Received July 21, 2019, accepted August 8, 2019, date of publication August 13, 2019, date of current version August 26, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2935014

Virtual Network Function Migration Based on Dynamic Resource Requirements Prediction

LUN TANG, XIAOYU HE, PEIPEI ZHAO, GUOFAN ZHAO, YU ZHOU, AND QIANBIN CHEN, (Senior Member, IEEE)
School of Communication and Information Engineering, Chongqing University of Posts and Telecommunications, Chongqing 400065, China
Chongqing Key Laboratory of Mobile Communications Technology, Chongqing 400065, China

Corresponding author: Qianbin Chen (cqb@cqupt.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61571073, and in part by the Science and Technology Research Program of Chongqing Municipal Education Commission under Grant KJZD-M201800601.

ABSTRACT Network function virtualization (NFV) enables flexible deployment of virtual network function (VNF) in 5G mobile communication network. Due to the inherent dynamics of network flows, fluctuated resources are required to embedding VNFs. VNF migration has become a critical issue because of the time-varying resource requirements. In this paper, we propose a real-time VNF migration algorithm based on the deep belief network (DBN) to predict future resource requirements, which resolves the problem of lacking effective prediction in the existing methods. Firstly, we propose optimizing bandwidth utilization and migration overhead simultaneously in VNF migration. Then, to model the resource utilization that evolves over time, we adopt online learning with the assistant of offline training in the prediction mechanism, and further introduce multi-task learning (MTL) in our deep architecture in order to improve the prediction accuracy. Moreover, we utilize adaptive learning rate to speed up the convergence speed of DBN. For the migration, we design a topology-aware greedy algorithm with the goal to optimize system cost by taking full advantage of the prediction result. In addition, based on tabu search, the proposed migration mechanism is further optimized. Simulation results show that the proposed scheme can achieve a good performance in reducing system cost and improving the service level agreements (SLA) of service.

INDEX TERMS Virtual network function, deep belief network, multi-task learning, migration.

I. INTRODUCTION

With the widespread access of mobile terminals and rapid development of internet technologies, there will be a tremendous growth in the amount of mobile transmission data. According to the latest Cisco's prediction, traffic in 2020 will grow faster than 2010 with 1000 times. In addition, network functions (NFs) are deployed in dedicated hardware to setup service function chains offering different services traditionally, which cause heavy operation cost and are difficult to dynamically scale the capabilities to react against time-varying traffic. Therefore, service providers are faced with an urgent need to find innovative and cost-effective ways to achieve better service agility.

Today's mobile network is rapidly evolving into 5G in which ''enhanced Mobile Broadband'', ''massive Machine Type Communications'' and ''Ultra-Reliable and Low Latency Communications'' will play an important role [1]. 5G network is highly flexible to cope with business changes of mobile services based on the technology of network slicing. With the help of software defined network (SDN) and network function virtualization (NFV) technology [2], [3], network slicing is a technique for flexible resource allocation which cuts and regroups limited physical resources to form logically independent virtual network resources for each slice, therefore enabling centralized management and providing better quality for tenants [4]. In a network slice, service requests should be steered to traverse virtual machines implementing the required VNFs in a specific order satisfying service requirement, which means that a service request consists of several different VNFs interconnected by virtual links. We call such an ordered set of VNFs as a service function chain (SFC) [5]. Thanks to NFV, different NFs can evolve independently of hardware which can reduce operating expenditure of operators. Moreover, we can allocate

The associate editor coordinating the review of this article and approving it for publication was Muhammad Khandaker.

virtual network resources efficiently to each SFC by dynamic scaling according to its requirements.

So far, most of existing works such as [6]–[9] studied the deployment of SFCs to achieve efficient and scalable composition and resource allocation. However, they did not consider the dynamic changes of traffic traversing through SFCs, which means that resource requirements of SFCs change over time. When resource requirements of VNF (virtual link) exceed the thresholds of physical node (link) where it deployed to, service performance of SFC will be degraded and even invalidated. So real-time migration of the VNF (virtual link) is needed to ensure SLA of SFC.

The migration of VNF (virtual link) will consume a certain amount of time and resources which cause different system cost [10]. Further, when physical node (link) has become hot spot of resources, it will be too late to migrate VNF (virtual link) since spinning-up new resources needs to take some time (in case the VNFs run in VMs), and even may fail due to insufficient resources [11]. Therefore, the real-time migration mechanism should be able to predict future resource demand based on historical resource requirements, and then report potential hot spots of resource for migration ahead of time. Unfortunately, current works on VNF migration ignore the problem. As mentioned in [12], a greedy algorithm which allowed SFC to reconfigure and manage VNF instances to reduce the number of VNF instances through migration was proposed, and in [13], it proposed a cost model to evaluate the migration cost and designed a greedy algorithm to optimize the migration of VNFs. But both of them did not consider system cost during VNF reconstruction and the time lag problem. In [14], it used Markov Decision Process theory to consolidate VNFIs in as few servers as possible so as to reduce the energy consumption, thereby optimizing energy consumption and reconstruction cost, but it also did not consider the time lag of migration.

In order to solve the time lag problem of migration, a feasible method is to adopt proactive prediction mechanism. For instance Rashid Mijumbi and Sidhant Hasija proposed a GNN-based prediction algorithm for VNFC resource requirements. In order to predict future resource requirements of VNFC, they exploited VNF forwarding graph topology information and historical resource utilization of each VNF component (VNFC), but did not consider how to use the prediction results to optimize resource allocation further [15]. Now, existing prediction approaches can be divided into three categories [16]: time-series approaches, such as ARIMA [17]; probabilistic graph approaches, such as Markov random fields (MRFs) [18] and nonparametric approaches, such as support vector regression (SVR) [19]. Numerous existing studies [15], [20]–[22] have shown that nonparametric methods usually perform better because they have ability to capture the uncertainty and complex nonlinearities of resource requirements. The most representative method is neural network technology [23] which can extract features from resource requirements and then take advantage of the relationship between resource features to make an effect prediction. Although the prediction accuracy of neural network is higher than that of traditional statistical models, the problems of long training period, slow convergence rate and easiness to fall into local minimum intrinsic in neural network can not be ignored. Fortunately, Fortunately, Deep Belief Network (DBN) [24], [25] is one of the classic algorithms for deep learning [26], [27], which is formed by stacking multiple Restricted Boltzmann Machines (RBMs). Since RBM can make full use of non-labeled data in pre-training as a generation model, DBN not only has the capability of processing big data and mining hidden information depended on the multiple layer structure which are inherent in deep learning, but also can extract nonlinear features of samples more effectively. Moreover, it solves the problems exist in other neural network models those require a large amount of labels and fall into local optimal solution quickly as the number of layers increase. Based on the advantages above, we attempt to adopt DBN to predict resource requirements of SFC so as to develop migration strategy.

We combine prediction mechanism and migration research together to optimize the migration of VNF, effectively avoiding performance degradation of SFC caused by resource bottlenecks while existing related works always study them separately. The main technical contributions of this paper can be summarized as follows:

1) To solve the problem of real-time VNF migration, we firstly build a system cost evaluation model integrating bandwidth utilization and migration time. And then we design a prediction mechanism based on DBN to predict future resource requirements of VNFs in advance to react dynamic changes of service. Based on the predicted results, we propose a heuristic algorithm to get the near optimal solution since the coordinated VNF migration problem is formulated as a integer linear programming (ILP).

2) Since resource requirements of VNFs change over time, there will be a large error in prediction of the initial training model. In the prediction mechanism of DBN, we propose an online learning method with the assistant of offline training. Moreover, multi-task learning which shares the information contained in related VNFs is introduced to improve prediction accuracy, and the samples are approximated by k-step of Contrast Divergence (CD-k) to improve the learning efficiency of RBM. In addition, adaptive learning rate is adopted to speed up convergence rate of RBM;

3) According to the prediction results, we propose a topology-aware migration algorithm in which VNFs are migrated to physical nodes meeting resource threshold constraints through greedy selection with the goal to optimize system cost, and then using the strategy obtained above as the initial solution, an optimization algorithm of migration based on tabu search is designed to further improve the efficiency.

The paper is organized as follows: Problem of VNF migration in NFV architectures is analyzed in Section II, then the network model and problem definition are established in Section III. The prediction mechanism and migration

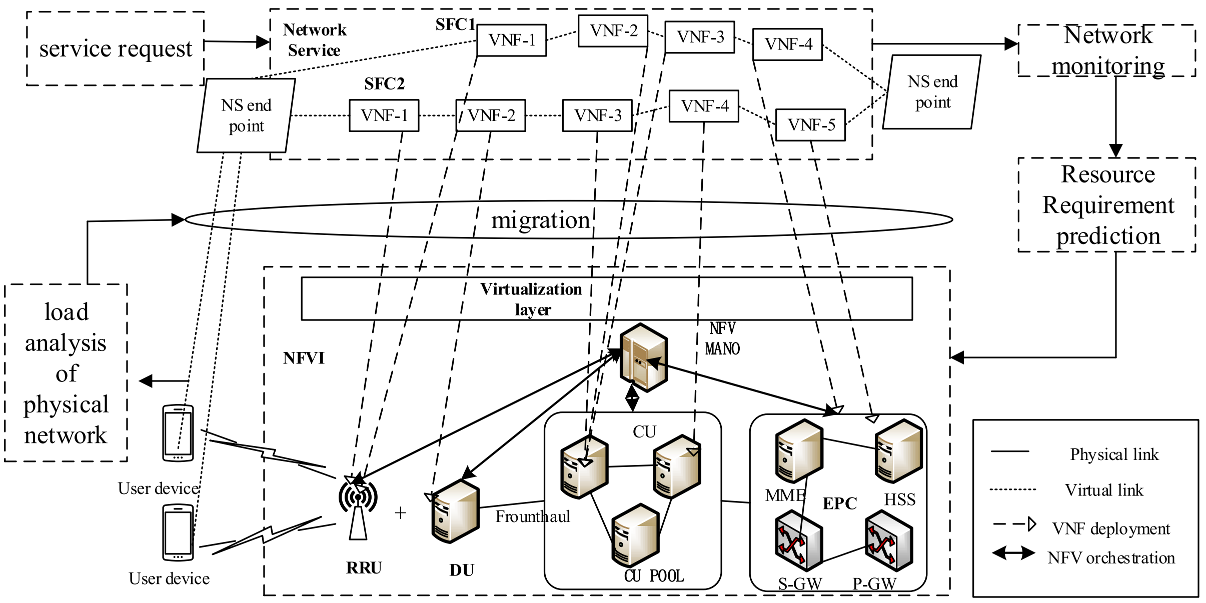

FIGURE 1. Network scenario in NFV architecture.

algorithm are proposed in Section IV and V respectively. Our proposal is evaluated in Section VI, and lastly in Section VII, we give a brief conclusion of our work.

II. PROBLEM OF THE MIGRATION IN NFV ARCHITECTURES

According to the vision of Next Generation Innovation Network (NovoNet 2020), NFs can be decoupled from dedicated devices and implemented in general NFV-based servers based on SDN/NFV concept. The resources of physical network consist of computing, memory and bandwidth resource. As shown in Fig.1, we consider the NFV architecture based on orchestration and control framework in which physical resources are provided to slice requests through virtualization. Physical network is composed of core network and access network which adopts a new architecture called C-RAN. A SFC is composed of some ordered VNFs and mapped to physical network according to their resource requirements. Through isolation, multiple VNFs can run on the same physical node without affecting each other, as shown in the Fig.1, VNF2 and VNF3 of SFC1 can be mapped to the same physical node. Different from [28] which assumed VNF only requests processing resources for the case study of CPU-intensive nodes, we consider CPU, memory and bandwidth resource at the same time. And according to the different performance, physical nodes are divided into three types: CPU-sensitive, memory-sensitive and bandwidth-sensitive. Each type of nodes has a different resource threshold for different kinds of resources.

Most of existing methods only focus on offline deployment of SFC, that is, according to the resource demand of SFC, a deployment strategy which assumes that resources required by SFC are constant during its lifetime is formulated to meet the SLA. However, actual flow through SFC is changing frequently, the total amount of resources allocated to it needs to be recalculated over time. Otherwise when resources required by the SFC are increased to exceed the resource thresholds of physical nodes and links to which it is mapped, the physical network may become invalid because of resource bottlenecks. Thereby it will affect the success rate of SFC deployment seriously. So it is necessary to migrate the VNF (virtual link) on the overloaded physical node (link) to other physical node (link) with lower load. In the Fig.1, VNF2 and VNF3 in SFC1 and VNF3 in SFC2 are deployed to the same node, we need to migrate one, two or all of the VNFs on it to other nodes when the resource requirements exceed its resource threshold at a certain time. The goal of the paper is to develop a spontaneous migration strategy, that is, through monitoring resource requirements of VNFs (virtual links) in real time, hot spots of resource in the physical network can be discovered in advance by prediction. Then dynamic migration of VNFs (virtual links) is performed according to the strategy formulated.

III. NETWORK MODEL AND PROBLEM DEFINITION

A. PHYSICAL NETWORK

We represent the physical network as an undirected graph $G^S = (N^S, L^S)$, where N^S and L^S denote the set of physical nodes and links respectively. One or more VNFs can be deployed in the same physical node. The physical nodes are characterized by CPU capacity C_m^S, each unit of CPU resources represents the resource required to process one packet, and memory capacity M_m^S, where $m \in N^S$. The physical links are characterized by bandwidth capacity B_{mn}^S between node m and n. L_{mn}^S denotes the set of paths with no loops between node m and n.

B. SFC REQUEST

Since the paper considers the inseparable flow, that is, a chain is linear and not allowed to be separated into two paths [29] so that the service performance of the SFC will not be affected. SFC is characterized by the ordered set $G^V = (N^V, L^V)$, SFCs are represented as $S = \{s_q|q = 1, 2, \cdots Q\}$, where N^V and L^V are the sets of all $VNF \in N_q^V \subseteq N^V$ and virtual links $l_{uv} \in L_q^V \subseteq L^V$. Each SFC is consisted of some ordered VNF. Each VNF $u \in N^V$ has a finite CPU resource demand denoted by C_u^V and memory resource demand denoted by M_u^V. Similarly, each virtual link l_{uv} has a finite bandwidth demand, denoted by B_{uv}^V. Binary variable $A_{mn}^{uv} = \{0, 1\}$ is defined to represent whether l_{uv} is mapped on $l_{mn} \in L^S$. The flow will be compressed or expanded after passing through one VNF so that the required bandwidth will change. So we assume that the bandwidth demand is denoted as follows:

$$B_{uv}^V(t) = C_u^V(t) \cdot L_p/t_{proc}. \tag{1}$$

where L_p denotes the packet length to be processed, and t_{proc} denotes the processing time of one packet.

C. PROBLEM FORMULATION

The main performance parameters considered in this paper consist of CPU demand, memory demand of node and bandwidth demand of virtual link. By monitoring the historical data of these three indicators comprehensively, it is determined whether physical node (link) is overloaded by the DBN prediction model of resource requirements based on online learning. Then performs migration according to the prediction result, thereby avoiding performance degradation of SFC due to resource bottlenecks. According to the difference of resource requirements, we use multi-threshold trigger mode to determine the physical nodes or links those need to execute migration, and the VNFs or virtual links those need to be migrated. The thresholds of resource utilization for CPU, memory, and bandwidth are denoted by $\gamma_C, \gamma_M, \gamma_B$ respectively. x_m^u is a binary indication, and $x_m^u = 1$ represents that VNF u is mapped to physical node m. If the resource requirements of SFCs at slot $t + 1$ predicted by using the historical data before slot $t+1$ exceed the thresholds of physical network, that is, when the following formula is satisfied, VNFs (virtual links) migration is automatically triggered to meet the SLA of service.

$$\begin{aligned}
&\sum_{u \in N^V} x_m^u(t+1) \cdot C_u^V \geq C_m^V \cdot \gamma_C \quad \forall m \in N^S \\
&\sum_{u \in N^V} x_m^u(t+1) \cdot M_u^V \geq C_m^V \cdot \gamma_M \quad \forall m \in N^S \\
&\sum_{l_{uv} \in L^V} (A_{mn}^{uv}(t+1) + A_{nm}^{uv}(t+1) \cdot B_{uv}^V(t+1) \geq B_{mn}^S \cdot \gamma_B \\
&\qquad\qquad\qquad\qquad \forall l_{mn} \in L^S
\end{aligned} \tag{2}$$

When the resource requirements of SFCs exceed the thresholds of physical node (link), it is necessary to develop a policy for migrating the VNF (virtual link), that is, which VNF (virtual link) is selected for migration and where to migrate. VNF migration requires to complete the migration of running context of virtual CPU and current state of memory. Because resource usage status of different VNFs are different, the appropriate VNF should be selected to migrate to meet the QoS of SFC and reduce the system overhead caused by VNF migration.

In summary, the paper develops a system overhead model for VNF migration, and designs a VNF migration algorithm for minimizing the system overhead caused by migration. The system overhead defined in this paper consists of two parts: migration overhead and bandwidth overhead. In [14], it pointed out that the transmission data brought by VNF migration is positively correlated with the energy consumption caused by VNF migration, and the amount of data migrated mainly comes from memory data. Similarly, in live migration, we need to copy memory and running states of VNF to the target physical node to ensure that the VNF continues to work. Therefore, VNF migration mainly includes the migration of the running state and memory resources. Since memory resources account for the majority of the migrated data, it is considered that the overhead of migrating VNF is equivalent to the time of occupying network bandwidth by migrating the memory data of VNF. The larger the amount of memory data to be migrated and the smaller the available bandwidth between physical nodes, the longer time the migration occupies the network bandwidth, and so the greater the impact on the normal operation of physical link. The migration overhead for migrating VNF u from the physical node n to m is defined as the following:

$$C_M^t(u, m) = \sum_{d \in P(m,n)} \frac{M^t(u) \cdot x_n^u(t)}{B^t(d)}. \tag{3}$$

wherein $B^t(d)$ denotes the remaining available bandwidth at time t of link in path $P(m, n)$ between physical node m and n. n indicates the physical node where the VNF was mapped to, and m indicates the target node to which the VNF will be migrated. $M^t(u)$ indicates the amount of memory resources for the VNF u at time t.

The bandwidth overhead defined here generated by VNF u and v for physical node m and n is defined as:

$$C_B^t(u, m, v, n) = B_{uv}^V \cdot x_m^u(t) \cdot x_n^v(t) \cdot hop^t(m, n). \tag{4}$$

wherein $hop^t(m, n)$ indicates the shortest distance between physical node m and n at time t.

And the bandwidth overhead of migrating u to m is defined as:

$$C_B^t(u, m) = \sum_{m \in N^S} \sum_{v \in N^V} \sum_{n \in N^S} C_B^t(u, m, v, n). \tag{5}$$

Then the overall system overhead of migrating u to m is defined as:

$$C_{tot}^t(u, m) = \alpha \cdot C_M^t(u, m) + \beta \cdot C_B^t(u, m). \tag{6}$$

wherein α and β are the corresponding coefficient. Therefore, the overall overhead of VNF migration during time t is:

$$C_{tot}^{t} = \sum_{u \in N^V} C_{tot}^{t}(u, m). \tag{7}$$

The optimization goal of the paper is to minimize the system overhead caused by migration, which is expressed as:

$$\begin{aligned}
Q1:\ & \min_{x,A} C_{tot}^{t} \\
s.t.\ & C1: \sum_{m \in N^S} x_m^u(t) = 1 \forall u \in N^V \\
& C2: \sum_{u \in N^V} x_m^u(t) \cdot C_u^V(t) \le C_m^V \cdot \gamma_C \quad \forall m \in N^S \\
& C3: \sum_{u} \in N^V x_m^u(t) \cdot M_u^V(t) \le C_m^V \cdot \gamma_M \quad \forall m \in N^S \\
& C4: \sum_{l_{uv} \in L^V} (A_{mn}^{uv}(t) + A_{nm}^{uv}(t) \cdot B_{uv}^V(t) \ge B_{mn}^S \cdot \gamma_B \\
& \qquad \forall l_{mn} \in L^S \\
& C5: \sum_{m \in N^S} A_{mn}^{uv}(t) - \sum_{m \in N^S} A_{nm}^{uv}(t) = x_n^u(t) - x_n^v(t) \\
& \qquad \forall n \in N^S,\ \forall l_{uv} \in L^V \\
& C6: A_{mn}^{uv}(t) = \{0, 1\} \quad \forall l_{uv} \in L^V,\ \forall l_{mn} \in L^S \\
& C7: x_m^u(t) = \{0, 1\} \quad \forall u \in N^V, \forall m \in N^S
\end{aligned} \tag{8}$$

Constraint C1 guarantees that each VNF should be only mapped to one physical node. Expression C2-C4 ensure that the resource requirements of the SFC cannot exceed the resource thresholds of the physical node and link, C5 expresses that each virtual link must exist a continuous path between physical node m and n to which the neighboring VNFs have been mapped. Finally constraint C7 represents the binary variable constraint of the node and link mapping.

IV. DBN PREDICTION MODEL OF RESOURCE REQUIREMENTS BASED ON ONLINE LEARNING

A. PREDICTION MODE OF DBN

Now most of the machine learning is single task learning [30]. The single-task learning mode is shown in Fig.2. It can be seen that it breaks a SFC into VNFs, and further decomposes resource requirements prediction of each VNF into two subtasks, which is composed of CPU and memory resource demand forecasting. The model space (training model) between tasks is independent, and ignores the rich correlation information between resource requirements in SFC, which affects the prediction accuracy of the model. Therefore, based on the interrelationship of resource requirements between VNFs in an SFC, the paper introduces the method of multi-task learning (MTL) [31]. With the aim of mutual benefit, MTL learns multiple related tasks together and has a better generalization effect by sharing the information contained in related tasks [32], [33]. The information from one task helps to learn related tasks more effectively. On the one hand, there are relevant and also irrelevant information among the multiple related tasks. The information

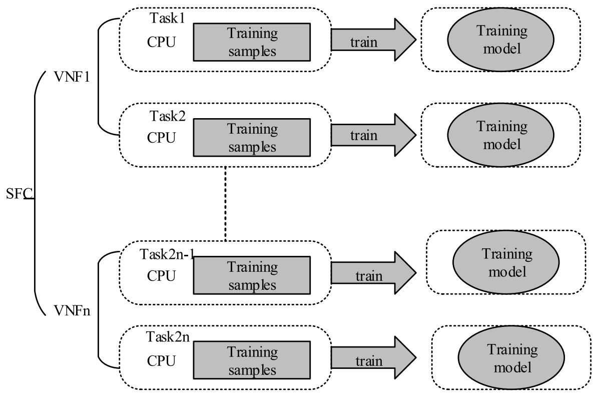

FIGURE 2. Single task learning mode of SFC.

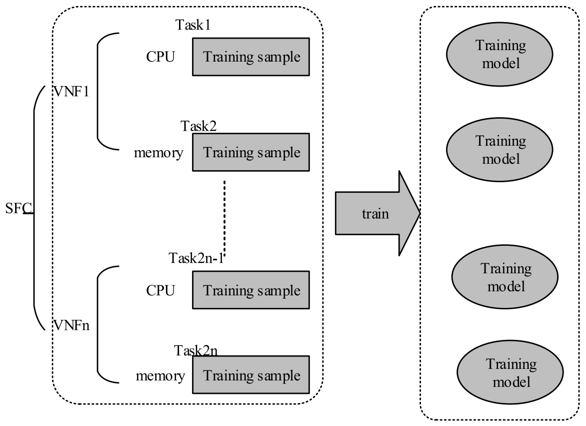

FIGURE 3. Multi-task learning mode of SFC.

irrelevant to a certain task can be regarded as introduced noise while learning the task, so the generalization performance can be improved. On the other hand, gradient back propagation is easy to fall into local minimum in the single task learning mode, while for MTL, the interaction (weight sharing) among these related tasks with different local minimums can help escape local minimums. Thus, improvements are gained from information sharing. As shown in Fig.3, MTL predicts resource requirements of VNF in the same SFC simultaneously, and jointly fine-tunes the features generated by deep layers in the shared model space, which can get better prediction results and improve generalization performance. So we adopt MTL mode to establish a DBN resource requirements prediction model in this paper.

B. DBN PREDICTION MODEL OF RESOURCE REQUIREMENTS

In order to optimize the target objective, the paper firstly uses DBN to complete the prediction of resource requirements. The prediction framework based on online learning consists of three parts: offline training, online learning and online migration which is shown in Fig.4.

In the offline training phase, historical resource requirements of CPU, memory and bandwidth of SFC are firstly

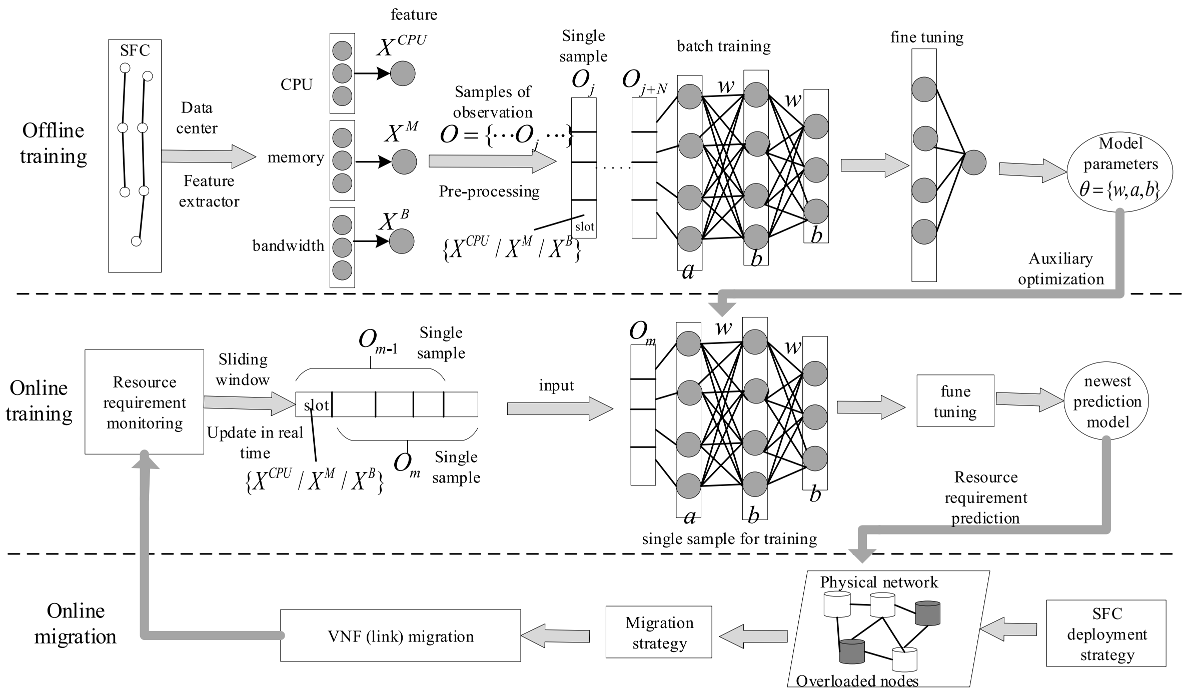

FIGURE 4. DBN prediction of resource requirements based on online learning.

collected. Since CPU resource and memory resource required by a VNF are related, our system extracts CPU and memory resource requirements as the features for CPU resource demand forecasting, represented as $X^{CPU} = \{C^V_{VNF}, M^V_{VNF}\}, \forall VNF \in SFC$. Similarly, CPU and memory resource requirements are also used as the features of memory resource demand prediction, represented as $X^M = \{C^V_{VNF}, M^V_{VNF}\}, \forall VNF \in SFC$. The prediction of the bandwidth can be obtained by the formula (1), so the features collection is not performed here. For each SFC, the historical observation sample set is represented as $O = \{\cdots O_j \cdots\}$, the jth sample is $O_j = [X_t, X_{t-1}, \cdots X_{t-d+1}]$, d represents the number of slots in the sample, and also refers to the length of the sliding window in online learning. $X_t = \{X^{CPU}_t, X^M_t\}$ represents the resource requirement characteristics of the SFC at time t, and because the characteristics of CPU and memory resource requirement prediction in the paper are same, choose any one of them expressed as $X_t = \{X^{CPU}_t or X^M_t\}$. Every sample selects d slots of resource characteristics in sequence according to the time series. After pre-processing the data, DBN model is constructed to perform batch training on the model parameters $\theta = (\boldsymbol{w}, \boldsymbol{a}, \boldsymbol{b})$ to improve training speed, wherein $\boldsymbol{w}, \boldsymbol{a}, \boldsymbol{b}$ represent connected weights between adjacent two layers, bias terms of visible layer, and bias term of hidden layer respectively. Then inverse fine tuning is performed to complete the initial prediction model.

Online learning enables real-time optimization of the prediction model. Since the monitored information of SFC resource requirements changes over time, there will be a large error in prediction of the initial training model based on the new sample after a period of time. Therefore, it is necessary to conduct online learning. Here, we use the model parameters obtained in offline training to assist online learning. Sliding window mechanism is used to update the sample in real time, which means that once a new slot of sample characteristics is added, the oldest slot of sample characteristics will be discarded so that the size of the sample is kept unchanged. Different from batch training mode in the offline phase, single sample training method is used in the training to improve computational efficiency of the DBN model. And then inverse fine tuning is performed to optimize model parameters.

Finally, in the online migration phase, overloaded nodes in physical network are judged according to the predicted results, and then migration strategy is formulated to perform VNF (virtual link) migration. After the migration is completed, in order to provide a reference for the next predication and migration, the sample is updated with the monitored information of resource requirements.

C. PARAMETERS TRAINING OF DBN

As shown in Fig.5, based on the MTL mode, the DBN resource requirements prediction model is formed by stacking multiple RBMs and a multi-task regression layer. RBM is an energy-based probability distribution model, and each RBM is a bipartite graph in which undirected edges connect its visible layer and hidden layer. The training sample $[X_t, X_{t-1}, \cdots X_{t-d+1}]$ of SFC is put into the visible layer neurons $\boldsymbol{v}$ in chronological order as the input. The first RBM is trained through the unsupervised stage, enabling hidden layer neurons $\boldsymbol{h}$ to extract features of resource requirement from sample data. The connection between hidden node and visible node is represented by a matrix of weights $\boldsymbol{w}$, which

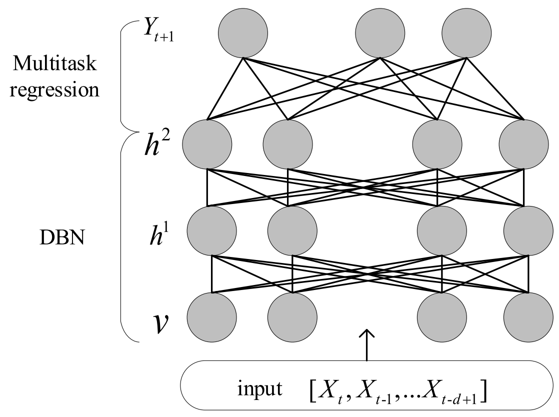

FIGURE 5. Resource requirements prediction based on DBN with two hidden layers.

indicating the influence on the extracted features of resource demand at different slots. The bias term for the visible layer and hidden layer are represented by ***a*** and ***b*** respectively. The training of RBM is easier because the node of same layer is independent and the hidden layer can obtain high-order characteristics of the visible layer. For the given input vector ***v*** of CPU, memory, and bandwidth resource requirements and feature extraction vector ***v***, the energy function of RBM can be expressed as:

$$E(\boldsymbol{v},\boldsymbol{h}) = -\boldsymbol{a}^T\boldsymbol{v} - \boldsymbol{b}^T\boldsymbol{h} - \boldsymbol{h}^T\boldsymbol{w}\boldsymbol{v}. \tag{9}$$

The joint probability of visible layer and hidden layer based on the energy function of RBM is

$$P(\boldsymbol{v},\boldsymbol{h}) = \frac{1}{Z}e^{-E(\boldsymbol{v},\boldsymbol{h})}. \tag{10}$$

where Z is the normalization constant which can be found by summing over all the possible pairs of visible and hidden vectors:

$$Z = \sum_{\boldsymbol{v}}\sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})}. \tag{11}$$

By summing over all the hidden layer vector, the edge distribution of the visible vector can be obtained:

$$P(\boldsymbol{v}) = \sum_{\boldsymbol{h}} P(\boldsymbol{v},\boldsymbol{h}) = \frac{1}{Z}e^{-E(\boldsymbol{v},\boldsymbol{h})}. \tag{12}$$

The maximization of (12) can be determined by taking its partial log derivative with respect to its patameters $\theta = (\boldsymbol{w}, \boldsymbol{a}, \boldsymbol{b})$:

$$\frac{\partial lnP(\boldsymbol{v})}{\partial\theta} = \sum_{\boldsymbol{v},\boldsymbol{h}} P(\boldsymbol{v},\boldsymbol{h})\frac{\partial lnP(\boldsymbol{v})}{\partial\theta} - \sum_{\boldsymbol{h}} P(\boldsymbol{h}/\boldsymbol{v})\frac{\partial lnP(\boldsymbol{v})}{\partial\theta}. \tag{13}$$

Here, "data" and "model" is used to keep a brief description of the probability distribution $P(\boldsymbol{h}/\boldsymbol{v})$ and $P(\boldsymbol{v},\boldsymbol{h})$, typically (13) can also be written as:

$$\frac{\partial lnP(\boldsymbol{v})}{\partial\theta} = < v_ih_j >_{data} - < v_ih_j >_{model}. \tag{14}$$

where $< v_ih_j >_{data}$ denotes the expectation of samples, $< v_ih_j >_{model}$ represents the expectation defined in the model. However, it is difficult to obtain an unbiased sample of CPU and memory resource requirements of VNFs in SFC, so the expectation $< v_ih_j >_{model}$ in the maximum log likelihood function cannot be easily computed and is thus estimated using contrastive divergence [34]. Specifically, the vector of the visible layer is firstly mapped to the hidden layer in the training, and then the visible layer is reconstructed by the hidden layer. Next, the reconstructed visible layer is mapped to the hidden layer again to acquire new hidden layer, and the repeated method is called Gibbs sampling. The correlation between the hidden layer and the visible layer is used as the main basis to update weights, which leads to the following parameter update equation:

$$\begin{aligned} w_{ij}^k &= w_{ij}^k + \xi(< v_i^kh_j >_{data} - < v_i^kh_j >_{model}) \\ &= w_{ij}^k + \xi(P(h_j = 1|\boldsymbol{v})v_i - v_i^kP(h_j = 1|\boldsymbol{v}^k)) \end{aligned} \tag{15}$$

$$\begin{aligned} a_i^k &= a_i^k + \xi(< v_i^k >_{data} - < v_i^k >_{model}) \\ &= a_i^k + \xi(P(v_i = 1|\boldsymbol{h})v_i - v_i^k) \end{aligned} \tag{16}$$

$$\begin{aligned} b_i^k &= b_i^k + \xi(< h_j^k >_{data} - < h_j^k >_{model}) \\ &= b_j^k + \xi(P(h_j = 1|\boldsymbol{v}) - P(h_j = 1|\boldsymbol{v}^k)) \end{aligned} \tag{17}$$

where ξ represents the learning rate, and ***k*** represents the ***k***th step of contrastive divergence. The neuron activation probabilities are given by the following equations:

$$\begin{aligned} P(h_j = 1|\boldsymbol{v}) &= \frac{P(h_j = 1|\boldsymbol{v})}{P(h_j = 1|\boldsymbol{v}) + P(h_j = 0|\boldsymbol{v})} \\ &= sigmoid(b_j + \sum_i w_{ij}v_i) \end{aligned} \tag{18}$$

Similarily,

$$P(h_j = 1|\boldsymbol{v}) = sigmoid(b_j + \sum_i w_{ij}v_i). \tag{19}$$

The CD-***k*** algorithm is used to iteratively obtain preliminary model parameters in the first RBM training. The learned features are used as the input of the next layer, then the second RBM is trained. Finally the features obtained after all RBM training in the DBN are used as the input of the multi-task regression layer. Moreover, since we employ sigmoid regression in the prediction layer, the whole structure can be seen as a complete structure of an NN. Once the optimal parameters (weight ***w*** and bias ***a***, ***b***) have been determined in the unsupervised stage, a supervised fine-tuned phase is performed via error backpropagation algorithm on the whole structure to adjust the weights and bias. This is done by setting labels, which represent the actual CPU and memory resource requirements of each VNF in SFC. BP algorithm only needs to perform a local search in the optimization. Due to the randomization of weights in the traditional BP neural network, it overcomes the disadvantages that optimization time is too long. Finally, the predicted output Y_{t+1} including $\widehat{M}_u^V(t+1)$ and $\widehat{C}_u^V(t+1)$ of each VNF in the SFC is obtained.

D. ADAPTIVE LEARNING RATE

For DBN, learning rate ξ in the pre-training stage has a great effect on the learning model. If ξ is too small, it will be difficult for RBM in the DBN model to converge in a short period. If it is too large, the RBM may not converge to extract effective features. In this paper, we judge whether the current ξ is appropriate through observing reconstruction error, so that the prediction model can automatically adjust learning rate according to the actual training state. If the reconstruction error decreases, we increase ξ by multiplying a number greater than 1, and vice versa, we decrease it by multiply a number lower than 1.

E. DBN PREDICTION OF RESOURCE REQUIREMENTS BASED ON ONLINE LEARNING

In summary, the DBN prediction of resource requirements based on online learning is as follows. Here, a SFC among all services is taken as an example for description.

Step 1: CPU and memory resource requirements of VNF in the SFC before slot t are taken as the observations. The sample set is normalized by the maximum and minimum method, which is represented as $\bar{x}_u = \frac{x_u - x_{min}}{x_{max} - x_{min}}$, where x_u denotes C_u^V or M_u^V, x_{max} and x_{min} refer to the maximum and minimum of resource requirement. Firstly, offline phase with batch training used to improve learning efficiency is carried out. A batch includes N samples in chronological order, and each sample consists of d slots. Then the initial prediction model is obtained through offline training;

Step 2: The resource requirements monitored at slot $t+1$ of the SFC are input as a label, and BP algorithm is used to obtain better model parameters $\boldsymbol{w}, \boldsymbol{a}, \boldsymbol{b}$ in the fine tuning;

Step 3: Perform online prediction and update the sample in real time. Let $t = t+1$, and make $[X_t, X_{t-1}, \cdots X_{t-d+1}]$ which represents the resource demand of d slots now as the training sample. Then use pre-training method of single sample to continuously update $\boldsymbol{w}, \boldsymbol{a}, \boldsymbol{b}$ by the formula (15)(16)(17) until the maximum number of iterations of RBM is reached;

Step 4: Predict Y_{t+1};

Step 5: Repeat steps 2-4 until all test samples have been predicted.

V. DYNAMIC MIGRATION OPTIMIZATION ALGORITHM BASED ON TABU SEARCH (TADM-DBN)

Calculate the resource utilization of physical nodes and links by using the resource requirements predicted of next slot in Section IV. When resource utilization exceeds the threshold of physical node, one or more VNFs deployed on the physical node need to be migrated to other physical node with lower resource utilization. Similarly, the virtual link needs to be migrated to other physical link with lower resource utilization. This paper firstly proposes a topology-aware algorithm for global dynamic migration to migrate VNF to physical node that satisfies the constraint of resource threshold through greedy selection, and then optimize the migration strategy by using the obtained solution as the initial solution based on tabu search algorithm [35].

A. TOPOLOGY-AWARE ALGORITHM FOR DYNAMIC GLOBAL MIGRATION (TPGDM-DBN)

The topology-aware algorithm for dynamic global migration proposed in the paper selects target physical node greedily for VNF mapped on overloaded node that is predicted. Firstly the local migration algorithm is used to calculate the overhead $C_{tot}^t(u, m)$, and the VNF u with minimal system overhead is selected and migrated to the target node m that satisfy the resource constraints. The procedures are detailed in ***Algorithm 1*** below.

Algorithm 1 Topology-Aware Algorithm for Dynamic Local Migration(TPLDM-DBN).

```
1:  Input: Overloaded physical node $S_s$ and the VNFs
    mapped on it (VNFList)
2:  Output: $VNF_m$that is needed to migrate and the target
    physical node $S_d$ to migrate
3:  $C_{min} \leftarrow \infty$
4:  for each $VNF_i \in VNFList$ do
5:      For each $S_j \in S$ do
6:          If $check\_constraints(VNF_i, S_j) = False$
7:              Continue;
8:          endif
9:          Add $S_j$ to $S_{VNF_i}$
10:     Endfor
11:     Calculate the minimum system overhead $C_i$ of migrat-
        ing $VNF_i$ to the selected set of physical nodes $S_{VNF_i}$
        that meet the resource constraints by topology-aware
        algorithm, and the target physical node is represented by
        $D_i$.
12:     If $C_i < C_{min}$ and $D_i \neq S_s$ then
13:         $C_{min} \leftarrow C_i$
            $VNF_m \leftarrow VNF_i$
            $S_d \leftarrow D_i$
14:     Endif
15: Endfor
16: Return $(VNF_m, S_d)$
```

The input is any one S_s of overloaded physical nodes those are predicted. Calculate the overhead $C_{tot}^t(VNF_i, S_j)$ of migrating VNF_i in the VNFlist to physical node S_j. Line 4 firstly determines if the selected physical node S_j satisfies its resource constraints, which means that S_j must be able to meet the future resource requirements of all VNFs deployed to it after migrated. If not, continue to judge the next physical node. Put all the satisfied physical node into S_{VNF_i}, the complexity is $O(|N^S|)$. Then use topology-aware algorithm to calculate the system overhead and save the minimum value in C_i. The topology-aware migration is defined as follows, the VNF should be migrated to the physical node closest to the mapped node of its neighbor VNF in SFC, where the distance refers to the hops of the reconstructed path of virtual link.

$$M(u,m) = \begin{cases} sum_{v\in neig(u)} D(m,n), \ where\ x_m^u = 1,\ x_n^v = 1 \\ m \in \{m | C_m^{\tilde{S}} > C_u^V \ \ and\ B^{\tilde{S}}(p(m,n)) > B_{uv}^V\} \end{cases}$$

where $D(m,n)$ denotes the hops between node m and n. $C_m^{\tilde{S}}$ denotes the available resource capacity of node m, $p(m,n)$ denotes the shortest path between node m and n. $B^{\tilde{S}}(p(m,n))$ denotes the available bandwidth capacity along path between m and n, the complexity is $O(|L^S| + |N^S| log|N^S|)$. Lines 10-12 calculate the minimum system overhead for migrating VNF in the VNF list. VNF_i that has the minimum system overhead will be the VNF_m that is needed to be migrated, and then migrate VNF_m to node S_d. The complexity of the Algorithm 1 is $O(|N_v|(|N^S|+|L^S|+|N^S|log|N^S|))$, where N_v is the number of VNFs on the overloaded node. Then a global migration algorithm is performed in ***Algorithm 2*** below.

Algorithm 2 Topology-Aware Algorithm of Dynamic Global Migration (TPGDM-DBN).

```
1: Input the set of all overloaded physical modes and the
   VNFList mapped on them:
2: for each $S_i \in SList$ do
3:     Execute Algorithm 1 to select the VNF to migrate
4:     while $S_i$ is still overloaded
           Continue to execute algorithm 1 to select the
         VNF with low system overhead for migration.
5:     Endwhile
6: Endfor
7: Output the VNF set to be migrated and the corresponding
   target physical nodes.
```

The input of algorithm 2 is the set of all overloaded physical nodes and the VNF list. For every overloaded physical node, algorithm 1 is adopted to select the VNF with the minimal overhead to migrate until all nodes do not exceed the resource threshold. Finally, output the solution. When executing 3-6 lines, the overloaded physical nodes are automatically removed. So the complexity of Algorithm 2 is $O(|N_s||N_v|(|N^S - N_s| + |L^S| + |N^S|log|N^S|))$, where N_s is the number of overloaded physical nodes.

B. DYNAMIC MIGRATION OPTIMIZATION ALGORITHM BASED ON TABU SEARCH (TADM-DBN)

Tabu search (TS) is a metaheuristic search method based on local search used for optimization. Use the tabu list to block the area that has just been searched to avoid roundabout search. Some special solutions can be released in the tabu list to ensure the diversity of search. Moreover, in order to implement global search, tabu search algorithm allows accepting inferior strategy to escape local optimal solution. Since the above algorithm is based on greedy heuristic method and the obtained solution is not optimal, hence we introduce tabu search algorithm to optimize the migration strategy by exchanging the order of the VNFs.

In order to design a TS algorithm, five major components must be determined: the initial solution, the neighborhood solutions, tabu list, aspiration criterion and stopping condition.

1) INITIAL SOLUTION

A good initial solution can greatly improve the efficiency of TS algorithm. Conversely, a poor initial solution will reduce the convergence speed of the algorithm. In the paper, solution obtained in the algorithm 2 is used as the initial solution of tabu search.

2) NEIGHBORHOOD SOLUTIONS

A set of new solutions formed according to a certain mobile strategy based on the current solution. The paper adopts exchange strategy. To be specific, for the n overloaded physical nodes, one feasible solution of the migration is Z. The neighborhood solutions $N(Z)$ are obtained by exchanging the order of VNF to migrate on any two physical nodes, which may change the type of VNF that need to be migrated according to the principle of minimum system overhead. By comparing the neighborhood solutions with the current solution, a better solution $\tilde{Z}$ is selected.

3) TABU LIST

If the VNFs on i and j are migrated, we declare i and j as a tabu to be searched during the next $n-1$ iterations, where n represents the number of overloaded physical nodes. The reason for using $n-1$ is to avoid the next $n-1$ iterations returning back to the same order. Therefore, the tabu in this paper is recorded in short-term memory as a 2-tupple $T(i,j)$.

4) ASPIRATION CRITERION

We allow for aspiration in which the criterion to allow a tabu move if it results in a solution with a lower system overhead than that of the current optimal solution Z^*. This would imply that the solution should be removed the tabu tag and added to the available candidate migration strategy.

5) STOPPING CONDITION

Finally, we have defined two criteria which determine when the algorithm stops: (1) the maximum number of iterations is reached. (2) there is no optimization in the system overhead after a certain consecutive iterations. The details are shown in ***Algorithm 3***.

As shown in algorithm 3, the initialization sets the obtained solution in algorithm 2 as the current solution Z meanwhile as the best solution Z^*. We also initialize the tabu list T as empty. The while loop starting at line 2 will continue searching for a solution until the stopping condition is met. In lines 4 to 6, the neighborhood solutions are checked to release those are in tabu list. If the candidate solution has a lower system overhead than the current best solution (line 8-10), then we set it as the new best solution and add it to the tabu list. Finally the counters in the tabu list are updated in line 11. Since the complexity of choosing neighborhood solutions is $O(|N_s|^2)$,

Algorithm 3 Migration Optimization Algorithm Based on Tabu Search (TADM-DBN)

```
1:  Determine Initial Solution:$Z = Z_0, Z^* = Z_0, T = \emptyset$,
    Apiration Value: $A(Z^*) = C_{tot}^t(Z^*)$
2:  While StopConditionNotMet() do
3:      determine Candidate Set W in the neighborhood
    solutions $N(Z)$, and choose the optimal solution $X^*$ in
    W;
4:    If $C_{tot}^t(X) < A(Z^*), X \in T$ and $C_{tot}^t(X) < C_{tot}^t(X^*)$
        $X^* = X$
        update $C_{tot}^t(X^*)$
5:    Endif // Aspiration criterion
6:    If $C_{tot}^t(X^*) < C_{tot}^t(Z^*)$
7:      $Z^* = X^*$,
8:      $C_{tot}^t(Z^*) = C_{tot}^t(X^*)$,
9:      $A(Z^*) = C_{tot}^t(Z^*)$
10:   Endif
11: update Tabu List $T$, $T = T \cup X^*$,
12: $Z = X^*$
13: Endwhile
14: return $Z^*$.
```

the complexity of any remaining step is $O(1)$. Assume that the number of iterations is I when the stopping condition is satisfied, the corresponding complexity of the whole algorithm is $O(I|N_s|^3|N_v|(|N^S - N_s| + |L^S| + |N^S|log|N^S|))$. The complexity of the algorithm is under control, because if N_s is relatively small compared with N^S, the value outside the bracket is smaller, then the complexity is relatively low and vice versa. If N_s is relatively large compared with N^S, the number of target physical nodes that VNFs can be migrated to is limited (the network may be severely overloaded). Now there are few choices for migration, that is, the value in brackets is small, and the complexity of the algorithm is not high. So the proposed migration algorithm is applicable for small and medium-sized networks.

After the VNF of the overloaded node and its adjacent link are migrated, the virtual link selected with minimal system overhead mapped to the overloaded physical link is migrated. The minimum here is the minimal bandwidth overhead due to the migration overhead is zero.

VI. EVALUATIONS

To evaluate the efficiency of our algorithms, we compare the accuracy of prediction with the BP-NN prediction [20] and compare the performance of the migration algorithm with the heuristic algorithm proposed in [13].

A. SIMULATION ENVIRONMENT

Simulations are carried out on a Laptop with Intel Core i7-4790, 8GB of memories and CPU frequency of 3.60GHz using Matlab 2015b. The physical network graphs and SFC are generated by the GT-ITM tool [36]. CPU capacity of physical node is uniformly distributed between 50 and 100 units, while memory capacity is between 1 to 3 GB. And physical link bandwidth is uniformly distributed between 60Mbps and 110Mbps. Given 15 SFCs, the number of VNFs in each SFC can be 2,3,4. In order to verify the performance of DBN prediction model and the migration algorithm proposed in this paper, we use the superposition of sinusoidal and cosine signals to simulate the sample data of SFC resource requirements and discretize it. In order to simulate the real environment, each sample is added with a randomly generated value following a Poisson distribution. And in this way, it is reasonable and conform to the actual data flow characteristics of periodicity and burstiness. The relationship between different types of VNFs and resources in a SFC is achieved by slightly changing the amplitude, angular frequency or initial phase angle of the sinusoidal or cosine signal, and then periodically collect CPU, memory and virtual link resource requirement of each SFC. We set the sampling period to one hour and collect 10,000 samples totally, in which the first 70% are as training samples for DBN to train, while the remaining 30% are as test samples to perform prediction. Intercept 120 slots of data from the prediction results to describe the effect.

B. RESOURCE REQUIREMENTS PREDICTION

A SFC with 3 VNFs is used as an example to describe the efficiency of prediction when using the first 5 or 10 time slots to predict resource requirements of the next time slot, that is, d is 5 or 10. For the sake of simplicity, DBN prediction of resource requirements based on online learning adopts a classic 5-layer model. The number of nodes per layer is 30-30-20-10-6 and 60-128-32-16-6, respectively. The maximum iteration of RBM is 300, and the initial learning rate is set to be 0.0001. The root mean square error (RMSE) used to measure the accuracy of the prediction is defined as follows.

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(Y_t - \hat{Y}_t)^2}. \tag{20}$$

where Y_t represents the tag, $\hat{Y}_t$ represents the predicted value, n is the number of test samples. The smaller the RMSE, the higher the accuracy of prediction. It is verified by simulation that the RMSE is 0.1893 in the case of 30-30-20-10-6 and 0.1562 in the case of 60-128-32-16-6. Therefore, we use a DBN prediction model with 60-128-32-16-6 nodes per layer. The prediction of CPU resource requirements for one of the VNFs in the SFC is selected to illustrate the accuracy of the prediction model. Fig.6 and Fig.7 show the comparison of the DBN predicted value with the true value of CPU resource requirement and the BP-NN predicted value with the true value. The RMSE in DBN case is 0.1552 which is better than 0.2793 obtained by BP-NN algorithm. This indicates that DBN prediction model overcomes the shortcomings of traditional neural networks. It can fully utilize a large number of non-labeled data in the unsupervised stage, so as to obtain closely optimal model parameters, and extract features from input more effectively which improves the prediction

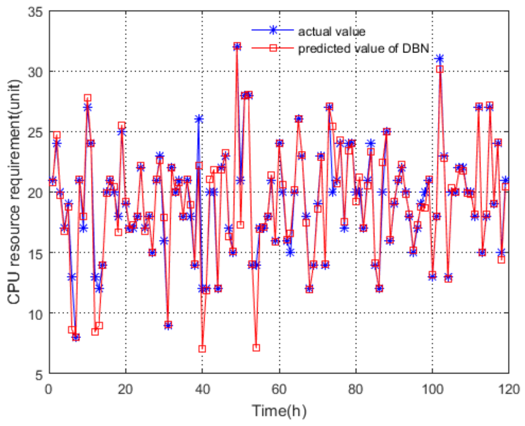

FIGURE 6. CPU resource requirements based on DBN prediction.

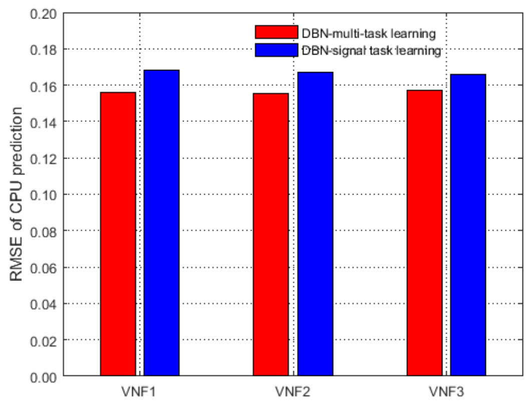

FIGURE 8. CPU resource requirements based on DBN prediction.

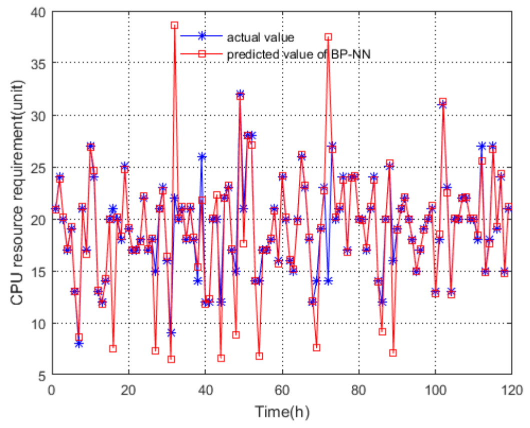

FIGURE 7. CPU resource requirements based on BP-NN prediction.

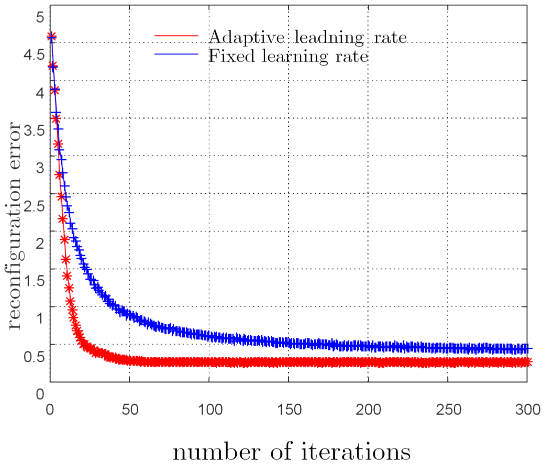

FIGURE 9. Reconstruction error of RBM for learning rate.

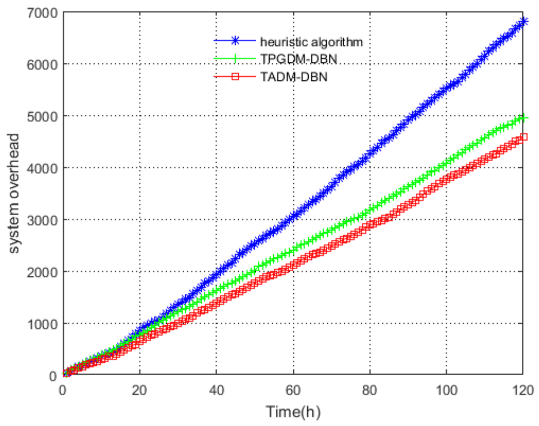

FIGURE 10. System overhead of different algorithms.

accuracy of resource requirements. While BP-NN requires a large amount of labeled data to train the model parameters, and as the number of layers increases, the model will quickly fall into local optimal solution, which will affect prediction accuracy. In addition, from the comparison of CPU resource requirements prediction under the two learning modes in Fig.8, it can be seen that MTL has higher prediction accuracy because it makes full use of rich information in related tasks to improve overall performance of the prediction.

In order to compare the effects of fixed learning rate and adaptive learning rate on convergence speed of RBM, the number of layers in RBM is set to one, and then compare the error curves of RBM reconstruction in two cases. For adaptive learning rate, multiply the learning rate by 1.2 when reconstruction error is reduced, and multiply the learning rate by 0.8 when increased. It can be seen from Fig.9 that reconstruction error of RBM with adaptive learning rate tends to be relatively stable after 50 iterations, and after 200 iterations with fixed learning rate. We can conclude that adaptive learning rate accelerates convergence speed of RBM compared to fixed learning rate, and improves computational efficiency of DBN to some extent.

C. MIGRATION ALGORITHM

Fig.10 shows a comparison of system overhead for different algorithms. In order to more objectively compare the performance of different algorithms, we describe the accumulation of system overhead over time, since migration strategy of VNF or virtual link in each time slot will affect system

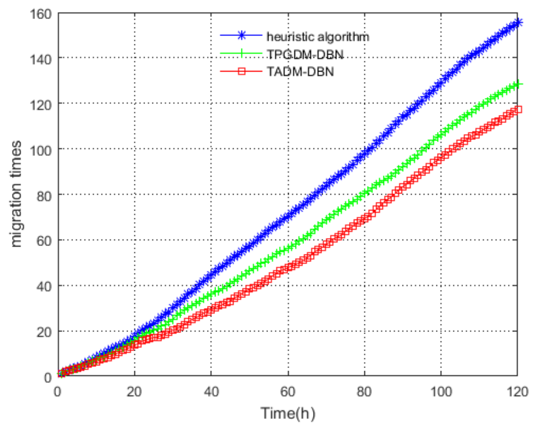

FIGURE 11. Migration times of different algorithms.

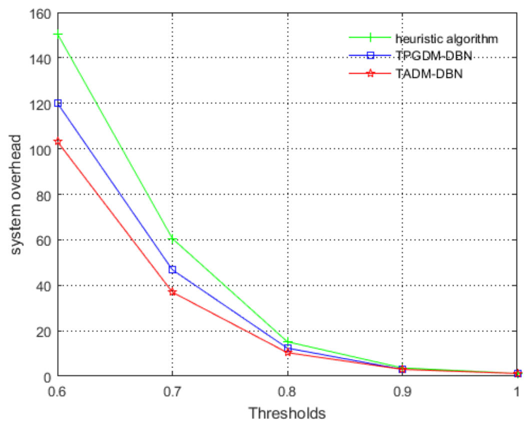

FIGURE 13. System overheads with different thresholds.

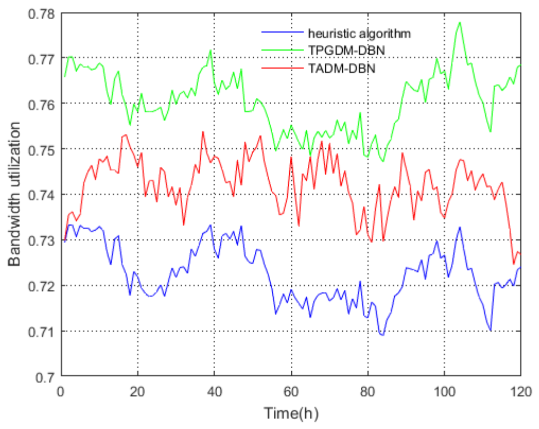

FIGURE 12. Bandwidth utilization of different algorithms.

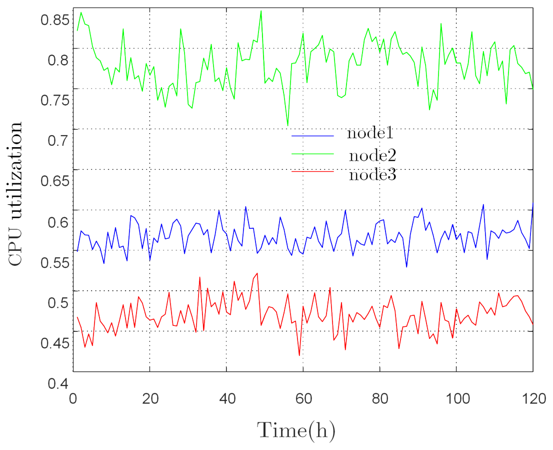

FIGURE 14. CPU utilization of different nodes.

overhead of next time slot. Due to migration overhead of this paper can represent the time required for migration and bandwidth overhead reflects bandwidth resource utilization or transmission delay, system overhead composed of them shows performance of service. It means that the smaller the system overhead, the better the service performance. As can be seen from the figure, system overhead is increasing over time, and the growth of two algorithms proposed in this paper is relatively small. It is because TPGDM-DBN algorithm considers migration overhead and bandwidth overhead together, so that system overhead is optimized. TADM-DBN algorithm uses tabu search to further optimize system overhead based on the solution obtained from TPGDM-DBN algorithm. However, the heuristic algorithm proposed in [13] only focuses on optimization of bandwidth overhead, hence the performance is relatively poor compared with proposed algorithms. This is also reflected in the comparisons of migration times and bandwidth utilization of different algorithms in Fig.11 and Fig.12. Since the algorithm in [13] only pursues optimization of bandwidth overhead unilaterally, it is slightly better than the proposed algorithm in terms of bandwidth utilization, but its system overhead has increased significantly because it is easier to cause frequent migrations of VNF or virtual link.

Fig.13 shows system overhead for different thresholds. In order to analyze the impact of thresholds more vividly, the difference between CPU, memory and bandwidth resource requirements and the difference of service performance between physical nodes are temporarily masked, that is, the simulation is performed by setting same threshold. As can be seen from the figure, as the threshold increases, system overhead decreases gradually. This is because the larger the threshold, the less likely the resource demand exceeds the threshold, therefore the less times the migration needs to be performed.

Fig.14 shows an example of CPU utilization for different nodes. CPU resource thresholds of three nodes selected for analysis are 0.65, 0.85 and 0.55 respectively. It can be seen that CPU resource requirements of VNF deployed in any node do not exceed their resource thresholds, thereby ensuring service performance of physical nodes and satisfying SLA of the SFC. Similarly, memory and bandwidth resource utilization of physical network are also guaranteed. For the sake of space, no further analysis is performed here. Fig.15 shows

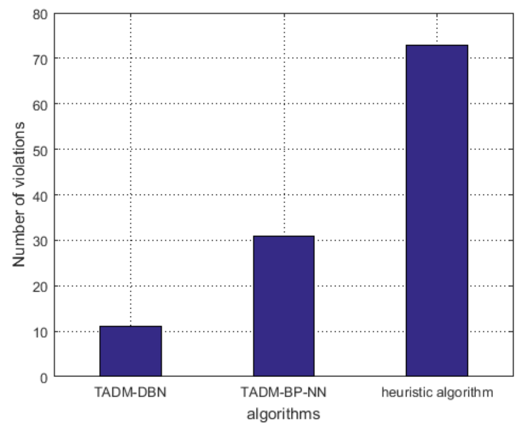

FIGURE 15. Number of violations for different algorithms.

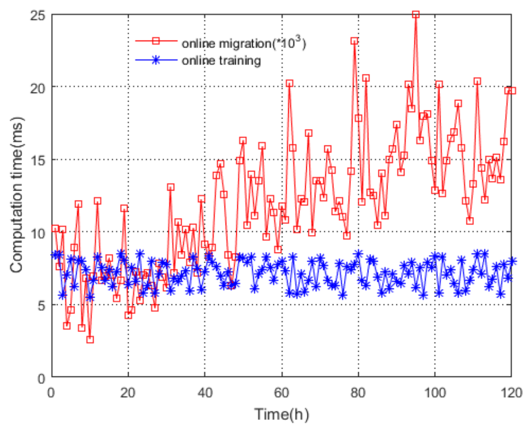

FIGURE 16. Computation time.

the number of SLA violations for different algorithms. The number of violations refers to the times of occurrences that resource requirements exceed thresholds of physical network. It can be seen that TADM-DBN proposed significantly reduces the number of violations during migration because it can report resource requirements in advance. Then make migration decision based on the predicted requirements to ensure performance of SFC and improve users' experience. The prediction accuracy based on BP-NN is relatively low, so the number of violations is higher. Algorithms that adopt prediction mechanism generate less violations than those which do not execute prediction first, which reflects the importance of prediction to improve users' experience.

As in [37], we further study the computation time of online training and online migration in Fig.16. After offline training period, the model parameters of DBN in the model are saved to assist online training. Therefore, our evaluations showed that each online prediction required about 5-10ms. In addition, we observed that an online migration can be obtained in about 20s. The prediction time and migration times are comparatively low, given that they are performed for resource requirements 1h ahead of time.

VII. CONCLUSION

In order to solve the poor service performance problem caused by dynamic changes of resource requirements, the paper first establishes a system overhead model taking account of migration overhead and bandwidth overhead together. Then it proposes a resource requirements prediction algorithm using DBN based on online learning to predict the future resource requirements. Adaptive learning rate and multi-task learning mode are introduced in the prediction to improve the efficiency of prediction. Finally, by using the result of prediction, a topology-aware dynamic migration algorithm is designed, and the migration strategy used as initial solution is further optimized by tabu search. The simulation results show that prediction algorithm in the paper has high accuracy and the convergence speed of training has also been improved. In addition, the combination of prediction and migration algorithms effectively reduces system overhead and SLA violations which result in better performance of SFC. Due to different divisions of the protocols may have different performances in the proposed 5G C-RAN architecture, effective deployment scheme of VNFs in access network needs to be proposed. Therefore, the migration of VNFs in access network will be more accurately modeled in future work.

REFERENCES

[1] N. H. Mahmood, M. Lauridsen, G. Berardinelli, D. Catania, and P. Mogensen, "Radio resource management techniques for eMBB and mMTC services in 5G dense small cell scenarios," in *Proc. IEEE 84th Veh. Technol. Conf. (VTC-Fall)*, Montreal, QC, Canada, Sep. 2016, pp. 1–5.

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

[3] F. Z. Yousaf, M. Bredel, S. Schaller, and F. Schneider, "NFV and SDN—Key technology enablers for 5G networks," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2468–2478, Nov. 2017.

[4] S. Vassilaras, L. Gkatzikis, N. Liakopoulos, I. N. Stiakogiannakis, M. Qi, L. Shi, L. Liu, M. Debbah, and G. S. Paschos, "The algorithmic aspects of network slicing," *IEEE Commun. Mag.*, vol. 55, no. 8, pp. 112–119, Aug. 2017.

[5] Y. Xie, Z. Liu, S. Wang, and Y. Wang, "Service function chaining resource allocation: A survey," Jul. 2016, *arXiv:1608.00095*. [Online]. Available: https://arxiv.org/abs/1608.00095

[6] M. M. Rahman, C. Despins, and S. Affes, "Design optimization of wireless access virtualization based on cost & QoS trade-off utility maximization," *IEEE Trans. Wireless Commun.*, vol. 15, no. 9, pp. 6146–6162, Sep. 2016.

[7] L. Qu, C. Assi, K. Shaban, and M. J. Khabbaz, "A reliability-aware network service chain provisioning with delay guarantees in NFV-enabled enterprise datacenter networks," *IEEE Trans. Netw. Service Manag.*, vol. 14, no. 3, pp. 554–568, Sep. 2017.

[8] R. Riggio, A. Bradai, D. Harutyunyan, T. Rasheed, and T. Ahmed, "Scheduling wireless virtual network functions," *IEEE Trans. Netw. Service Manag.*, vol. 13, no. 2, pp. 240–252, Jun. 2016.

[9] 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. Manage. (IM)*, Ottawa, ON, Canada, May 2015, pp. 98–106.

[10] J. Zhang, L. Li, and D. Wang, "Optimizing VNF live migration via para-virtualization driver and QuickAssist technology," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Paris, France, May 2017, pp. 1–6.

[11] J. Xia, D. Pang, Z. Cai, M. Xu, and G. Hu, "Reasonably migrating virtual machine in NFV-featured networks," in *Proc. IEEE Int. Conf. Comput. Inf. Technol. (CIT)*, Nadi, Fiji, Dec. 2016, pp. 361–366.

[12] T. Wen, H. Yu, G. Sun, and L. Liu, ''Network function consolidation in service function chaining orchestration,'' in *Proc. IEEE Int. Conf. Commun. (ICC)*, Kuala Lumpur, Malaysia, May 2016, pp. 1–6.

[13] J. Xia, Z. Cai, and M. Xu, ''Optimized virtual network functions migration for NFV,'' in *Proc. IEEE 22nd Int. Conf. Parallel Distrib. Syst. (ICPADS)*, Wuhan, China, Dec. 2016, pp. 340–346.

[14] V. Eramo, E. Miucci, M. Ammar, and F. G. Lavacca, ''An approach for service function chain routing and virtual function network instance migration in network function virtualization architectures,'' *IEEE/ACM Trans. Netw.*, vol. 25, no. 4, pp. 2008–2025, Aug. 2017.

[15] R. Mijumbi, S. Hasija, S. Davy, A. Davy, B. Jennings, and R. Boutaba, ''Topology-aware prediction of virtual network function resource requirements,'' *IEEE Trans. Netw. Service Manag.*, vol. 14, no. 1, pp. 106–120, Mar. 2017.

[16] W. Huang, G. Song, H. Hong, and K. Xie, ''Deep architecture for traffic flow prediction: Deep belief networks with multitask learning,'' *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 5, pp. 2191–2201, Oct. 2014.

[17] J. Yang, L. Li, Y. Shi, and X. Xie, ''An ARIMA model with adaptive orders for predicting blood glucose concentrations and hypoglycemia,'' *IEEE J. Biomed. Health Inform.*, vol. 23, no. 3, pp. 1251–1260, May 2019.

[18] D. Huang, Z. Deng, L. Zhao, and B. Mi, ''A short-term traffic flow forecasting method based on Markov Chain and Grey Verhulst model,'' in *Proc. 6th Data Driven Control Learn. Syst. (DDCLS)*, Chongqing, China, May 2017, pp. 606–610.

[19] S. Alam, M. Kang, J.-Y. Pyun, and G. Kwon, ''Performance of classification based on PCA, linear SVM, and multi-kernel SVM,'' in *Proc. 8th Int. Conf. Ubiquitous Future Netw. (ICUFN)*, Vienna, Austria, Jul. 2016, pp. 987–989.

[20] J. Pati, B. Kumar, D. Manjhi, and K. K. Shukla, ''A comparison among ARIMA, BP-NN, and MOGA-NN for software clone evolution prediction,'' *IEEE Access*, vol. 5, pp. 11841–11851, 2017.

[21] K. Lu, W. Zhang, and B. Sun, ''Multidimensional data-driven life prediction method for white LEDs based on BP-NN and improved-Adaboost algorithm,'' *IEEE Access*, vol. 5, pp. 21660–21668, 2017.

[22] S.-H. Lin, M. Paolieri, C.-F. Chou, and L. Golubchik, ''A model-based approach to streamlining distributed training for asynchronous SGD,'' in *Proc. IEEE 26th Int. Symp. Modeling, Anal., Simulation Comput. Telecommun. Syst. (MASCOTS)*, Milwaukee, WI, USA, Sep. 2018, pp. 306–318.

[23] W. Wang, Y. Gao, and Z. Jin, ''Interpretive reservoir: A preliminary study on the association between artificial neural network and biological neural network,'' in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Rio de Janeiro, Brazil, Jul. 2018, pp. 1–8.

[24] C. Zhang, Y. He, L. Yuan, and S. Xiang, ''Analog circuit incipient fault diagnosis method using DBN based features extraction,'' *IEEE Access*, vol. 6, pp. 23053–23064, 2018.

[25] Y. Mao, J. Shen, and X. Gui, ''A study on deep belief net for branch prediction,'' *IEEE Access*, vol. 6, pp. 10779–10786, 2018.

[26] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, Q. V. Le, M. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, and A. Y. Ng, ''Large scale distributed deep networks,'' in *Proc. Adv. Neural Inf. Process.*, 2012, pp. 1223–1231.

[27] L. Deng, D. Yu, and J. Platt, ''Scalable stacking and learning for building deep architectures,'' in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Mar. 2012, pp. 2133–2136.

[28] J. G. Herrera and J. F. Botero, ''Resource allocation in NFV: A comprehensive survey,'' *IEEE Trans. Netw. Service Manage.*, vol. 13, no. 3, pp. 518–532, Sep. 2016.

[29] W. Ma, C. Medina, and D. Pan, ''Traffic-aware placement of NFV middleboxes,'' in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, San Diego, CA, USA, Dec. 2015, pp. 1–6.

[30] J. Deutsch and D. He, ''Using deep learning-based approach to predict remaining useful life of rotating components,'' *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 48, no. 1, pp. 11–20, Jan. 2018.

[31] R. Caruana, *Multitask Learning*. New York, NY, USA: Springer-Verlag, 1998.

[32] R. Collobert and J. Weston, ''A unified architecture for natural language processing: Deep neural networks with multitask learning,'' in *Proc. 25th Int. Conf. Mach. Learn.*, Jul. 2008, pp. 160–167.

[33] Y. Li, X. Tian, T. Liu, and D. Tao, ''On better exploring and exploiting task relationships in multitask learning: Joint model and feature learning,'' *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 5, pp. 1975–1985, May 2018.

[34] G. E. Hinton, ''Training products of experts by minimizing contrastive divergence,'' *Neural Comput.*, vol. 14, no. 8, pp. 1771–1800, 2002.

[35] M. H. Mashinchi, M. R. Mashinchi, and M. Mashinchi, ''Tabu search solution for fuzzy linear programming,'' in *Proc. 7th IEEE/ACIS Int. Conf. Comput. Inf. Sci.*, Portland, OR, USA, May 2008, pp. 82–87.

[36] E. W. Zegura, K. L. Calvert, and S. Bhattacharjee, ''How to model an internetwork,'' in *Proc. Conf. Comput. Commun.*, San Francisco, CA, USA, Mar. 1996, pp. 594–602.

[37] L. Huang, X. Feng, A. Feng, Y. Huang, and L. P. Qian, ''Distributed deep learning-based offloading for mobile edge computing networks,'' in *Mobile Networks and Applications*. New York, NY, USA: Springer, Nov. 2018, pp. 1–8.

LUN TANG received the Ph.D. degree in communication and information system from Chongqing University, Chongqing, China.

He is currently a Professor with the Key Laboratory of Mobile Communication Technology, School of Communication and Information Engineering, Chongqing University of Posts and Telecommunications.

XIAOYU HE received the B.S. degree in electronic information engineering from the Southwest University of Science and Technology, China, in 2017. She is currently pursuing the M.S. degree with the Key Laboratory of Mobile Communication Technology, Chongqing University of Posts and Telecommunications (CQUPT), China. Her research interests include network function virtualization (NFV), resource allocation in 5G C-RAN, and the applications of reinforcement learning technique in mobile networks.

PEIPEI ZHAO received the B.S. degree from the Chongqing University of Posts and Telecommunications, Chongqing, China, in 2016, and the M.S. degree in communication and information systems from the Key Laboratory of Mobile Communication Technology, School of Communication and Information Engineering, Chongqing University of Posts and Telecommunications, in 2019.

GUOFAN ZHAO received the B.S. degree from the Chongqing University of Posts and Telecommunications, Chongqing, China, in 2016, and the M.S. degree in communication and information systems from the Key Laboratory of Mobile Communication Technology, School of Communication and Information Engineering, Chongqing University of Posts and Telecommunications, in 2019.

YU ZHOU received the B.S. degree from the Chongqing University of Posts and Telecommunications, Chongqing, China, in 2016, and the M.S. degree in communication and information systems from the Key Laboratory of Mobile Communication Technology, School of Communication and Information Engineering, Chongqing University of Posts and Telecommunications, in 2019.

QIANBIN CHEN (M'03–SM'14) received the Ph.D. degree in communication and information system from the University of Electronic Science and Technology of China, Chengdu, China, in 2002.

He is currently a Professor with the School of Communication and Information Engineering, Chongqing University of Posts and Telecommunications, and the Director of the Chongqing Key Laboratory of Mobile Communication Technology. He has authored or coauthored more than 100 papers in journals and peer-reviewed conference proceedings, and has coauthored seven books. He holds 47 granted national patents.