onto clouds and the assignment of VNFs instances to service chains according to Alg. 4. It also calculates a batch of states by using BTRANS procedure and send them to SCs whenever there are idle SCs. A script at the SC performs a basic procedure combining with

TABLE II
SIMULATION PARAMETERS

Parameter	Value
Number of VNFs	70
Number of clouds	8
Number of IoT gateways	15
VNF instances (κ_v)	(4, 8)
Service arrival rate λ_c (ms^{-1})	(0.1, 0.9)
Service rate $\mu^n_{v_i}$, μ_g (ms^{-1})	(0.1, 0.3)
Sensor data rate λ^{sen}_v (ms^{-1})	(0.5, 1.0)
Bandwidth b^{out}_v & b^{sen}_v (Mbps)	(1.5, 2.5) & (0.3, 0.7)
Link latency l^n_g & $l^{n'}_n$ (ms)	(1.4, 0.02) & (1, 0.02)

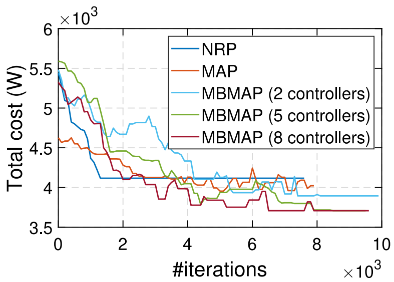


Fig. 6. Evaluation of convergence of proposed algorithms.

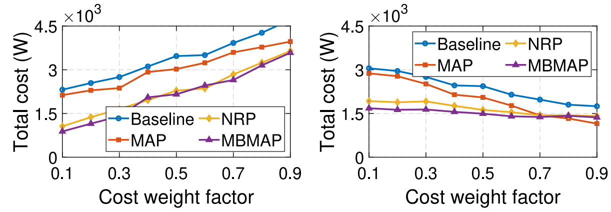
a batching technique and sends back to the MC the state with the lowest cost using Alg. 5.

To evaluate the effect of the IoT network on VNF placement decision making, we define an IoT density as the ratio between the number of IoT-based VNFs and the total number of VNFs. We consider two density levels, i.e., low, and high with the ratios 0.1, 0.7 respectively. Simulation parameters are summarized in Table II. The value of r_n ranges between 2 and 4. System cost is considered from the aspect of power consumption (Watt). According to [23], the power consumption by a router port supporting 1Gbps connection speed is about 21.25W and 11.25W to run a CPU per hour. For the purpose of comparison, normalized unit costs of computing resource and bandwidth are set to 1W and 2W respectively.

B. Simulation Results

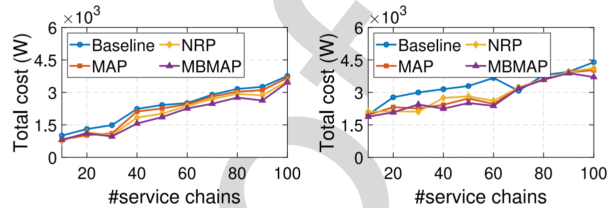
We next present our simulation results on our proposed MBMAP framework and NRP from three aspects, namely convergence time, system cost and resource utilization. The simulation is performed through time slots during which the controllers receive different service demands and makes a decision of placing VNFs. It is assumed that during each slot, system configuration parameters, e.g., network topology, physical/virtual node settings, etc., remains unchanged. The algorithm is assumed to converge during this slot and the deployment of VNFs is performed in the remaining time of the slot.

1) Convergence: We investigate the convergence of the proposed algorithms including MBMAP with different numbers of controllers, NRP and basic MAP. Fig. 6 shows that NRP converges very fast and returns the solution after several iterations. This is due to NRP mainly depends on ranking



(a) Computation resource cost (b) Network traffic cost

Fig. 7. Cost component comparison with different cost weight factors.



(a) Low-density environment (b) High-density environment

Fig. 8. Cost comparison with different level of IoT density.

functions to retrieve an optimal placement. In contrast, it takes more time for Markov-based approaches, i.e., MBMAP and MAP, to converge toward an optimal result, especially with a large space of states. Unlike MAP, MBMAP leverages the presence of multiple controllers at each IoT edge clouds to implement the multistart and batching technique. It not only allows MBMAP to explore more potential states but also prevents MBMAP to get trapped forever at a locally optimal solution. As a result, our proposed mechanism can converge faster than MAP within 500 iterations and approximate the optimal solution as the number of controllers increases.

2) System Cost: In Fig. 7, we run all the algorithms on 100 service chains by varying ω from 0.1 to 0.9 to see how cost components, i.e., M^{net}, M^{com} are affected. The results show that NRP and baseline have more impact on the computation cost and as a result, the improvement of the network cost is very limited even when emphasizing the importance of the network traffic cost (i.e., $\omega = 0.1$). This is due to NRP and the baseline relies on the function $\mathcal{P}$ which is attributed more by the computation resource than the network resource. MBMAP and MAP jointly control the computation cost and the network traffic cost in a more dynamic way and therefore obtain the lower total cost in all considered cost importance. From Fig. 7, the approaches obtains the balance between computing and bandwidth cost at various ω, i.e., 0.3, 0.32, 0.33 and 0.35 for the baseline, MAP, NRP and MBMAP respectively. Regarding the difference between cost components, these ω's values can be seen as Pareto optimal solutions. However, for the purpose of simplicity, we set ω to the average value 0.33 so that the cost difference incurred by different approaches is not significant to avoid extreme cases.

We next evaluate the total cost incurred by using placement approaches given different parameters, i.e., the number of service chains or service arrival rate λ_c, IoT density levels and cost's weight factor ω. In Fig. 8, the MAP approach adopts the standard Markov-approximation framework as described in Section V-A2 and the baseline is a ranking-based heuristic like

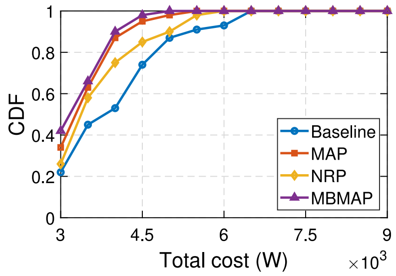


Fig. 9. Distribution of system cost in various service rates.

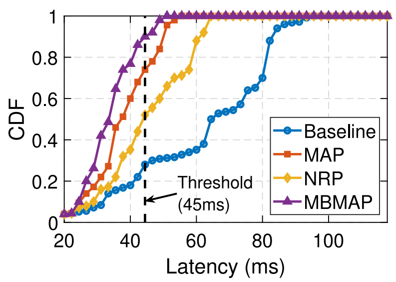


Fig. 10. Distribution of service latency in various service rates.

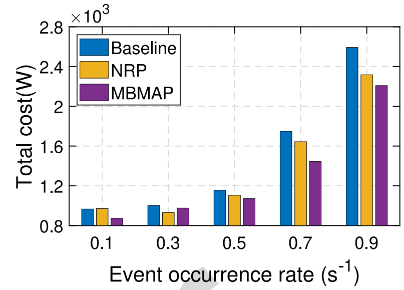


Fig. 11. Evaluation of total system cost.

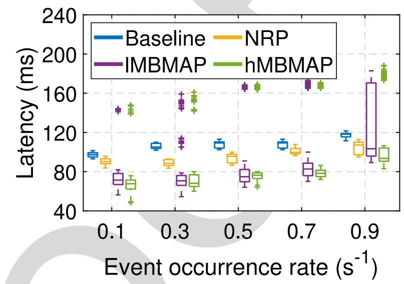


Fig. 12. Evaluation of surveillance session setup latency.

NRP but excludes the delay parameter $l_{g,n}$ from Equ. (28). We can observe that such the exclusion induces a significant gap in system cost between the baseline and the other strategies, especially when more VNFs related to IoT devices present in the system. Three remaining algorithms are comparable to each other, i.e., 10-30 service chains with a negligible cost difference. However, at a high load of more than 40 service chains, MBMAP steps out of the others with a reduction of 13.8% on the total cost. To analyze this difference between the algorithms, we investigate the CDF of system cost across different service demands. Fig. 9 shows that MBMAP overlaps with MAP, which indicates how close these approaches are. From the perspective of Markov chain, it guarantees that the combination of multistart and batching techniques into the original Markov approximation framework does not break Markov property when constructing Markov chain. On the other hand, NRP results in a better cost than the baseline and this matches with the results of component costs in Fig. 7.

3) Service Latency: We perform the analysis under the high-density condition because it is close to the practical environment in which some VNFs are IoT-based entities and some are not. This setting is used in the rest of the paper, except where the differentiation is required. To understand the performance of proposed algorithms on Quality of Service (QoS), we plot the CDF of service latency across different service arrival rate and the number of service chains. As can be seen in Fig. 10, MBMAP and MAP result in better latency than NRP and the baseline. More than 90% service requests are served by MBMAP with latency less than the threshold. For other algorithms, this value is 78% with MAP, 53% with NRP, and 29% with the baseline. Notice that the number of IoT endpoints does not have a significant impact on service latency as the difference of service latency between a 1-target chain and a 6-targets chain is small. This is because the latency is computed as the maximum value among those between source nodes and all the destination nodes. In contrast, the length of service chains affects not only service latency but also demonstrates the improvement of the proposed algorithms. With "longer" service chains, there are likely more instances of each VNF that need to be allocated and therefore resulting to more feasible options of placement (for variables $x^n_{v_i}$) and assignment (for variables $x^c_{v_i}$) instances to a chain even with the same length. Simple heuristics like NRP or the baseline do not leverage this fact to improve their result whereas methods like MBMAP and MAP exploit the introduction of new feasible solutions to obtain a more optimal placement scheme.

C. Experimental Analysis

1) Testbed Settings: The experimental analysis is conducted on the basis of communication sessions between IoT endpoints as shown in Fig. 3. The set N is composed of 8 clouds that are realized by 8 blade servers each of which has 24 physical CPUs and 96GB of memory. VNFs are implemented via Virtual Machines (VMs) configured with different settings depending on the requirements of corresponding service functions, i.e., between 2∼4 virtual CPU (vCPU) and 4∼16 GB virtual memory (vMem) as detailed in Table III. This configuration for heavy tasks is reasonable and has been used in several related works, i.e., [24].

The number of VMs is limited to not cause over 85% CPU usage in order to guarantee the system performance. According to the testbed scenario, the set V is composed of one standalone VNF, i.e., Decision Maker, and five IoT-enabled VNFs. Each VNF has from 1 to 6 instances. Eight controllers in MBMAP are deployed along with VMs for VNFs on all the blade servers.

Toward IoT side, 6 OpenWRT-based Access Points (APs) are set up as IoT gateways that handle traffic from 3 sensors, 2 laptops and 3 mobile phones. A script is deployed at the

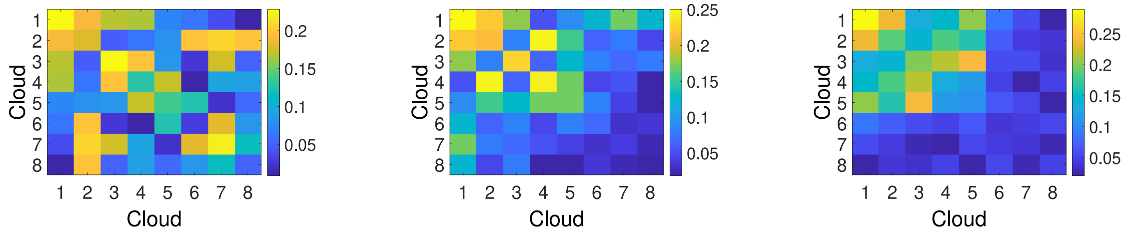


Fig. 13. Evaluation of link utilization.

TABLE III
VNF RESOURCE CONFIGURATION

Service functions	(vCPU, vMem, #instances)
Motion Analyzer	(2, 8, 4)
Video Processor	(4, 16, 3)
Decision Maker	(2, 4, 4)
Dispatcher	(2, 4, 1)
Web Server	(4, 8, 2)
Mobile Proxy	(2, 8, 5)

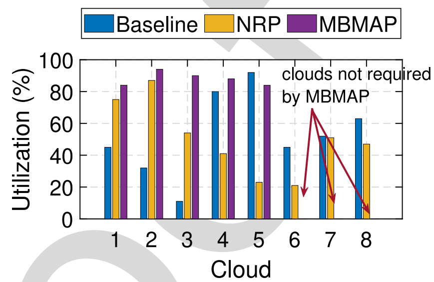


Fig. 14. Evaluation of host resource utilization with fewer clouds to deploy VNFs by MBMAP.

AP to control the transmission of sensor data towards corresponding VNFs. Without affecting the final result, a script is programmed to send data at specific time to represent the occurrence of an intrusion which causes a significant difference of recorded data between two consecutive moments. Network bandwidth between gateways and clouds are pre-configured by APs while the delay is managed by scripts at blade servers.

2) Experimental Results: To show the advantage of our proposed method, we measure the total latency of surveillance service. The event occurrence rate varies from 0.1 to 0.9 according to Poisson distribution during 10 time slots to represent service demand on the network. To show how the convergence of the algorithms affect the overall QoS, we perform the experiments under two conditions, i.e., high and low rate change of occurrence rate, with the duration of time slots 1 and 10 minutes respectively. Accordingly, a proxy is deployed to hold packets from the source node until the placement scheme is obtained by the algorithm. Fig. 11 shows that while MBMAP and NRP obtain scheme at a lower cost, i.e., 11∼21% the baseline. However, in Fig. 12, MBMAP causes a long delay for sessions occurred at the beginning of each slot. In the case of 1-minute slots (hMBMAP), the extremely long sessions represented as outliers significantly affect the latency median and make this value higher a bit than that of 10-minute slots (lMBMAP). Especially, at the event rate $0.9s^{-1}$, lMBMAP results in a higher variation of sessions' delays and more skewed data than other approaches as well as hMBMAP. In addition, at both of time slot's durations, with the execution time approximating 0 due to the small size of input data, NRP and the baseline barely induce any overhead to the hypervisors and therefore the performance of deployed VMs as the MBMAP's controllers do. Note that the MAP algorithm does not appear in Fig. 12 because its performance is comparable to MBMAP regarding the small-scale testbed.

Regarding resource utilization, Fig. 13 represents the link utilization between all pairs of clouds and Fig. 14 illustrate how computing resource is distributed across the clouds (or blade servers). By using the baseline Fig. 13(a), VNFs are placed onto all the 8 clouds and thus entails the utilization of network bandwidth at every link connecting them. In Fig. 13(b), the communication traffic barely go through the links between clouds 7, 8 with clouds 5, 6. In contrast, as shown in Fig. 13(c), most of the network usage is concentrated at some clouds for MBMAP. Unlike prior approaches that try to place VNFs on the same place as much as possible, our algorithm places them in accordance with the impact of the IoT gateways. Correspondingly, computation resource is over-provisioned by the baseline since all the clouds are active but operating with less than 80% allocated resource. For NRP, even though the resource is used more efficiently with more than 85% of resource utilized at clouds 1, 2, and 4, there is a large bias in the amount of virtual resource between them and the other clouds. For example, the 6^{th} cloud needs only 31% CPU usage whereas the 7^{th}, 8^{th} asks for 12% and 9%. On the other hand, MBMAP requires only 5 clouds with a more efficient mechanism of provisioning in such a way that blade servers are fully used with the CPU utilization close to 93%.

VII. CONCLUSION

This paper studies the VNF optimal placement problem in NFV-based edge cloud systems taking IoT network topology into consideration. We consider IoT service chains composed of multiple VNFs that are geographically deployed onto edge clouds close to IoT endpoints. The VNFs communicate not only with each other but also with IoT gateways that typically aggregate data from IoT sensor network as contextual information into discrete messages and forward them toward

VNFs at the server side. We define an analytical model of system cost in terms of computation resource and network bandwidth with regard to service latency and the availability of each resource at edge clouds. We then formulate the problem of minimizing the total system cost with respect to constraints on available resource and QoS requirements. To obtain an optimal placement solution, two algorithms for small and large-scale network settings are proposed respectively, namely a Markov-based approximation approach that leverages the presence of multiple edge cloud to adopt multistart and batching techniques, and a node ranking heuristic.

We implement these two algorithms and validate their performance via simulation and testbed. The testbed is configured according to an IoT-base surveillance use case. The results show that with the consideration of IoT network topology in making VNF placement decision can save on system cost up to 21% depending on the size of the network.

In future, we will take into account the mobility of IoT devices that requires to update the proposed model to reflect the dynamic connection between VNF and IoT gateways. The online placement algorithm in this situation is needed to handle highly dynamic IoT network change.

References

[1] *Network Functions Virtualisation: Architectural Framework*, ISG, ETSI, Sophia Antipolis, France, Oct. 2013.

[2] D. Miorandi, S. Sicari, F. D. Pellegrini, and I. Chlamtac, "Internet of Things: Vision, applications and research challenges," *Ad Hoc Netw.*, vol. 10, no. 7, pp. 1497–1516, 2012.

[3] R. Cziva, C. Anagnostopoulos, and D. P. Pezaros, "Dynamic, latency-optimal vNF placement at the network edge," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Honolulu, HI, USA, Apr. 2018, pp. 693–701.

[4] D. Li, P. Hong, K. Xue, and J. Pei, "Virtual network function placement considering resource optimization and SFC requests in cloud datacenter," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 7, pp. 1664–1677, Jul. 2018.

[5] S. Agarwal, F. Malandrino, C.-F. Chiasserini, and S. De, "Joint VNF placement and CPU allocation in 5G," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Honolulu, HI, USA, Apr. 2018, pp. 1943–1951.

[6] G. Zheng, A. Tsiopoulos, and V. Friderikos, "Optimal VNF chains management for proactive caching," *IEEE Trans. Wireless Commun.*, vol. 17, no. 10, pp. 6735–6748, Oct. 2018.

[7] M. Dieye *et al.*, "CPVNF: Cost-efficient proactive VNF placement and chaining for value-added services in content delivery networks," *IEEE Trans. Netw. Service Manag.*, vol. 15, no. 2, pp. 774–786, Jun. 2018.

[8] M. Mechtri, C. Ghribi, and D. Zeghlache, "VNF placement and chaining in distributed cloud," in *Proc. IEEE 9th Int. Conf. Cloud Comput. (CLOUD)*, San Francisco, CA, USA, Jun./Jul. 2016, pp. 376–383.

[9] M. Ghaznavi, N. Shahriar, S. Kamali, R. Ahmed, and R. Boutaba, "Distributed service function chaining," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2479–2489, Nov. 2017.

[10] M. Habibi, M. Fazli, and A. Movaghar, "Efficient distribution of requests in federated cloud computing environments utilizing statistical multiplexing," *Future Gener. Comput. Syst.*, vol. 90, pp. 451–460, Jan. 2019.

[11] T. Taleb, M. Bagaa, and A. Ksentini, "User mobility-aware virtual network function placement for virtual 5G network infrastructure," in *Proc. IEEE Int. Conf. Commun. (ICC)*, London, U.K., Jun. 2015, pp. 3879–3884.

[12] A. Laghrissi, T. Taleb, M. Bagaa, and H. Flinck, "Towards edge slicing: VNF placement algorithms for a dynamic & realistic edge cloud environment," in *Proc. IEEE Glob. Commun. Conf. (GLOBECOM)*, Singapore, Dec. 2017, pp. 1–6.

[13] L. Gong, H. Jiang, Y. Wang, and Z. Zhu, "Novel location-constrained virtual network embedding LC-VNE algorithms towards integrated node and link mapping," *IEEE/ACM Trans. Netw.*, vol. 24, no. 6, pp. 3648–3661, Dec. 2016.

[14] H. Cao, L. Yang, and H. Zhu, "Novel node-ranking approach and multiple topology attributes-based embedding algorithm for single-domain virtual network embedding," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 108–120, Feb. 2018.

[15] T.-W. Kuo, B.-H. Liou, K. C.-J. Lin, and M.-J. Tsai, "Deploying chains of virtual network functions: On the relation between link and server usage," in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, San Francisco, CA, USA, Apr. 2016, pp. 1–9.

[16] Z. Allybokus, N. Perrot, J. Leguay, L. Maggi, and E. Gourdin, "Virtual function placement for service chaining with partial orders and anti-affinity rules," *Networks*, vol. 71, no. 2, pp. 97–106, 2018.

[17] D. Bhamare, M. Samaka, A. Erbad, R. Jain, L. Gupta, and H. A. Chan, "Optimal virtual network function placement in multi-cloud service function chaining architecture," *Comput. Commun.*, vol. 102, pp. 1–16, Apr. 2017.

[18] L. Gupta, M. Samaka, R. Jain, A. Erbad, D. Bhamare, and C. Metz, "COLAP: A predictive framework for service function chain placement in a multi-cloud environment," in *Proc. IEEE 7th Annu. Comput. Commun. Workshop Conf. (CCWC)*, Las Vegas, NV, USA, Jan. 2017, pp. 1–9.

[19] J. F. Riera *et al.*, "Tenor: Steps towards an orchestration platform for multi-PoP NFV deployment," in *Proc. IEEE NetSoft Conf. Workshops (NetSoft)*, Seoul, South Korea, Jun. 2016, pp. 243–250.

[20] M. C. Luizelli, L. R. Bays, L. S. Buriol, M. P. Barcellos, and L. P. Gaspary, "Piecing together the NFV provisioning puzzle: Efficient placement and chaining of virtual network functions," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manag. (IM)*, Ottawa, ON, Canada, May 2015, pp. 98–106.

[21] J. Halpern and C. Pignataro, "Service function chaining (SFC) architecture," IETF, RFC 7665, Oct. 2015.

[22] M. Chen, S. C. Liew, Z. Shao, and C. Kai, "Markov approximation for combinatorial network optimization," *IEEE Trans. Inf. Theory*, vol. 59, no. 10, pp. 6301–6327, Oct. 2013.

[23] L. Nonde, T. E. H. Elgorashi, and J. M. H. Elmirgahni, "Virtual network embedding employing renewable energy sources," in *Proc. IEEE Glob. Commun. Conf. (GLOBECOM)*, Washington, DC, USA, Dec. 2016, pp. 1–6.

[24] J. Wang, J. Pan, and F. Esposito, "Elastic urban video surveillance system using edge computing," in *Proc. Workshop Smart Internet Things (SmartIoT)*, San Jose, CA, USA, 2017, pp. 1–6.

Duong Tuan Nguyen received the B.S. degree in information technology from the University of Science-Vietnam National University, Ho Chi Minh City, Vietnam, in 2009, and the M.S. degree from Soongsil University, South Korea, in 2013. He is currently pursuing the Ph.D. degree with the École de technologie supérieure, University of Quebec, Montreal, Canada. His current research interests include smart multimedia service and network function virtualization.

Chuan Pham received the B.S. degree in electrical and computer engineering from the Ho Chi Minh City University of Transport in 2004, the master's degree in electrical and computer engineering from the Ho Chi Minh City University of Science in 2008, and Ph.D. degree in electrical and computer engineering from Kyung Hee University in 2017, where he was a Post-Doctoral Fellow with Department of Computer Science and Engineering from August 2017. Since 2018, he has been a Post-Doctoral Fellow with Synchromedia-École de Technologie Supérieure, Université du Québec. His research interest is applying analytic techniques of optimization and machine learning to network applications in terms of cloud and mobile-edge computing, datacenters, resource allocation for virtual networks, and Internet of Things.

Kim Khoa Nguyen received the Ph.D. degree in electrical and computer engineering from Concordia University. He is an Associate Professor with the Department of Electrical Engineering, École de technologie supérieure, University of Quebec, Montreal, Canada. He served as a CTO of Inocybe Technologies (currently, Kontron), Canada, a leading company in software-defined networking solutions. He was an Architect of the Canarie's GreenStar Network and also involved in establishing CSA/IEEE standards for green ICT. He has led research and development in large-scale projects with Ericsson, Ciena, Telus, and InterDigital. He has authored 60 publications, and holds several industrial patents. His expertise includes network optimization, cloud computing, IoT, big data, machine learning, smart city, high speed networks, and green ICT. He was a recipient of the Microsoft Azure Global IoT Contest Award in 2017 and the Ciena's Aspirational Prize in 2018.

Mohamed Cheriet received the Ph.D. degree in computer science from the University of Pierre et Marie Curie (Paris VI). He is a Full Professor with the Department of Automation Engineering, École de technologie supérieure, University of Quebec, Montreal, Canada. He has published over 300 technical papers in renowned international journals and conferences. He has authored and published 6 books on pattern recognition, document image analysis and understanding, and computer vision. He was a recipient of the 2016 IEEE Canada J. M. Ham Outstanding Engineering Educator Award, the 2013 ÉTS Research Excellence Prize, and the 2012 Queen Elizabeth Diamond Jubilee Medal for outstanding contribution to knowledge improvement in computational intelligence and mathematical modeling for image processing. He has delivered over 50 invited talks. He holds NSERC Canada Research Chair Tier 1 in Sustainable Smart echo-Cloud. He is a fellow of IAPR, CAE, and EIC.