

Received April 20, 2020, accepted April 28, 2020, date of publication May 4, 2020, date of current version May 19, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2991986

Virtual Network Embedding With Dynamic Speed Switching Orchestration in Fog/Edge Network

**YAO CHIANG¹, (Student Member, IEEE), YU-HSIANG CHAO², CHIH-HO HSU¹,
CHUN-TING CHOU², (Member, IEEE), AND HUNG-YU WEI^{ID1}, (Senior Member, IEEE)**

¹Graduate Institute of Electrical Engineering, National Taiwan University, Taipei 10617, Taiwan

²Graduate Institute of Communication Engineering, National Taiwan University, Taipei 10617, Taiwan

Corresponding author: Chun-Ting Chou (chuntingchou@ntu.edu.tw)

This work was supported in part by the Foxconn, and in part by the Ministry of Science and Technology (MOST) of Taiwan under Grant 108-2221-E-002-033-MY3 and 108-2218-E-002-060.

ABSTRACT In the future 5G networks, network deployment flexibility and low network latency are two of the most critical requirements and issues. Recently, network virtualization and Fog/Edge computing have been proposed as two potential solutions to enable the desired future network environment. This paper investigates the Virtual Network Embedding (VNE) problem in a Multi-access Edge Computing (MEC) architecture, according to the standards proposed by European Telecommunications Standards Institute (ETSI). We propose an embedding algorithm, called PSO-CSNR, to optimize end-to-end latency constraints in an MEC network. In addition, we adopt Activity on Vertex (AOV) network as our Virtual Network Request (VNR), which is more realistic to real applications. Moreover, we consider the latest processor technologies for substrate nodes, where the CPUs are deployed with asymmetric core frequencies, and propose the second algorithm, called DSS. The DSS can dynamically orchestrate the processing speed of each virtual function, in order to decrease the processing time of virtual functions on virtual nodes, so that the Infrastructure Providers (InPs) can gain more profit in the same amount of time. We then combine the PSO-CSNR with DSS, and refer to it as VNE-DSSO. The simulation results show that the VNE-DSSO algorithm outperforms the other existing algorithms in terms of revenue, acceptance ratio and embedding cost.

INDEX TERMS Network function virtualization, virtual network embedding, multi-access edge computing, asymmetric frequency core, dynamic speed switching.

I. INTRODUCTION

With the explosive growth of smartphones and social media, the network traffic has grown exponentially and it is expected to increase by 40-fold over the next five years, according to a white paper of Cisco System [1]. The enormous data transmitting between the data centers and users causes traffic bottlenecks in the core and backhaul networks. Thus, the users' Quality of Experience (QoE) sharply declines, and real-time applications become hard to be implemented. Moreover, due to the increasing of the multimedia services and heterogeneous applications with diverse scenarios and requirements, network operators are taking enormous efforts to keep up with these demands. However, the gap between traffic growth and revenues produced by the operators becomes greater and greater [2]. This reduces the operators' willingness to upgrade their network facilities, and further causes stagnation to the

whole network. To take down these problems, Fog/Edge computing and network virtualization are regarded as game changers and two of the most promising technologies for the future 5G networks.

Proposed by ETSI, MEC is an emerging technology to cope with a large number of low-latency requests in a Fog/Edge network, ensuring users' QoE under limited back-haul traffic capacity [3]. MEC reduces both network latency and cloud resource demands by offloading computing and storage capacities from the Internet cloud to places close to end users. Applications can be directly hosted on MEC servers operated by InPs. These MEC servers can be located in different places in the network edge, such as base stations or smart cells [4]. MEC servers are able to receive requests from the end users, and offer services or information locally in order to provide a low-latency environment. Network virtualization is another promising technology for future networks. In a network virtualization environment, multiple Service Providers (SPs) are able to create Virtual Network

The associate editor coordinating the review of this manuscript and approving it for publication was Zehua Guo .

(VNs), and each VN offers a customized service in a specific scenario [5]. The VNs consist of several virtualized network functions, called virtual nodes. These virtual nodes are connected by virtual links, which require bandwidth resources. SPs hand these VNs to InPs in a VNR format. The VNRs are then mapped on virtualized substrate resources (e.g. CPU, bandwidth), deployed and maintained by one or more InPs. The network resources are virtualized and isolated from other users on the same physical infrastructure. This increases the network flexibility, and mitigate the ossification of the physical network [6]. Mapping multiple virtual networks onto a given Substrate Network (SN) is a major resource allocation challenge in network virtualization and is usually referred to as VNE problem.

Although some previous efforts have been made to design algorithms for the VNE problem with different objectives, most of the studies of the works considered VNE problems in a signal data center environment [7]–[13], or focused on the offline solutions [14]. Those VNE methods cannot be carried out in an MEC network environment, which consists of distributed MEC servers in different edge areas. Besides, the end-to-end latency is the most important issue in an MEC network; however, solutions considering the cloud environment or offline cases fail to apply in real-time MEC applications. The VNRs in most of the previous researches are arbitrary network requests [7]–[17], which consist of nodes and links in a random method. Here, we consider Activity on Vertex (AOV) networks as our VNRs [18], [19], which the nodes are represented by network functions, and they will process one after another. We believe that AOV network VNRs are more realistic to most of the MEC applications, such as video stream and facial detection. Take facial detection application for example, the photo goes through decompressing before the face recognition and then proceeds to the next step if the identity is verified. Additionally, the substrate nodes in most VNE problems are resources, such as CPU or storage, with fixed capacities and symmetric processing speed [7]–[17]. VNRs with different service priorities or different latency requirements will be ignored and treated equally in terms of processing time. The fixed capacities and symmetric processing speed substrate nodes are not suitable for serving heterogeneous virtual requests simultaneously in that they are comprised of latency-sensitive MEC applications and non-latency-sensitive applications. With the emerge of new processor technologies, such as Intel Speed Select Technology (SST) [20], [21], high priority or latency-sensitive workloads are able to be powered up, while lower the processing speed of the other workloads, leading to higher software performance. This kind of concept perfectly meets the characteristic of MEC applications in a virtualized 5G environment or even 6G networks in the future. If InPs can dynamically adjust the processing speed of the substrate nodes for each network request, the overall processing delay will decrease, and the revenue in a long run will increase at the same time.

This paper proposes a novel algorithm for VNE based on dynamic speed switching orchestration in MEC networks. We enhance the architecture in our previous work [22], and consider the MEC network as a three-tier hierarchy architecture, which follows the ETSI standards [3]. This architecture contains an orchestrator, control nodes and compute nodes. The orchestrator is a centralized scheduler, which is responsible for calculating the solutions. And the control nodes manage the virtualized infrastructure and execute the VNE procedure on the compute nodes based on the solution. The previous two-tier architecture, which only deals with a signal edge area, is not able to handle requests from different edge areas because it merely contains a signal control node. Different from the previous architecture, the three-tier hierarchy architecture contains an orchestrator with a global view of every edge area, which allows the InP to manage resources across edge areas. The main idea is to dynamically adjust the processing speed of instructions on virtual nodes to accelerate the overall processing process and increase InP's long term revenue when types of asymmetric working frequency cores are supported. Moreover, we consider AOV format as our VNRs since many of the MEC applications are in AOV networks [18], [19]. In addition, we consider three types of delay that each VNR will experience in the substrate network. To the best of our knowledge, this is the first attempt to propose dynamic speed switching for asymmetric processing speed processors in a VNE-MEC integrated system. Specifically, our contributions in this paper can be summarized as follows:

- 1) We extend the VNE problem into an MEC architecture, which follows the ETSI standards, and propose a novel VNE-DSSO algorithm to efficiently minimize the end-to-end service time for each virtual request.
- 2) We transform the Particle Swarm Optimization (PSO) from continuous to discrete to formulate the VNE problem and consider AOV networks as our VNRs, which are more realistic to real MEC applications.
- 3) We investigate the newest asymmetric processor technologies, which can be adopted to substrate networks. Then, we design the first MEC environment VNE algorithm, which is able to dynamically adjust the processing speed of virtual network functions to accelerate the end-to-end processing time.

The rest of the paper is organized as follows. In Section II, we provide an overview of the related work, including optimizing revenue of InP and network latency in a virtualized environment. Section III provides the network model and the considered VNE problem. In Section IV, we provide details about our VNE-DSSO algorithms. Simulation results of the proposed algorithms are presented in Section V. Finally, we conclude this work in Section VI.

II. RELATED WORK

Different approaches for various objectives have been proposed so far for the VNE problem. In this section, we outline

some of the most related works based on revenue or latency optimization.

A. REVENUE

The authors in [7] proposed a heuristic algorithm that allowed the substrate network to split a virtual link over multiple substrate paths, and it employed path and node migration to periodically re-optimize the utilization of the substrate network. The goal was to maximize the revenue of the InP. Besides, they explored node-mapping algorithms that are customized to common classes of virtual network topologies. In [8], [9], the authors improved [7] with a novel node selection, and details on embedding VN in a cloud environment were presented. Two reliable VNE algorithms were proposed in [15]. The authors took the reliability requirements of end-users into consideration in order to improve the Quality of Service (QoS) and QoE. Besides, the methods would reduce the chance to re-embed the virtual network requests or migrate substrate nodes and links when facility failure happens. According to the simulation results, the algorithms achieve better performance in terms of acceptance ratio and revenue. In [16], the authors proposed a novel embedding model that considered the label, CPU and bandwidth resource constraints. Furthermore, two window-based heuristic algorithms, called VNE-LIA and VNE-iLIA, using the greedy algorithm and the proximity principle were proposed to solve the VNE problem. The simulation experiments showed that the proposed algorithms successfully increased the revenue to cost ratio.

Metaheuristic algorithms for VNE problems have been widely studied [10]–[13]. Some bionic algorithms are redesigned to solve the optimization problem. In [10], the authors proposed the first Ant Colony based algorithm to solve the VNE problem. It aimed at minimizing the usage of substrate resource, in order to minimize the reject rate and maximize the business profit. To reduce the cost, the authors in [11] pointed out that it is important to take the distance message related to links into consideration in the node mapping phase, so that they can reduce the link cost of VN requests. This allows them to accept more requests and gain better business profit. In [12], a unified enhanced particle swarm optimization-based VN embedding algorithm, called VNE-UEPSO, is presented. Besides, a large to large and small to small preferred node mapping strategy is proposed to achieve better convergence and load balance of the substrate network. In [13], the authors proposed the first genetic-based VNE algorithm. According to the simulation results, GA-based algorithm outperforms PSO-based algorithm in terms of revenue.

B. LATENCY

Several studies on optimizing the latency in VNE problem are as follows. The authors of [23] investigated VNE in a LTE-A cellular network with the goal to minimize the total end-to-end delay on the path under different service priorities. In addition, the user mobility effect was taken into account,

making the algorithm suitable for various scenarios in low-latency applications. However, the architecture of MEC is not involved in the structure and the placement of network functions has not been considered. In [24], the authors suggested that link utilization significantly influences queuing delay of routers. As discussed in this paper, there was a trade-off between delay-awareness and cost-efficiency, but no algorithms were proposed. In [2], the authors discussed new challenges for VNE in MEC network. New VNE parameters and optimization objectives specific to the MEC scenarios were analyzed, which were not considered in traditional VNE problems. However, no VNE algorithms for optimizing the MEC network latency were proposed. The work presented in [17] considers the embedding of virtual switches onto substrate switches. The authors proposed an algorithm, called KCL-vSDNE, which considered not only load balance but also controller to switch latency. K-means algorithm was employed to find out candidates of the substrate switches. Simulation results showed that the KCL-vSDNE increased the acceptance ratio while maintaining the latency less than the threshold.

Similar issues have been discussed in the Service Function Chain (SFC) placement problem. A Latency-Aware SFC placement problem is studied in [25]. The authors used integer linear programming techniques to formulate and solve an SFC placement problem. The objective is to minimize the end-to-end latency of requested services, service cost, and service function migration frequency, respectively. A heuristic algorithm has been proposed in order to optimize the problem in large scales. In [18], the authors investigated Virtual Network Function (VNF) SFC scheduling problem. They considered VNF transmission and processing delays and formulated the joint problem of VNF scheduling and traffic steering as a mixed integer linear program. A dynamic virtual link bandwidth allocation was proposed with the objective to minimize the latency of the overall service chain. This allows operators to serve more customers, and consequently increase revenue. A genetic algorithm-based method was developed for large scale problems.

In all the above previous research works, the substrate networks are equipped with fixed capacity CPUs [7]–[17] while asymmetric processing speed processors have not been considered in any existing VNE problems. Moreover, none of the previous works investigate in details how to embed MEC application VNRs with latency constraints in an MEC environment, and how the MEC architecture works in this scenario from end-to-end. Besides, the considered VNRs in [7]–[17] are arbitrary network requests, which may be hard to relate to real applications or services. As mentioned above, these are important technologies and issues that need to be taken into account when applying VNE in the future MEC networks. A list of the related works is presented in TABLE 1. To compare the features with each other, we classified the works according to their objectives, VNR topology, using Fog/Edge or not, optimization methods and processor types. As we can see, most of the works did not apply Fog/Edge and

TABLE 1. Taxonomy of related works.

#	Objective	VNR Topology	Fog/Edge	Method	Processor
[2]	X	Specific network	O	X	Fixed
[7]	Maximize revenue, Maximize acceptance ratio	Arbitrary network	X	Heuristic	Fixed
[8]	Maximize revenue, Maximize acceptance ratio	Arbitrary network	X	Heuristic	Fixed
[9]	Maximize revenue, Maximize acceptance ratio	Arbitrary network	X	Heuristic	Fixed
[10]	Maximize revenue, Maximize acceptance ratio	Arbitrary network	X	Metaheuristic	Fixed
[11]	Maximize revenue, Maximize acceptance ratio	Arbitrary network	X	Metaheuristic	Fixed
[12]	Maximize revenue, Maximize acceptance ratio	Arbitrary network	X	Metaheuristic	Fixed
[13]	Maximize revenue, Maximize acceptance ratio	Arbitrary network	X	Metaheuristic	Fixed
[15]	Maximize revenue, Maximize acceptance ratio	Arbitrary network	X	Heuristic	Fixed
[16]	Maximize revenue, Maximize acceptance ratio	Arbitrary network	X	Heuristic	Fixed
[17]	Maximize revenue, Maximize acceptance ratio Minimize latency	Arbitrary network	X	Heuristic	Fixed
[18]	Minimize latency	AOV chain	X	Heuristic, Brute Force	Fixed
[23]	Minimize latency	Arbitrary network	X	Heuristic	Fixed
[24]	Minimize latency	Arbitrary network	X	X	Fixed
[25]	Maximize revenue, Maximize acceptance ratio	AOV chain	X	Heuristic, Brute Force	Fixed
This work	Maximize revenue, Maximize acceptance ratio Minimize latency	AOV chain	O	Metaheuristic	Asymmetric

AOV design, and the variation of processors were not taken into account.

III. NETWORK MODEL AND PROBLEM FORMULATION

In this section, we first introduce our hierarchy MEC system architecture. Second, the comparison between novel processor technologies and the traditional ones is discussed. Third, we model the substrate network and virtual network request provided by InP and SP, respectively. Fourth, the formulation and definition of the general VNE problem are presented. In the last three subsections, we describe the VNE problem in more detail according to our considered scenario and the proposed algorithm.

A. EDGE SYSTEM ARCHITECTURE

We consider a three-tier hierarchy edge network architecture as shown in Fig. 1 [3]. In this scenario, at the top there is an orchestrator with the global view of the whole edge network. The orchestrator is designed to provide real-time and

rule-driven service orchestration and automation, including the start-up and configuration of virtual and physical network functions. The orchestrator offers access points that allow users and third parties to access in order to subscribe to applications or deploy servers. When the orchestrator receives VNRs from all its clients as inputs, it calculates the VNE solution based on the constraints and the network resource usage. This solution is then handed over to one or more control nodes to implement the VNE solutions. Each control node is responsible for the virtualized infrastructure preparation, and system resource usage information report. It also manages the virtualized infrastructure and the resource of the compute nodes and substrate links in its manage edge area. Each control node is assigned to manage compute nodes in a specific geographical location. Accessing compute nodes in a different edge area directly is usually not allowed [3] due to management efficiency and security issue. The control nodes can be all kinds of Virtualized Infrastructure Manager (VIM), such as OpenStack, Kubernetes and ONAP, which are

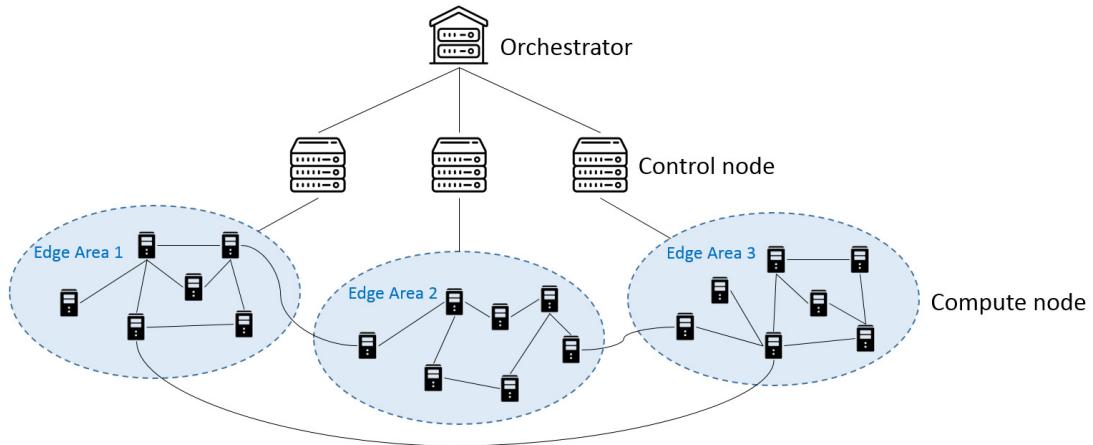


FIGURE 1. Three-tier hierarchy edge network architecture.

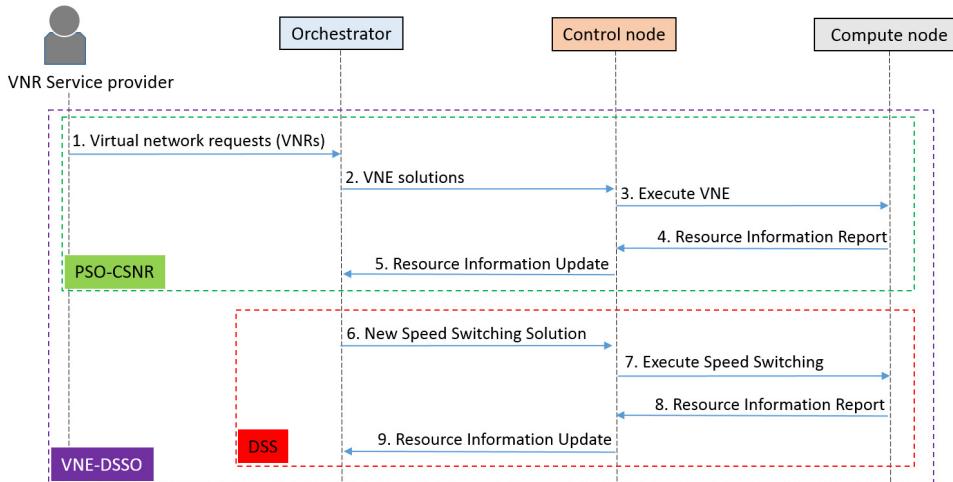


FIGURE 2. Message flow of VNE-DSSO.

composed of functions that are used to control and manage the virtualized infrastructure under the authority of the operator. The reason for splitting the managing jobs into orchestrator and control nodes is because this makes the whole deployment more flexible and efficient. The operator can easily manage and adopt optimization algorithms to control and compute nodes in different edge areas. Compute nodes are the basic processing or resource elements in the edge network, and they can be any kind of storages or processors, such as CPU or GPU. When a virtual node is embedded on the compute node, the compute node starts the working process, such as data storage and/or instruction processing until it is done. In the rest of the paper, compute nodes and networks in the edge area are represented by substrate nodes and substrate networks, respectively.

The message exchange flow between each role in the architecture is shown in Fig. 2. After the VNRs are embedded, all the compute nodes should report resource information to the control nodes and the orchestrator, so that the orchestrator can

find a new solution with a faster processing speed according to our proposed DSS algorithm (in section IV). Then all the resource information should be updated again for the next VNE round.

B. ASYMMETRIC PROCESSING SPEED OF PROCESSORS

In the past, research has mainly focused on fixed or single option CPU capacity [7]–[17]. The tasks, such as software applications and virtualized network functions, worked in the same processing speed or CPU core frequency. Thus, it was inflexible to schedule virtual requests with different priorities. Fig. 3a shows the concept of symmetric core frequency deployment. The task can be represented by any software functions, applications or instruction sets. It could be seen that no matter how high or low priority the requests are, they all function at the same processing speed. Recently, with the introduction of some advanced processor technologies, the overall system workload increases and satisfies the processing time constraints according to their priorities.

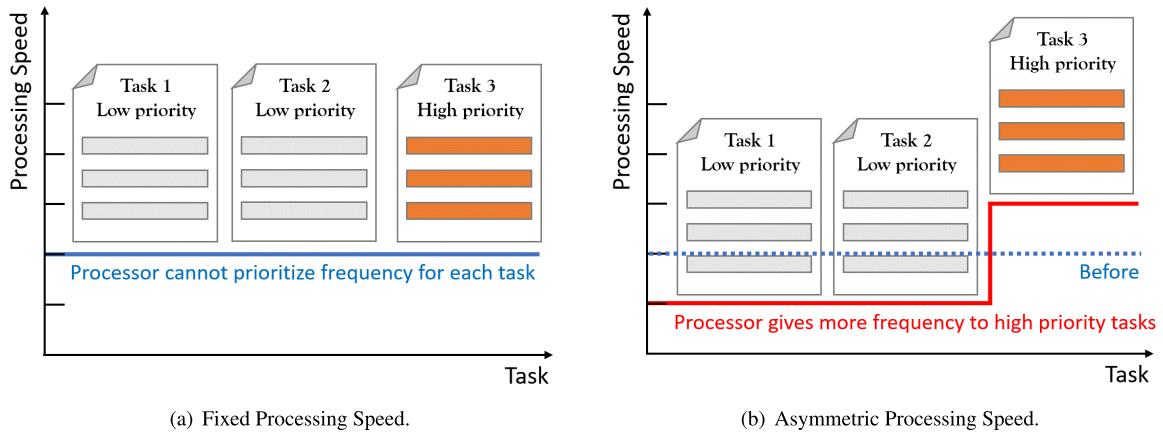


FIGURE 3. Core speed for different priority tasks deployment methods.

As shown in Fig. 3b, the high priority requests are able to run in a much faster speed CPU core, by moving some of the resources from other cores which handles low priority tasks. These technologies increase the system performance while staying within almost the same power consumption [20], [21]. In this work, we adopt the asymmetric processing speed deployment, so the cores in each substrate node can be divided into two sets according to their processing speeds. The two sets of CPU cores are able to deal with high and low priority requests, respectively.

C. SUBSTRATE NETWORK

We denote the topology of the SN by a undirected graph $G^s = (N^s, L^s)$, where N^s is a set of the substrate nodes, and L^s is a set of the substrate links. We use superscript to refer substrate (s) or virtual (v) network, and use subscript to refer to nodes (n) or links (l). Each substrate node $n^s \in N^s$ is associated with attributes, such as processing capacity, storage or location. In this paper, we consider processing capacities and geographic regions for node attributes. The geographic region indicates which edge area the substrate node is located. Besides, different from most of the previous works, where substrate nodes operate at the same speed, as shown in Fig. 3a, we take asymmetric processors into consideration. A substrate node consists of two processing capacity types of cores [20], [21], as shown in Fig. 3b. High processing capacities cores are denoted by *Type-H* and low processing capacities cores are denoted by *Type-D*. The processing capacities can be represented by how much instruction sets can be done per time slot. Here, the instruction set indicates how much workload the VNF has. The set can be kilo instructions, million instructions or even more. The capacities of *Type-H* and *Type-D* are denoted by $C^h(n^s)$ and $C^d(n^s)$, respectively. In addition, the two types of capacities have their own scheduling table, denoted by $AN^h(n^s)$ and $AN^d(n^s)$. They provide the computing resource usage information for each time slot. We assume each slot for an asymmetric processor cannot be shared by multiple requests. Each substrate link $l_{n_i^s, n_j^s}^s \in L^s$ between two substrate nodes n_i^s and n_j^s is associated with attributes,

such as bandwidth, processing capacity or delay. In this paper, we consider processing capacity in terms of megabytes per time slot for link attributes, denoted by $C(l^s)$. In addition, each substrate link has its own scheduling table $AL(l^s)$, and it provides the information of how much bandwidth is occupied at each time slot. We assume each slot cannot be shared by multiple requests in the same bandwidth. We also denote by P^s the set of all loop-free paths in the SN, $P^s(n_s^s, n_d^s)$ and the set of substrate paths from node n_s^s to node n_d^s .

D. VIRTUAL NETWORK REQUEST

Different from substrate network, the topology of the VN is denoted by a directed graph $G^v = (N^v, L^v)$, where N^v is a set of the virtual nodes, and L^v is a set of the virtual links. Each virtual node $n^v \in N^v$ is associated with the number of instruction sets processed on the virtual node, denoted by $D(n^v)$. Each virtual link $l_{n_i^v, n_j^v}^v \in L^v$ from virtual nodes n_i^v to n_j^v is associated with the number of megabytes carried on the virtual link, denoted by $D(l^v)$. The virtual link between each virtual node is responsible for carrying the processed instructions to the next node. General VNE problems consider arbitrary topologies of VNRs and substrate networks [7]–[17]. However, in this study, we focus on chain AOV network because many network applications are in service chain topologies as mentioned previously. These applications could be abstracted into multiple processes such as firewalls, network address translators. The traffic should go through these processes in a pre-defined order. The instructions on a virtual node would start to process after the previous nodes and links end their procedure. The resource is released when the nodes or links finish their procedures. Two types of VNRs are considered in this paper. The first one is VNRs with high priority or latency-sensitive constraints, such as video application or vehicle-to-vehicle communication. The end-to-end latency requirements for these scenarios should be less than 1 ms [26]. This kind of VNRs can only be mapped on high capacity substrate nodes to ensure the requirement. Here, it is reasonable to set the larger capacity for high capacity substrate nodes than the high priority or

TABLE 2. Notation list.

Notation	Definition
G^s	Substrate network
N^s	A set of substrate nodes
L^s	A set of substrate links
$C^h(n^s)$	Capacities of Type-H of substrate node n^s
$C^d(n^s)$	Capacities of Type-D of substrate node n^s
$C(l^s)$	Capacities of substrate link l^s
$AN^h(n^s)$	Scheduling table of Type-H of substrate node n^s
$AN^d(n^s)$	Scheduling table of Type-D of substrate node n^s
$AL(l^s)$	Scheduling table of substrate link l^s
P^s	A set of substrate paths
$R_N(n^s, t)$	Residual capacity of a substrate node n^s at time slot t
$R_L(l^s, t)$	Substrate link l^s available indication at time slot t
$Dur(n^s, k)$	Duration between current time and the first time slot of type k of a substrate node n^s that contains available resources
$Dur(l^s)$	Duration between current time and the first time slot of a substrate link l^s that is available
G^v	Virtual network
N^v	A set of virtual nodes
L^v	A set of virtual links
$D(n^v)$	Number of processing requirements on virtual node n^v
$D(l^v)$	Number of processing requirements on virtual link l^v
$E(n^v, t)$	Number of substrate resources that n^v occupies at time slot t
$BN_{n^s}^{n^v}$	A binary variable, if $BN_{n^s}^{n^v} = 1$ indicates that n^v is mapped on n^s and 0 otherwise.

latency-sensitive node requirements [20], [21]. That is, in a reasonable arrival rate, the proposed VNE-DSSO algorithm would find an embedding solution for the high priority or latency-sensitive node requests, and the high capacity substrate nodes are able to complete all the requests under such latency requirements. Thus, in this case, the request is accepted. If the arrival rate is too high and the physical resources are not sufficient to serve all the requests, the orchestrator may not be able to find a proper solution to meet the requirement for the request, then the VNR would be rejected. Second, VNRs with low priority or non-latency-sensitive constraints, such as some IoT applications, can only be mapped on low capacity substrate nodes initially. If sufficient physical resources are to serve the request, then the VNR is accepted and vice versa.

E. SUBSTRATE NETWORK RESOURCE MEASUREMENT

We denote by $R_N(n^s, t)$ the residual or the available capacity of type k of a substrate node $n^s \in N^s$ at time slot t as

follows:

$$R_N(n^s, t) = C^k(n^s) - \sum_{\forall n^v \uparrow n^s} E(n^v, t) \quad (1)$$

where $n^v \uparrow n^s$ denotes that the virtual node n^v is mapped on the substrate node n^s . All of the mathematical symbols and operators in this work are listed in TABLE 3. $E(n^v, t)$ represents the number of resources that n^v occupies on n^s at time slot t . The sum of $E(n^v, t)$ during the instruction processing time (for example, from time slot t_1 to t_2) of n^v is equal to the instructions requirement $D(n^v)$ as shown in (2). Besides, the duration between current time and the first time slot of type k of a substrate node $n^s \in N^s$ that contains available resources can be denoted by $Dur(n^s, k)$.

$$D(n^v) = \sum_{t_1}^{t_2} E(n^v, t) \quad (2)$$

The capacity of a substrate link $l^s \in L^s$ is a binary value indicating if the substrate link is occupied at time slot t or not, denoted by $R_L(l^s, t)$ as follows:

$$R_L(l^s, t) \in \{0, 1\} \quad (3)$$

Similarly, the duration between current time and the first time slot of substrate link l^s that contains available resources can be denoted by $Dur(l^s)$. All the VNE notations are listed in TABLE 2.

F. VNE

The virtual embedding problem is defined as a mapping M from G^v to a subset of G^s , while the constraints in G^v are satisfied, i.e.,

$$M : G^v \rightarrow (N_{alloc}^s, P_{alloc}^s) \quad (4)$$

where $N_{alloc}^s \in N^s$ and $P_{alloc}^s \in P^s$. Besides, two virtual nodes in the same VNR cannot be mapped on the same substrate node. The VN embedding can be decomposed into two procedures as follows:

1) Node mapping M^n : Each virtual node from a request is mapped to a different substrate node by a mapping $M^n : N^v \uparrow N_{alloc}^s$ as shown in (5), which satisfies the node constraints, such that for all $n^v \in N^v$,

$$\begin{aligned} & M^n(n^v) \in N^s \\ \text{s.t. } & c1. \quad \forall n^s \in N^s, \sum_{n^v \in N^v} BN_{n^s}^{n^v} \leq 1 \\ & c2. \quad \forall n^v \in N^v, \sum_{n^s \in N^s} BN_{n^s}^{n^v} = 1 \\ & c3. \quad \sum_{n^v \in N^v} BN_{n^s}^{n^v} \cdot E(n^v, t) \leq R_N(n^s, t) \end{aligned} \quad (5)$$

The constraints c1. and c2. ensure that every virtual node will be mapped on a substrate node, and each virtual node in the same VNR cannot be mapped on to the same substrate node. The constraint c3. specifies that all the processing requirements on the virtual nodes will not exceed the residual capacity of the substrate node, which they map on.
2) Link mapping M^l : Each virtual link in a request is mapped to a substrate path between the two substrate nodes that the

TABLE 3. Definition of mathematical operators.

Operators	Explanation
\uparrow	Where $n^v \uparrow n^s$ denotes that the virtual node n^v is mapped on the substrate node n^s .
\oplus	In the PSO-CSNR, $p_i v_i \oplus p_j v_j$ indicates that the particle keeps the velocity v_i with a probability p_i and keeps v_j with the probability p_j , where $p_i + p_j = 1$. Where '*' denotes that the value is either 0 or 1 with the corresponding probability.
\ominus	In the PSO-CSNR, $x_i \ominus x_j$ indicates the difference between the two position x_i and x_j . The result value is 1, if x_i and x_j have the same value, otherwise, it is 0.
\odot	In the PSO-CSNR, $x_i^d \odot v_i^d$ indicates the particle i updates its position x_i^d with velocity v_i^d . The result of this operation indicates whether the position in each dimension needs to be adjusted or not. If $v_i^d = 1$, then the value of x_i^d will remain unchanged; otherwise, the value of x_i^d should reselect another substrate node.

two end virtual nodes of that virtual link is mapped on. It is defined by a mapping $M^l : L^v \uparrow P_{alloc}^s$ as shown in (6), which satisfies the link constraints such that for all $l_{n_i^v, n_j^v}^v \in L^v, n_i^v, n_j^v \in N^v$,

$$\begin{aligned} M^l(n_i^v, n_j^v) &\subseteq P^s(M^n(n_i^v), M^n(n_j^v)) \\ \text{s.t. c4.} \quad \forall n_i^s \in N^s, \forall l_{n_u^v, n_w^v}^v \in L^v \\ \sum_{l_{n_i^s, n_j^s}^s \in L^s} f_{l_{n_i^s, n_j^s}^s}^{l_{n_u^v, n_w^v}^v} - \sum_{l_{n_j^s, n_i^s}^s \in L^s} f_{l_{n_j^s, n_i^s}^s}^{l_{n_u^v, n_w^v}^v} \\ &= \begin{cases} 1 & \text{if } BN_{n_i^s}^{n_u^v} = 1 \\ -1 & \text{if } BN_{n_i^s}^{n_w^v} = 1 \\ 0 & \text{otherwise} \end{cases} \quad (6) \end{aligned}$$

where $f_{l_{n_i^s, n_j^s}^s}^{l_{n_u^v, n_w^v}^v}$ is a binary variable, it is 1 if virtual link $l_{n_u^v, n_w^v}^v$ is routed on substrate link $l_{n_i^s, n_j^s}^s$ and 0 otherwise. The constraint c4. ensures that equal amounts of flow due to virtual link $l_{n_u^v, n_w^v}^v$ enter and leave each substrate node that does not correspond to the source n_u^v or destination n_w^v . Moreover, the virtual node n_u^v has an exogenous input of 1 unit of traffic that has to find a path to the substrate corresponding to node n_w^v . Some works proposed to reserve backup resource when doing link mapping to overcome link failure [27]. However, since resources in Fog/Edge are limited compared to cloud-based data centers, our work does not map redundant link resource for each VNR.

G. DELAYS

Three types of delays for each VNR are considered in our work. First, the executing of VNFs or instruction sets for a virtual node on a substrate node will introduce processing delay. Let $T_n(n^v)$ be the instructions processing delay of virtual node n^v as follows:

$$T_n(n^v) = \frac{D(n^v)}{C^k(n^s)} \quad (7)$$

where $C^k(n^s)$ is the processing capacity of the substrate node n^s of type k that the virtual node is mapped on. The processing delay is represented in terms of time slots. The substrate node, where n^v is mapped on, will reserve resources in its scheduling table.

Second, the transmission delays of network services through virtual links are also considered. The packets that have been processed will experience delay as they transmit through a link connecting two substrate nodes. Let $T_l(l^v)$ be the delay of virtual link $l_{n_i^v, n_j^v}^v$ as follows:

$$T_l(l^v) = \sum_{l^s \in P'} \frac{D(l^v)}{C(l^s)} \quad (8)$$

where $D(l^v)$ is the number of megabytes generated after instructions $D(n_i^v)$ have been processed. These bytes will be carried from virtual node n_i^v passing through link $l_{n_i^v, n_j^v}^v$ to node n_j^v . $C(l^s)$ is the number of processing capacity of the substrate link that virtual link l^s maps on. P' is the set of substrate links that the virtual link passes through. The transmission delay is represented in terms of time slots. The substrate path, where l^v is mapped on, will reserve bandwidth and time slots in its scheduling table. Fig. 4 shows an example of mapping a low priority VNR onto a shared SN with asymmetric processors. Because the request is a low priority request, the virtual nodes n_a^v, n_b^v, n_c^v , can only be mapped on substrate nodes' Type-D. If the mapping result is: $n_a^v \uparrow n_A^s, n_b^v \uparrow n_B^s, n_c^v \uparrow n_C^s$. The processing delay of virtual node n_a^v in the example is $T_n(n_a^v) = \frac{D(n_a^v)}{C^d(n_A^s)} = \frac{6}{5}$. Therefore, node n_a^v requires two entire time slots to process. The transmission delay of virtual link $l_{n_a^v, n_b^v}^v$ is $T_l(l_{n_a^v, n_b^v}^v) = \frac{D(l_{n_a^v, n_b^v}^v)}{C(l_{n_a^v, n_b^v}^v)} = \frac{1}{2}$. Therefore, virtual link $l_{n_a^v, n_b^v}^v$ needs one entire time slot to do the transmission. The scheduling tables of the substrate nodes and links that involved in the VNE are shown in Fig. 5 where the number on the time slot represents the remaining node capacity of the processor at that time.

Third, we consider the propagation delays between edge areas as well, which is a critical issue in MEC. We assume that the geographical distance between two edge areas is far enough, such as different states or cities. Therefore the packets that transmit through a substrate link across edges would experience a propagation delay. The propagation delays between edge areas will be much higher than in the same edge area. For simplicity, we assume the propagation delays within the same edge area are small enough to ignore. Let $T_p(l^s)$ be the propagation delay of substrate link l^s as

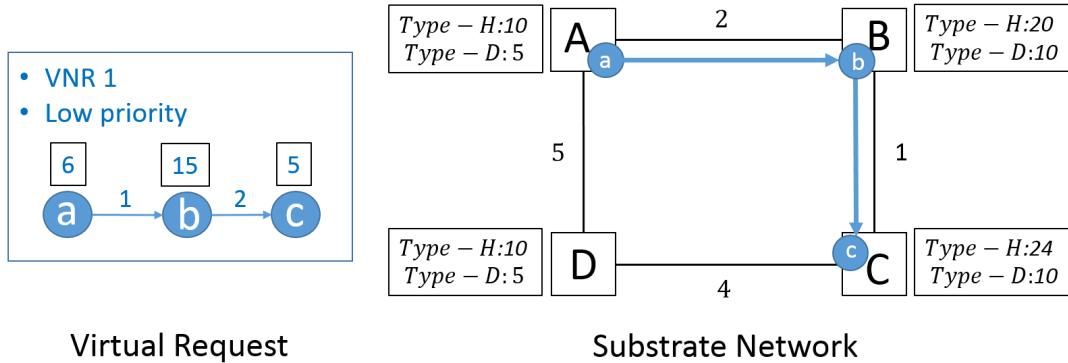


FIGURE 4. An example of embedding a low priority request.

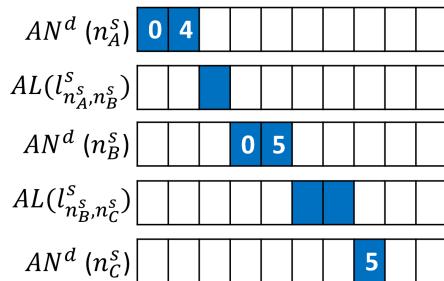


FIGURE 5. Scheduling table of the substrate nodes and links.

follows:

$$T_p(l^s) = \frac{Dis(l^s)}{S} \quad (9)$$

where $Dis(l^s)$ is the geographical length of substrate link l^s which crosses edges, S be the wave propagation speed in the substrate link [24].

H. OBJECTIVE

Our main interest in this paper is to propose a Dynamic Speed Switching (DSS) algorithm for online VNE problem, where VN requests arrive and depart over time. The goal is to minimize the total processing time of each request as well as use the substrate resource efficiently so that the InP is able to accept more requests in the same amount of time, given by the following:

$$\min(\sum_{n^v \in N^v} T_n(n^v) + \sum_{l^v \in L^v} T_l(l^v) + \sum_{l^s \in L_s} T_p(l^s)) \quad (10)$$

I. GOAL AND PERFORMANCE METRIC

The revenue can be defined in various ways according to the needs of the InP [6]. Similar to the previous works [7]–[17], the node and link capacity are the main SN resources in this paper. Thus our revenue model is set as the sum of revenues for virtual links and nodes by the following equation:

$$Rev(G^v) = (w_{c1} \cdot \sum_{n^v \in N^v} D(n^v) + w_{b1} \cdot \sum_{l^v \in L^v} D(l^v)) \quad (11)$$

where w_{c1} and w_{b1} are the charge per CPU capacity unit demand and the charge per link capacity unit demand, respectively.

The cost of embedding a VN request is defined as the sum of the total time slots of substrate nodes and links allocated to the VN as follows:

$$Cost(G^v) = (w_{c2} \cdot \sum_{n^v \in N^v} T_n(n^v) + w_{b2} \cdot \sum_{l^v \in L^v} T_l(l^v)) \quad (12)$$

where w_{c2} and w_{b2} are the cost per time slot for CPU capacity and link capacity, respectively.

IV. PROPOSED VNE-DSSO ALGORITHM

In this section, we propose a batch scheduling-based embedding algorithm based on the speed switching technology. It collects a group of input requests during a time window and then starts to allocate substrate resources to satisfy their requirements. Here we set the time window as one time slot. As shown previously in Fig. 2, the proposed VNE-DSSO algorithm can be divided into two parts. First, we redesign the previous work [12] to create a PSO based algorithm, and add a Greedy Comprehensive Substrate Node Ranking (Greedy-CSNR) to do the node selection. We denote this algorithm by PSO-CSNR. Second, after the mapped is done, a DSS algorithm is proposed to execute the speed switching process.

A. PSO-CSNR

PSO is a stochastic population-based optimization method proposed by Kennedy and Eberhart in 1995 [28]. It is inspired by social behavior and movement dynamics of insect swarms, birds flocking, and fish schooling. In PSO, a swarm of particles is represented as potential solutions, which moves through the solution space, searching for the best solution. Each particle records the best solution it has found so far and shares it with the others. How the position changes for each particle is according to the persistence, which is the direction it was previously going, the best history position it recorded and the best solution shared by all particles. The new position is updated as follows:

$$\begin{aligned} NewPosition &= CurrentPosition + Persistence \\ &\quad + SelfImpact + SocialInfluence \end{aligned} \quad (13)$$

The original PSO can only handle continuous optimization problems [28]. Thus, similar to the previous work [12], we redefined the parameters and operations of the particles in PSO to meet the discrete characteristic of VNE. There is a swarm of particles in the search space. Each particle has its position and velocity. The particles fly in the search space by updating their positions and velocities. The position vector $X_i = (x_i^1, x_i^2, \dots, x_i^D)$ of the particle i denotes a possible VNE solution, where x_i^d is the substrate node that virtual node is mapped on. D is the number of virtual nodes in the request. The velocity vector $V_i = (v_i^1, v_i^2, \dots, v_i^D)$ of the particle i guides the particle to a better solution, where v_i^d is a binary variable. If $V_i^d = 1$, it means that the current VNE solution in x_i^d is not suitable for virtual node and should be adjusted by reselecting another substrate node from its candidate node list; if $V_i^d = 0$, it then remains the current selection. The velocity of each particle can be initialized randomly within the corresponding ranges; whereas, the position vector can be initialized either randomly or with a node ranking system according to the objective [9], [12]. The shortest path in terms of delay between the pair of nodes is selected according to Dijkstra algorithm. A fitness function f is defined to let the particle know whether the new position is more suitable than the previous one or not. It is defined as follows:

$$f(X) = \frac{1}{(\sum_{n^v \in N^v} T_n(n^v) + \sum_{l^v \in L^v} T_l(l^v) + \sum_{l^s \in L_s} T_p(l^s))} \quad (14)$$

The position with the best fitness that the particle i has achieved so far denotes by $pBest_i = (p_i^1, p_i^2, \dots, p_i^D)$. The position with the best fitness in the swarm is denoted by $gBest = (g^1, g^2, \dots, g^D)$. The denominator of the function f is the total delay of the virtual request, which is our objective function. The smaller the total delay of this solution is, the bigger the fitness value is. The particles will try to search for the biggest fitness value. The idea of the fitness function f is to indicate the performance of current solution, and guide the particles to the optimal solution.

During the optimization process, the position and velocity of a particle on dimension are updated according to the concept of equation (13). Here we formulate the concept onto calculable function as follows:

$$v_i^d = w v_i^d \oplus c_1(pBest_i \ominus x_i^d) \oplus c_2(gBest^d \ominus x_i^d) \quad (15)$$

$$x_i^d = x_i^d \odot v_i^d \quad (16)$$

where w is the inertia weight, c_1 and c_2 is the cognition and global weight, respectively. Typically, w , c_1 and c_2 are set to constant values, where $w + c_1 + c_2 = 1$.

Similar to [12], the operations in equation (15) and (16) are redefined for discrete PSO as follows.

Operation "⊕": $p_i v_i \oplus p_j v_j$ indicates that the particle keeps the velocity v_i with a probability p_i and keeps v_j with the probability p_j , where $p_i + p_j = 1$. For example,

$$0.2(1, 0, 0, 1, 1) \oplus 0.8(1, 0, 1, 1, 0) = (1, 0, *, 1, *)$$

where "*" denotes that the value is either 0 or 1 with the corresponding probability.

Operation "⊖": $x_i \ominus x_j$ indicates the difference between the two position x_i and x_j . The result value is 1, if x_i and x_j have the same value; otherwise, it is 0. For example,

$$(1, 3, 5, 7, 9) \ominus (1, 2, 6, 7, 8) = (1, 0, 0, 1, 0)$$

Operation "⊗": $x_i^d \odot v_j^d$ indicates the particle updates its position x_i^d with velocity v_j^d . The result of this operation indicates whether the position in each dimension needs to be adjusted or not. If $x_i^d = 1$, then the value of x_i^d remains unchanged; otherwise, it should reselect another substrate node in its candidate list. For example, $(1, 3, 4, 7, 9) \odot (1, 0, 0, 1, 1)$ means that the solutions of second and third virtual nodes need to be adjusted.

B. GREEDY-CSNR NODE SELECTION

In the traditional PSO, it is common to initialize and update the positions of the particles randomly with equal probability during the optimization process. Although some of the previous works provide measurements to decide the node ranking of substrate nodes [7], [12], they depend on attributes of substrate nodes, such as remaining CPU and bandwidth resource, degree, load. A classic measurement of the resource of a node is proposed, which is called Resource Availability (RA). This measurement can be applied to measure the resource of a substrate or virtual node. They are formulated as follows:

$$RA(n^s) = C(n^s) \sum_{l^s \in L(n^s)} C(l^s) \quad (17)$$

$$RA(n^v) = C(n^v) \sum_{l^v \in L(n^v)} D(l^v) \quad (18)$$

where $L(n^s)$ and $L(n^v)$ are the sets of all the adjacent links of a substrate and virtual node, respectively. However, in our scenario, only considering the resource availability is still problematic. In order to take the resource availability and delays into consideration, we propose a greedy comprehensive node ranking algorithm, which is formulated as follow:

$$CSNR(n^s) = \frac{C^k(n^s)}{Dur(n^s, k) + \sum_{l^s \in L(n^s)} Dur(l^s) + 1} \quad (19)$$

The greedy-CSNR node selection algorithm is a greedy algorithm, which selects substrate node according to CSNR value. The algorithm makes the virtual node with a larger RA value have a higher probability to be mapped on to the substrate node with a larger CSNR value. The two time duration $Dur(n^s, k)$ and $Dur(l^s)$ are placed in the denominator, which means that the smaller total waiting time duration is, the larger the CSNR value is. The substrate node capacity in the fraction means that the larger substrate node capacity is, the larger the CSNR value is. We add 1 in the denominator to ensure the CSNR is a rational number. The greedy-CSNR node selection algorithm is presented in Algorithm 1. With this algorithm, the resource requirement of each request is

Algorithm 1 Greedy-CSNR Node Selection

```

1: Calculate the RA value for every virtual node (or the
   virtual nodes that need to be remapped). Enqueue all
   these virtual nodes according to their RA value to priority
   queue Q in a non-increasing order. The virtual node with
   higher RA value has priority to select substrate nodes.
2: while if Q is not empty do
3:   Dequeue a virtual node  $n^v$  from Q. Select the
      substrate node according to its CSNR value. The
      probability of a substrate node  $n^s$  being selected is
       $CSNR(n^s) / \sum_{n^s \in N^s} CSNR(n^s)$ . The larger CSNR value
      of a substrate node is, the higher probability it will be
      selected.
4:   Remove the substrate node that has just been selected
      to ensure the virtual nodes in a VNR will not be
      mapped onto the same substrate node.
5: end while

```

Algorithm 2 PSO-CSNR**Input:** VNR

```

1: For each particle, adopt Algorithm 1 to initialize the
   position vector  $X$ , and randomly initialize the velocity
   vector  $V$ .
2: Get the fitness value of  $pBest_i$  and  $gBest$  for each particle
   and swarm, respectively according to the fitness function
   (13).
3: while if the max iteration threshold is not satisfied do
4:   for each particle  $i$  do
5:     update the position vector  $X$ , and velocity vector
         $V$  according to equations (15) and (16). The nodes
        which need to be remapped will select their new
        position according to Algorithm 1.
6:   end for
7:   compute the fitness value of each particle according to
   the fitness function (13).
8:   if  $f(X_i) > f(pBest_i)$  then
9:     update  $pBest_i$ 
10:   end if
11:   if  $f(pBest_i) > f(gBest)$  then
12:     update  $gBest$ 
13:   end if
14: end while

```

Output: VNE solution

more likely to be satisfied. Moreover, the processing scheduling delay is considered as well, which helps to minimize the overall delays of each request.

The whole PSO-CSNR contains a greedy-CSNR node selection algorithm to do the node selection and a PSO based update process to find the optimal solution. The detailed PSO-CSNR algorithm is shown in Algorithm 2.

C. VNE-DSSO

After the VNRs are mapped onto the SN based on their requirements and priorities with appropriate-optimal delays,

Algorithm 3 DSS

```

1: Embedding procedure ends
2: for each time slot  $t$  do
3:   for each substrate node  $n_s$  do
4:     if available resources on processor  $n_s$  Type-H not
       equal to 0 then
5:       calculate the number of available resources =  $r_a$ 
6:       for each low priority virtual node  $n_v$  on  $n_s$  do
7:         if virtual node  $n_v$  still occupies CPU resources
           on  $n_s$  after  $t$  then
8:           switch the instructions of  $n_v$  to processor  $n_s$ 
              Type-H
9:           update the remaining  $r_a$ 
10:        end if
11:      end for
12:    end if
13:  end for
14: end for

```

the orchestrator updates the resource usage of the whole network. The asymmetric processors reserve different types of resources for low priority and high priority requests, respectively. The low priority requests are mapped on Type-D with less computing capacities, while high priority requests are mapped on Type-H with more computing capacities to ensure the priority requirement. However, if the arrival rate of the high priority requests is less than that of the low priority requests, the resources in the Type-H is seriously wasted. Therefore, we propose the DSS algorithm to dynamically switch the instruction sets on Type-D to Type-H. This allows the InP to accept more low priority requests and further increase revenue within the same amount of time. As previously shown in Fig. 2, resource usage information is exchanged between each role in the MEC architecture, and the orchestrator has the view of the global resource usage information. First, the orchestrator checks the resource allocation scheduling table after all the VNE procedures end in each time slot. The orchestrator runs the DSS algorithm and will notice that whether the substrate nodes where the virtual nodes are mapped on contain available resources in the processors' Type-H or not. If there are available resources, the orchestrator will inform the control nodes to execute the speed switching procedure, to migrate the instructions on the Type-D to those processors' Type-H. Then, the orchestrator will update the scheduling table of the whole substrate network so that it can calculate the solution for the next arriving request. Fig. 6 shows the concept of how the DSS algorithm works on the VNR1, which we previously explained in Fig. 4. We assume that the substrate network only serves VNR1 currently. Virtual node n_a^v is mapped on substrate node n_A^S 's Type-D, and it will take two time slots to process due to the low processing capacity of Type-D. The orchestrator detects available resources in n_A^S 's Type-H based on DSS algorithm and informs the control node, which manages this area. The five instruction sets, which n_a^v requires

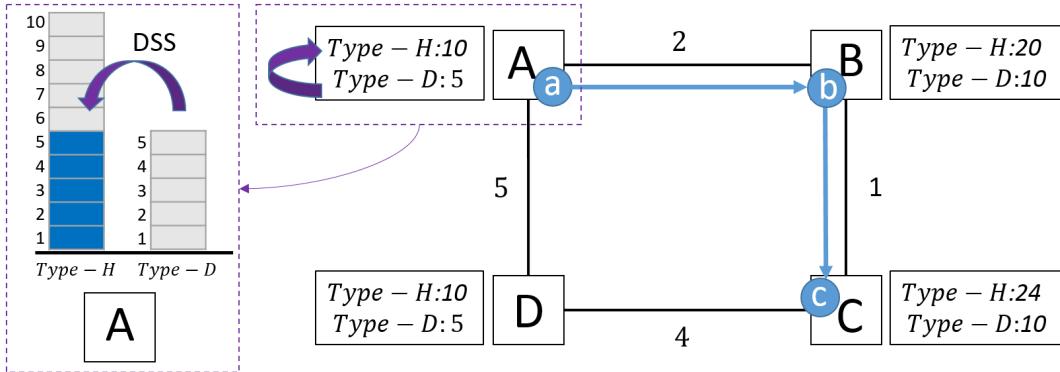


FIGURE 6. An example of the DSS algorithm.

are moved from n_A^s 's *Type-D* to *Type-H*. More importantly, the processing time reduces to one time slot due to sufficient capacity in *Type-H*. We describe the online DSS algorithm in Algorithm 3. Due to higher computing resources in *Type-H*, the instructions of the low priority requests processes faster. This makes the processing time of each virtual node shorter, and releases the resources on *Type-D* at the same time. The released *Type-D* resources can be used by the orchestrator to map other low priority requests in the future. Based on this algorithm, more low priority requests can be accepted due to a great amount of *Type-D* resources are released. In a long run, we believe that the InP can gain more profit. We then combine Algorithm 2 and 3 into VNE-DSSO algorithm, which runs the whole VNE procedure from receiving a VNR to getting a new solution. We describe the whole VNE-DSSO algorithm in Algorithm 4.

Algorithm 4 VNE-DSSO

Input: VNR

- 1: Do PSO-CSNR
- 2: Get VNE solution
- 3: Do DSS

Output: New VNE solution

V. PERFORMANCE EVALUATION

In this section, we first describe the evaluation environment, and then present our evaluation results. Several performance metrics for evaluation purpose are used, including revenue, acceptance ratio and cost. The acceptance ratio is defined as the ratio of the number of accepted VNRs to the total number of arrived VNRs during the measure duration [6]–[17].

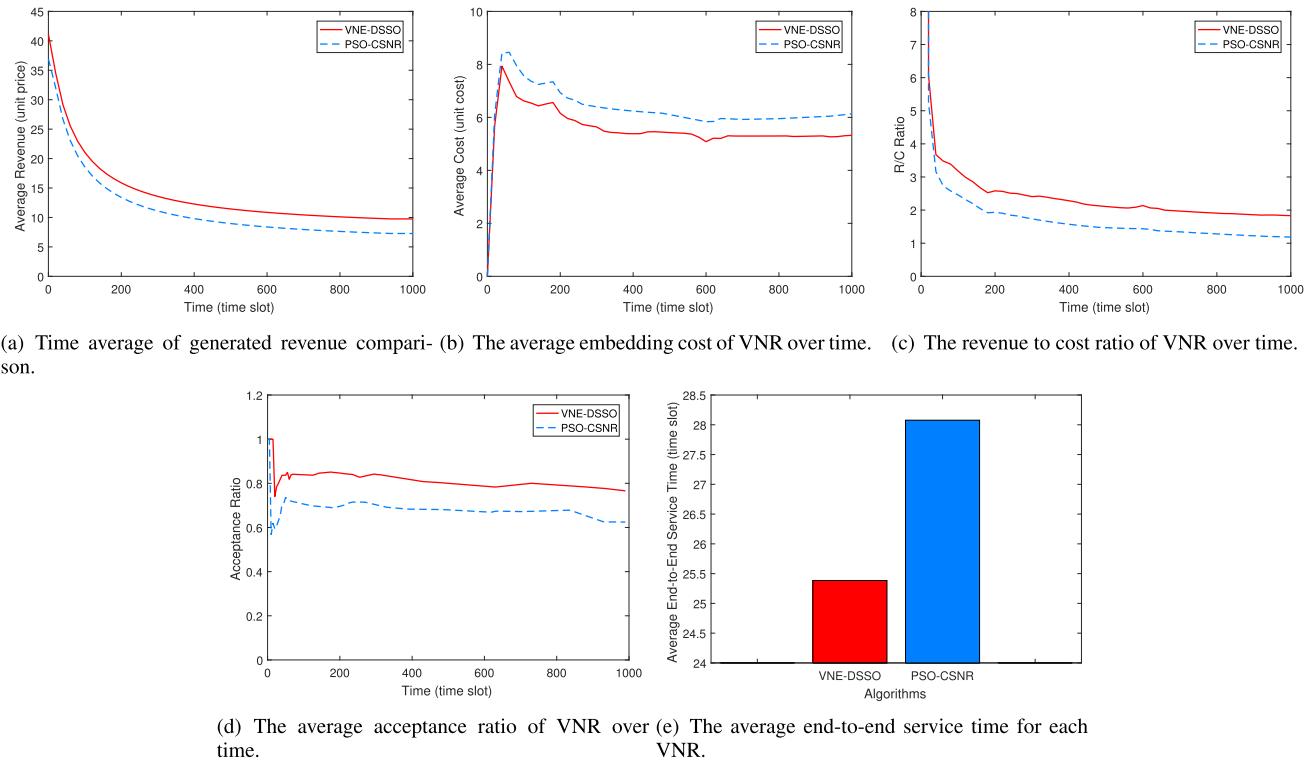
A. SIMULATION SETTINGS

We implement a VNE simulator to evaluate the performance of our algorithms. Similar to the previous works [7], [12], we randomly generate two SN topologies with uniform distribution around 20 to 50 nodes to represent two edge networks and randomly select two nodes from each SN to connect the two edges with a propagation delay of 10 time slots. Each pair of substrate nodes in a SN is randomly connected

with probability of 0.5. The link capacities are real numbers uniformly distributed between 50 and 100. For node capacities, as mentioned in section III, all the substrate nodes are deployed with two operating speeds, as shown in Fig. 2. Closer to reality, we also adopt a type of salable processor in [20] in our simulation, where the node processing capacity of *Type-H* is three times of *Type-D*. Therefore, the node capacities of *Type-D* are real numbers uniformly distributed between 50 and 100, and the node capacities of *Type-H* are real numbers uniformly distributed between 150 and 300. For each VNR, the number of virtual nodes is randomly determined by uniform distribution between 2 and 10. Since the considered VNR is a function chain network, the generated node is arranged in a chain and connected with a virtual link. The node instruction requirements of the virtual nodes are real numbers uniformly distributed between 1 to 20. The link requirements are uniformly distributed between 1 to 20. We assume that VN requests arrive in a Poisson process with mean 5 and 1 requests per 10 time slots for low priority requests and high priority requests, respectively. The high priority VNR would be rejected if the latency is unable to be satisfied. Note that here we set 10 time slots as the latency constraint [26]. The results are derived from the average of 30 simulations.

B. COMPARISON METHOD

In our evaluation, we re-implement three existing VNE algorithms from the previous research [14], [29] to fit in our scenario, and then compare with our PSO-CSNR VNE. Moreover, we compare the four algorithms with adding the VNE-DSSO algorithm. The notations used to refer to different algorithms are shown in Table 4. In [14], the substrate node selection is based on greedy algorithm according to equation (17), (18). The virtual node with a higher RA value has the priority to select a substrate node. Here, we replace the greedy algorithm with the greedy-CSNR node selection algorithm proposed in Section IV. B. to fit in the same scenario. In [29], an energy efficient VNE algorithm is proposed to minimize the energy consumption. The idea is to switch off or to hibernate as many network nodes and interfaces as possible.

**FIGURE 7. Performance comparison between VNE-DSSO and PSO-CSNR.****TABLE 4. Compared algorithm.**

Notation	Description
VNE-DSSO	The proposed dynamic speed switching orchestration algorithm.
PSO-CSNR	The proposed algorithm in section IV without the DSS algorithm.
G-SP-DSS [14]	A redesign of greedy node selection according to (19), with shortest path based link mapping and the DSS algorithm.
G-EE-SP-DSS [29]	Energy efficient node selection with shortest path based link mapping and the DSS algorithm.
R-SP-DSS	Randomized node selection with shortest path based link mapping and the DSS algorithm.

The node selection tends to select the substrate nodes which are already mapped by other VNRs to minimize the active nodes. Finally, the randomized node selection is done by randomly selecting substrate nodes which satisfies the basic VNE constraints.

C. EVALUATION RESULTS

We first compare VNE-DSSO with PSO-CSNR to find out the effect of DSS algorithm. Then, we add the DSS algorithm to the other three algorithms, and compare their performances.

1) WITH AND WITHOUT THE DSS ALGORITHM

Here we compare the proposed PSO-CSNR algorithm against VNE-DSSO. The long-term average generated revenue of both algorithms is presented in Fig. 7a. When DSS is enabled, the InP can generate about 30% more revenue than it is disabled in a long run. The reason is that the orchestrator has the global view of the resource usage of the SN. By switching the instructions on the *Type-D* to *Type-H*, many *Type-D* resources are released, so the scheduling table of these *Type-D* nodes is spreaded with available resources. Although the resources may be fragmented, it still allows InP to accept more low priority requests in the future.

The embedding cost and revenue to cost ratio of VNRs are shown in Fig. 7b, 7c. The embedding cost of VNE-DSSO has decreased by about 15% compared with PSO-CSNR in terms of the defined cost model (12). The VNE-DSSO algorithm allows the idle *Type-H* resources to be used wisely by other low priority VNRs. The reason is that the processing capacity on the *Type-H* is higher than that on the *Type-D*. When switching the instructions initially on the *Type-D* to *Type-H*, the processing time of the instructions on each virtual node may speed up, and further reduces the cost in terms of time slots occupied by the VNR.

In Fig. 7d, we show the comparison of the acceptance ratio. As we can see, the VNE-DSSO has a higher acceptance ratio than PSO-CSNR about 13%, which means at every time slot, the VNE-DSSO has a higher probability to accept more requests. The reason is similar to the above. The scheduling

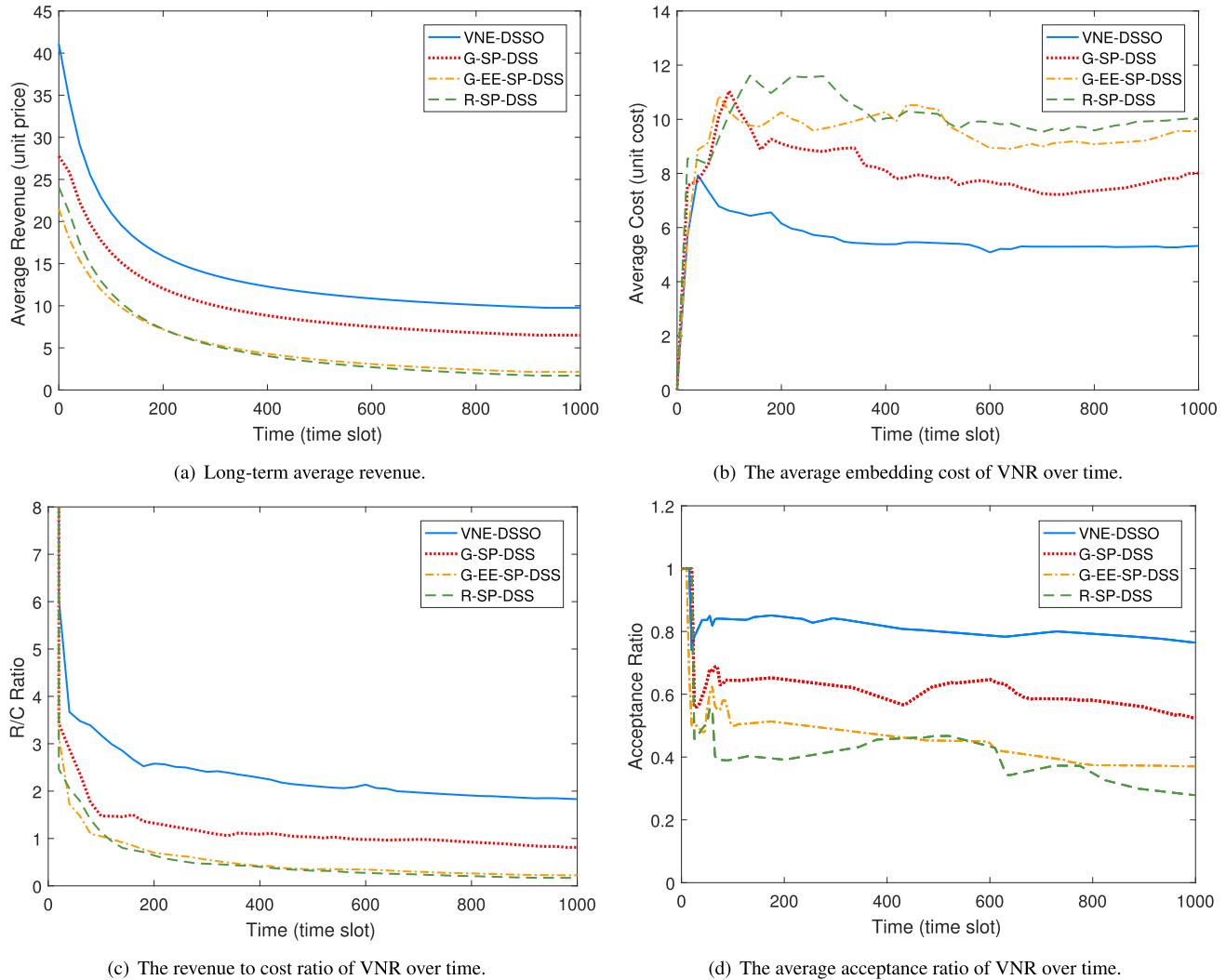


FIGURE 8. The overall performance comparison with the other three algorithms.

delay of the low priority requests is lower if DSS algorithm is added. Therefore, in the same amount of time, VNE-DSSO has the ability to accept more requests. Moreover, having higher revenue to cost ratio along with better acceptance ratio implies that the VNE-DSSO actually embeds VNRs which generate more revenue than small size VNRs just to increase the acceptance ratio.

The average end-to-end service time for each VNR is shown in Fig. 7e. The end-to-end service time is defined by the time when the request arrives to the time when the instruction sets on the last virtual node are completed. It could be observed that by adding the DSS algorithm, the average duration from a VNR arrives to all the instructions processing ends has decreased by about 10%. By dividing the resource in the substrate nodes into Type-H and Type-D, the high priority VNRs can have the privilege to use high speed resources, thus the latency requirement can be ensured. On the other hand, by adopting the DSS algorithm, the low priority VNRs can have the opportunity to use the high speed resources when they are available. This means that we not only

minimize the high priority requests' end-to-end service time but also accelerate the low priority requests' processing time simultaneously.

2) THE PROPOSED VNE-DSSO ALGORITHM HAS A BETTER OVERALL PERFORMANCE COMPARED WITH THE OTHER THREE ALGORITHMS

In Fig. 8a, we illustrate the long-term average generated revenue of each algorithm. The proposed VNE-DSSO generates about 50% more revenue than the second highest algorithm in a long run. We believe that it is because in VNE-DSSO algorithm, the two critical factors, scheduling delay and substrate node capacity, are taken into consideration when executing node selection, which ensures that the scheduling table of each substrate node is load balanced. In addition, VNE-DSSO searches the solution space with an iteration update method in a reasonable time. Each particle exchanges the current global best solution with each other. Once a particle finds a solution better than all the others, the rest of the particles will reference

this solution and try to find an even better one. This kind of self-learning method leads to finding an appropriate-optimal solution.

The average embedding cost and revenue to cost ratio of VNR over time are shown in Fig. 8b and 8c. We can see that the proposed VNE-DSSO enhances the average embedding cost compared with the other algorithms. It decreases the embedding cost about 50% than the G-SP-DSS. The G-SP-DSS selects the substrate nodes for each virtual node with a greedy algorithm according to the present RA value; however, because of the lack of considering the whole VNR's requirement, including nodes and links, the G-SP-DSS algorithm ends up with a higher cost. The cost of G-EE-SP-DSS algorithm is even higher, because the node selection tends to select the substrate nodes which are already mapped by the others to minimize the active nodes. This causes both the scheduling delay of each active node and the end-to-end service time increase, and thus leads to higher cost. On the other hand, our proposed algorithm increases the available resources in the whole SN and allows more VNRs to map on in the same amount of time. Therefore, it consumes fewer resources and generates more revenue in the whole embedding process. Moreover, VNE-DSSO's average cost is nearly constant throughout the simulation.

The acceptance ratio of the proposed VNE-DSSO outperforms the other algorithms, as shown in Fig. 8d. The VNE-DSSO has not only a higher probability to accept a request when it arrives but also the capability to accept more request in a long run due to the self-learning method. It is worth noting that G-EE-SP-DSS algorithm is lower than VNE-DSSO and G-SP-DSS. The reason is that G-EE-SP-DSS tends to map as many virtual nodes on to the active substrate nodes as possible in order to decrease overall power consumption. This makes the scheduling delay on these nodes increase rapidly, and further causes that the InP cannot accept more VNRs in a fixed amount of time. Additionally, it could be seen that the acceptance ratio of the proposed VNE-DSSO is around 0.8 throughout the simulation, which implies that the variation of the arrival rate has little effect on the VNE-DSSO.

VI. CONCLUSION

Network virtualization and Fog/Edge computing are regarded as two of the most promising technologies for the future 5G networks. This paper studies the VNE problem in an MEC architecture, which follows the ETSI standards, when the VNRs are AOV chain application networks. We enhance the previous works and propose a PSO-CSNR algorithm to optimize the virtual chain network embedding problem. Moreover, we take the novel processor technologies into consideration, where high priority workloads are able to speed up, while lower the processing speed of the others. According to these new technologies, we proposed a DSS algorithm for these technologies to dynamically switch the instructions to increase the virtual node's processing speed. We then combine the two proposed algorithms into the VNE-DSSO algorithm. For comparison, we adopt three existing

VNE algorithms and add the DSS algorithm to each of them for the sake of fairness. The simulation results show that the VNE-DSSO outperforms the other algorithms in terms of revenue, embedding cost, and acceptance ratio. Additionally, it reduces the end-to-end processing delay, which allows the operator to accept more requests in the same amount of time. Last but not least, the DSS algorithm can be added to any existing VNE algorithms. For the future work, we will consider link failure in a Fog/Edge and cloud co-exist environment for that cloud may be able to handle some of the low priority requests when failure happens at the Fog/Edge area.

REFERENCES

- [1] Cisco. (2014). *Global Mobile Data Traffic Forecast Update, 2013–2018*. [Online]. Available: <https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-738429.pdf>
- [2] M. T. Beck and M. Maier, "Mobile edge computing: Challenges for future virtual network embedding algorithms," in *Proc. 8th Int. Conf. Adv. Eng. Comput. Appl. Sci. (ADVCOMP)*, 2014, vol. 1, no. 2, pp. 65–70.
- [3] ETSI MEC ISG. (Apr. 2016). *Mobile Edge Computing (MEC); Framework and Reference Architecture*. ETSI, DGS MEC 003. [Online]. Available: http://www.etsi.org/deliver/etsi_gs/MEC/001_099/003/01.01.01_60/gs_MECC003v010101p.pdf
- [4] T.-Y. Kan, Y. Chiang, and H.-Y. Wei, "Task offloading and resource allocation in mobile-edge computing system," in *Proc. 27th Wireless Opt. Commun. Conf. (WOCC)*, Apr. 2018, pp. 1–4.
- [5] J. Li, N. Zhang, Q. Ye, W. Shi, W. Zhuang, and X. Shen, "Joint resource allocation and online virtual network embedding for 5G networks," in *Proc. GLOBECOM IEEE Global Commun. Conf.*, Dec. 2017, pp. 1–6.
- [6] A. Fischer, J. F. Botero, M. T. Beck, H. de Meer, and X. Hesselbach, "Virtual network embedding: A survey," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 4, pp. 1888–1906, 4th Quart., 2013.
- [7] M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking virtual network embedding: Substrate support for path splitting and migration," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 17–29, Mar. 2008.
- [8] W.-H. Hsu, Y.-P. Shieh, C.-H. Wang, and S.-C. Yeh, "Virtual network mapping through path splitting and migration," in *Proc. 26th Int. Conf. Adv. Inf. Netw. Appl. Workshops*, Mar. 2012, pp. 1095–1100.
- [9] W.-H. Hsu and Y.-P. Shieh, "Virtual network mapping algorithm in the cloud infrastructure," *J. Netw. Comput. Appl.*, vol. 36, no. 6, pp. 1724–1734, Nov. 2013.
- [10] I. Fajjari, N. Aitsaadi, G. Pujolle, and H. Zimmermann, "VNE-AC: Virtual network embedding algorithm based on ant colony Metaheuristic," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2011, pp. 1–6.
- [11] J.-B. Wang, W.-N. Chen, H. Cong, Z.-H. Zhan, and J. Zhang, "An ant colony system based virtual network embedding algorithm," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, Oct. 2017, pp. 1805–1810.
- [12] Z. Zhang, X. Cheng, S. Su, Y. Wang, K. Shuang, and Y. Luo, "A unified enhanced particle swarm optimization-based virtual network embedding algorithm," *Int. J. Commun. Syst.*, vol. 26, no. 8, pp. 1054–1073, Aug. 2013.
- [13] X. Mi, X. Chang, J. Liu, L. Sun, and B. Xing, "Embedding virtual infrastructure based on genetic algorithm," in *Proc. 13th Int. Conf. Parallel Distrib. Comput. Appl. Technol.*, Dec. 2012, pp. 239–244.
- [14] Y. Zhu and M. H. Ammar, "Algorithms for assigning substrate network resources to virtual network components," in *Proc. 25th IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Barcelona, Spain, vol. 1200, 2006, pp. 1–12.
- [15] Y. Zhu and M. Ammar, "Algorithms for assigning substrate network resources to virtual network components," in *Proc. IEEE INFOCOM . 25TH IEEE Int. Conf. Comput. Commun.*, vol. 1200, 2006, pp. 1–12.
- [16] P. Zhang, S. Wu, M. Wang, H. Yao, and Y. Liu, "Topology based reliable virtual network embedding from a QoE perspective," *China Commun.*, vol. 15, no. 10, pp. 38–50, Oct. 2018.
- [17] T. Chen, J. Liu, Q. Tang, T. Huang, and R. Huo, "Virtual network embedding algorithm for location-based identifier allocation," *IEEE Access*, vol. 7, pp. 31159–31169, 2019.
- [18] Z. Yan, N. Wei, Q. Jin, and X. Zhou, "Latency-aware resource-efficient virtual network embedding in software defined networking," in *Proc. 28th Wireless Opt. Commun. Conf. (WOCC)*, May 2019, pp. 1–5.

- [19] D. Harutyunyan, N. Shahriar, R. Boutaba, and R. Riggio, "Latency-aware service function chain placement in 5G mobile networks," in *Proc. IEEE Conf. Netw. Softwarization (NetSoft)*, Jun. 2019, pp. 133–141.
- [20] G. Sun, Z. Chen, H. Yu, X. Du, and M. Guizani, "Online parallelized service function chain orchestration in data center networks," *IEEE Access*, vol. 7, pp. 100147–100161, 2019.
- [21] *Optimize Your NFVI Performance With Wiwynn EP100 and Intel Speed Select Technology—Base Frequency*, Intel, Santa Clara, CA, USA, 2019.
- [22] T.-Y. Kan, Y. Chiang, and H.-Y. Wei, "QoS-aware mobile edge computing system: Multi-server multi-user scenario," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Dec. 2018, pp. 1–6.
- [23] G. Chochlidakis and V. Friderikos, "Low latency virtual network embedding for mobile networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2016, pp. 1–6.
- [24] M. T. Beck and C. Linnhoff-Popien, "On delay-aware embedding of virtual networks," in *Proc. 6th Int. Conf. Adv. Future Internet (AFIN)*, 2014, pp. 55–59.
- [25] L. Qu, C. Assi, and K. Shaban, "Delay-aware scheduling and resource optimization with network function virtualization," *IEEE Trans. Commun.*, vol. 64, no. 9, pp. 3746–3758, Sep. 2016.
- [26] NGMN Alliance, "NGMN 5G white paper," Next Gener. Mobile Netw., Frankfurt am Main, Germany, White Paper, 2015, pp. 1–125. [Online]. Available: <https://www.ngmn.org/work-programme/5g-white-paper.html>
- [27] N. Shahriar, S. R. Chowdhury, R. Ahmed, A. Khan, S. Fathi, R. Boutaba, J. Mitra, and L. Liu, "Virtual network survivability through joint spare capacity allocation and embedding," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 502–518, Mar. 2018.
- [28] M. R. Bonyadi and Z. Michalewicz, "Particle swarm optimization for single objective continuous space problems: A review," *Evol. Comput.*, vol. 25, no. 1, pp. 1–54, Mar. 2017.
- [29] J. F. Botero, X. Hesselbach, M. Duelli, D. Schlosser, A. Fischer, and H. de Meer, "Energy efficient virtual network embedding," *IEEE Commun. Lett.*, vol. 16, no. 5, pp. 756–759, May 2012.



CHIH-HO HSU is currently pursuing the B.S. degree in electrical engineering with National Taiwan University (NTU). His research interests include social networks and mobile communications design for multiaccess edge computing systems.



CHUN-TING CHOU (Member, IEEE) received the B.S. and M.S. degrees from National Taiwan University, in 1995 and 1997, respectively, and the Ph.D. degree from the University of Michigan, Ann Arbor, in 2004, all in electrical engineering. From 2004 to 2007, he was a Senior Member Research Staff with Philips Research North America, New York. He is currently an Associate Professor with the Graduate Institute of Communication Engineering, National Taiwan University. His research interests include dynamic spectrum access (DSA), medium access control (MAC) design, wireless and mobile communications, and the Internet of Things (IoT).



HUNG-YU WEI (Senior Member, IEEE) received the B.S. degree in electrical engineering from National Taiwan University, in 1999, and the M.S. and Ph.D. degrees in electrical engineering from Columbia University, in 2001 and 2005, respectively.

He is currently a Professor with the Department of Electrical Engineering and the Graduate Institute of Communications Engineering, National Taiwan University. He also serves as the Associate Chair with the Department of Electrical Engineering. He was a summer intern at Telcordia Applied Research, in 2000 and 2001. He was with NEC Labs America, from 2003 to 2005. He joined the Department of Electrical Engineering, National Taiwan University, in July 2005. His research interests include next-generation wireless broadband networks, the IoT, vehicular networking, fog/edge computing, cross-layer design for wireless multimedia, and game theoretical models for communications networks. He received NTU Excellent Teaching Award, in 2008 and 2018. He also received the Recruiting Outstanding Young Scholar Award from the Foundation for the Advancement of Outstanding Scholarship, in 2006, the K. T. Li Young Researcher Award from ACM Taipei/Taiwan Chapter and The Institute of Information and Computing Machinery, in 2012, the Excellent Young Engineer Award from the Chinese Institute of Electrical Engineering, in 2014, the Wu Ta You Memorial Award from MOST, in 2015, and the Outstanding Research Award from MOST, in 2020. He has been actively participating in NGMN, IEEE 802.16, 3GPP, IEEE P1934, and IEEE P1935 standardization. He serves as the Vice Chair of IEEE P1934 Working Group to standardize fog computing and networking architecture. He serves as a Secretary for IEEE Fog/Edge Industry Community. He also serves as an Associate Editor for the IEEE INTERNET OF THINGS JOURNAL. He is an IEEE certified Wireless Communications Professional. He was the Chair of the IEEE VTS Taipei Chapter, from 2016 to 2017. He is currently the Chair of IEEE P1935 working group for edge/fog management and orchestration standard.



YAO CHIANG (Student Member, IEEE) received the B.S. degree and the M.S. degree in management information systems from the National Pingtung University of Science and Technology (NPUST), Pingtung, Taiwan, in 2014 and 2016, respectively. He is currently pursuing the Ph.D. degree in electrical engineering with National Taiwan University (NTU). His research interests include data mining, machine learning, and mobile communications design for multiaccess edge computing systems.



YU-HSIANG CHAO received the B.S. degree in electronic and computer engineering from the National Taiwan University of Science and Technology (Taiwan Tech), Taipei, Taiwan, in 2018. He is currently pursuing the M.S. degree in communication engineering with National Taiwan University (NTU). His research interests include virtual network embedding and mobile communications design for multiaccess edge computing systems.