Accurate Prediction of Required Virtual Resources via Deep Reinforcement Learning

Haojun Huang, Zhaoxi Li, Jialin Tian, Geyong Min, Wang Miao, and Dapeng Oliver Wu

***Abstract*—Resource provisioning for the ever-increasing applications to host the necessary network functions necessitates the efficient and accurate prediction of required resources. However, the current efforts fail to leverage the inherent features hidden in network traffic, such as temporal stability, service correlation and periodicity, to predict the required resources in an intelligent manner, incurring coarse-grain prediction accuracies. To tackle this problem, in this paper, we propose an Accurate Prediction of Required virtual Resources (APRR) approach via Deep Reinforcement Learning (DRL). We first confirm the resource requests have more similar features and identify the high-dimensional required resources in computing, storage and bandwidth can be effectively consolidated into a single standardized value. Built upon these observations, we then model the required resources as a time-variant network matrix, which includes a number of elements, obtained from the network measurements, and some missing elements needed to be inferred. To obtain accurately predicted results, DRL-based matrix factorization with a set of available rules has been introduced into APRR and alternately executed in agent to minimize the prediction errors. Moreover, the error-prioritized learning experience samples, built on rewards, are designed for model training with quicker convergence. Simulation experiments on real-world datasets illustrate that APRR can accurately predict the required virtual resources compared with the related approaches.**

***Index Terms*—Virtual Network Functions, Virtual Resources, Network Matrix, Deep Reinforcement Learning.**

I. Introduction

THE emerging prevailing networking paradigms like Network Function Virtualization (NFV) [1], cloud computing and edge computing [2] enable on-demand provisioning of virtual computing, storage and bandwidth resources over networks [3]. Benefiting from these paradigms, many diversified network services, such as real-time multimedia sharing, autonomous driving, and online treatment, have been created by dynamically allocating highly-variable resources. Generally, the required resource variations of such services will incur the over-provisioning or under-provisioning of resources to host the necessary Virtual Network Functions (VNFs) like Firewall (FW), Intrusion Detection System (IDS), Wireless LAN Controller (WLC) and Wide-area Network Optimizer (WNO), running in software, and thus results in additional expenditures or poor Service Level Agreements (SLAs) in 4G/5G and campus networks [4], [5], [6]. Therefore, resource provisioning for these services necessitates the accurate prediction of required resources in a cost-efficient manner.

Due to the inevitable contradiction between the constantly changing network requirements and limited network resources, how to predict the required resources of such VNFs has become an urgent challenge for network services to achieve the desired Quality of Service (QoS) and Quality of Experience (QoE) [7], [8]. A number of efforts explicitly concerning with the efficient prediction of required resources and their operation performance have been conducted during the last years to enhance service performance [9], [10], [11], [12], [13], [14]. The main idea of them is to infer the unknown required resources with the historical data observed from applications and users in advance, enabling underlying resources to cope with the necessary Network Functions (NFs), mainly referring to VNFs. All of them mostly focus on the prediction of QoS [14], execution time, energy usage [11], resource availability [12], or workloads [15], [8], [16], [17], [18], [19], always built on distinct regularity, time-variant correlations and obvious tendency of the pre-existing workloads or consumed resources.

Although significant progress has been made, there still exist two main problems for effective and efficient prediction of required virtual resources to be addressed. First, how to design a model to describe the time-variant required VNF resources. The traffic workloads running on NFV-based infrastructure are ever-increasing and always change in different time scales with high fluctuations. However, in reality, a number of traffic workloads illustrated in Table I always consume the necessary resources in the approximately constant scale in computing, storage and bandwidth in each time scale, for example, 1:2:0.039 and 1:−:0.008 in Barracuda [20] and Cisco [21], respectively, to host the same VNFs. This is mainly because all VFNs are run in software on general-purpose equipment, rather than the traditional dedicated hardware, and thus consume approximately the same resources to provide the similar services. Besides, the correlations of network states between time-variant traffics and resource consumption, such as temporal stability, service correlation and traffic periodicity, which are necessary for accurate prediction of the required resources, cannot be well captured to build prediction models. It is requisite to explore the features behind the high variance of traffic workloads to accurately model highly-variable VNF resources. Second, how to accurately infer

This work was supported in part by the National Key Research and Development Program under Grant 2020YFC2005003, and the National Natural Science Foundation of China under Grant 92067206 and 61972222 (Corresponding author:Haojun Huang and Geyong Min).

H. Huang and Z. Li are with the School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan, 430074, China. Email: {hjhuang, zhaoxll}@hust.edu.cn.

J. Tian, G. Min and W. Miao are with the Department of Computer Science, University of Exeter, Exeter, EX44QF, UK. Email: jialin.tian@hotmail.com; {g.min, wang.miao}@exeter.ac.uk.

D. Wu is with the Department of Computer Science, City University of Hong Kong, Hong Kong. E-mail: dapengwu@cityu.edu.hk.

TABLE I
RELATIONS AMONG VARIOUS CONSUMED RESOURCES OF VNFs

Providers	VNFs	Comp (Core)	Stor (G)	Band (Mbps)	Ratio (Comp:Stor:Band)
Ref. [21]	FW_1	5	-	0.025	1:-:0.005
	FW_2	25	-	0.200	1:-:0.008
Ref. [22]	WLC_1	1	2	-	1:2:-
	WLC_2	2	8	-	1:4:-
Ref. [20]	WNO_1	16	32	0.622	1:2:0.039
	WNO_2	24	48	1.000	1:2:0.042
Ref. [23]	IDS_1	1	-	0.050	1:-:0.050
	IDS_2	1	-	0.080	1:-:0.080

the required resources with available measurement results. Given a designed predicted model, due to the ever-increasing network state and action space in network applications, it is NP-hard to infer the missing elements with traditional solutions, which require consumed resources with distinct regularity or obvious tendency. A number of typical machine learning (ML) based solutions have been developed with the exploration and exploitation of hidden knowledge included in consumed resources, but most of them failed to work together to infer the missing elements intelligently with more prediction rewards. For example, the widely-used matrix-factorization-based schemes often just exploit Stochastic Gradient Descent (SGD) [4], Momentum [6] or AdaDelta [5] to recover the unknowing required resources in a settled update manner, but are not well jointly adopted in model iterations with the learned historical experiences of action exploration and the network-related factors behind model computation. This will incur more prediction errors, due to the lack of comprehensive considerations of their superiorities.

In this paper, we propose APRR, an Accurate Prediction of Required virtual Resources solution via Deep Reinforcement Learning (DRL), to tackle the above-mentioned issues. We first quantitatively analyse current network traffics and confirm that all of them have similar features in temporal stability, service correlation and traffic periodicity. Then, we propose an efficient resource grading model to consolidate high-dimensional required resources of network traffic into a single standardized value for their prediction, while hardly without information loss. Based on these confirmations, we model the required network resources as a time-variant network matrix, which includes a number of elements, obtained from the network measurement, and the missing elements needed to be inferred. To obtain accurately predicted results, DRL-based matrix factorization with the superiorities of exiting mechanisms has been introduced into APRR and alternately executed to minimize the prediction errors. Differing from the existing efforts, the main contributions of this paper can be summarized as follows:

- Built upon the extensive analysis of real service traffic, we identify that such service data exhibits the features of temporal stability, service correlation, and traffic periodicity. Notice that the real-world service traffics always are referred to as highly-variable resources in computing, storage and bandwidth comprised in the approximately constant scale, which is challenging to be analysed in high-dimensional space, we propose a new efficient resource grading model to consolidate the high-dimensional required resources of services into one-dimensional standardized value for efficient resource prediction. Based on these observations, the network resource requirement has been modelled as a time-variant network matrix factorization problem to be addressed.
- A DRL-based matrix factorization approach is proposed for the accurate prediction of the required virtual resources, where the missing elements are inferred through alterative multi-collaboration matrix factorization step by step. The available matrix factorization rules with distinct superiorities are introduced into APRR and alternately used in agents to minimize the prediction errors. Moreover, the error-prioritized learning experience samples are used for model training with quicker convergence, built on rewards obtained from each iteration.
- Extensive simulation experiments are implemented for randomly generated network services to evaluate the performance of the proposed APRR. Simulation results over real-world datasets show that APRR achieve accurate prediction of the required network resources compared with the related approaches.

The rest of this paper is organized as follows. Firstly, we describe the related work in Section II, and present our empirical data analyses in Section III. Then, we describe preliminaries of our solution, referring to network model, resource grading model and problem formulation, in Section IV. After that, the details of APRR are presented in Section V, followed by its performance evaluation in Section VI. Finally, we conclude the work in Section VII.

II. RELATED WORK

Resource prediction is of great importance for various working and upcoming network services. Generally, the previous investigations are devoted mostly to the prediction of required resource information in QoS [14], energy usage [11], resource availability [12], [24], and traffic workloads [15], [8], [16], [18], [19], [25], etc. For example, to avoid over-/under-provisioning of cloud resources, an accurate prediction framework of real-world workload was proposed [25], by exploiting the weight-based ensemble model built on multiple predictors. However, these solutions nearly cannot well capture correlations of predicted models and the features of time-variant traffics from historically consumed resources, and thus achieve coarse-grained prediction results.

In order to reduce prediction errors, a number of ML-based approaches have been developed during past few years. In [26], a distributed online system, which seamlessly integrates various cooperating components, was developed for grid resource monitoring and prediction with QoS guarantee. To address the gradient disappearance issue caused by traditional RNN, Long Short-Term Memory (LSTM) is used [17], [27], [28] as an improvement of RNN to realise long-term prediction of workload. In [17], efficient LSTM-based prediction of highly variable workloads, by introducing compressed historical workload amount and GRU block into RNN, is proposed for cloud services. The efforts in [27] designed accurate LSTM-based price prediction to identify the stable cloud transient servers at reduced job budget with the performance guarantee. Both LSTM and k-nearest neighbor in [29] were jointly exploited to classify popularity trends, so that the cache hit rate of fog radio access networks close to optimal policy is achieved. In [30], the pooling-based deep RNNs, which can batch the load profiles of a set of customers, were introduced into a pool of inputs to learn the uncertainty during the household load forecasting. Besides, Gradient Descent and levenberg marquardt have been introduced in [28] to perform dynamic Bi-LSTM-based prediction of resource usage.

More recently, notice that tensor factorization using limited measured data can infer the unknown results but with coarse-grained accuracy, two efficient ML-based derivatives have been developed. In [13], a lightweight Deep Learning-based model with compressed built-in parameters, built on canonical polyadic factorization, is designed to predict the cloud workload in industry. In [14], Adaptive Matrix Factorization (AMF) via SGD-based online learning is proposed to perform online QoS prediction for candidate services. Differing from such solutions, APRR first applies the DRL-based matrix factorization, by introducing multiple rules with distinct superiorities, specified by the existing matrix factorization, to conduct accurate prediction of virtual resources for hosting the necessary VNFs. Besides, the high-dimensional required resources of services have been consolidated into one-dimensional standardized value for efficient resource prediction, by introducing resource grading model into APRR with less prediction errors.

III. Empirical Data Analyses

It has been proved that the feature similarities among changing traffic data are necessary for accurate prediction of the required resources. Therefore, in this section, we quantitatively investigate such features, concerned with network resource requests from their temporal stability, service correlation and traffic periodicity [31], with the empirical available datasets [32], [33]. The datasets are collected from 4G cells and campus networks, respectively, where the traffic data includes the essential information like data amount, the timestamp and IP addresses of origin-destination, and is stored at regular hour intervals. To facilitate the understanding, the key parameters and notations used in this paper are listed in TABLE II.

TABLE II
The Important Parameters and Notations

Notations	Descriptions
$D/\hat{D}$	Resource requirement/prediction matrix
X, Y	Factor matrices of $\hat{D}$ with $x_{ip}{\in}X$ and $y_{jp}{\in}Y$
f_i/t_j	VNF instance/time slot to record resources
$d(f_i, t_j, l)$	Required resource of f_i in computing, storage and bandwidth at t_j with $l = 1, 2, 3$
$d(f_i, t_j)/\hat{d}(f_i, t_j)$	Graded/prediction resource of f_i at t_j
$w(f_i, t_j)$	Binary variable to represent whether $d(f_i, t_j)$ is contained in D
$< \mathcal{S}, \mathcal{A}, \mathcal{R}, \gamma >$	Four-tuple MDP model containing state, action, reward space and discount factor
$b/b_{x_i}/b_{y_j}$	Overall offset of $\hat{D}/X/Y$
$Q(S_{\tilde{t}}, A_{\tilde{t}})$	Action-value function denoting the cumulative discount return over several iterations of DQN
$\pi^\star$	The optimal policy obtained by agent
θ/θ^-	Parameter of Q-network/target Q-network

A. Temporal Stability

Temporal stability means how the traffic data dynamically changes over adjacent time slots. Let t and i denote the time slot and the type of services, respectively, with which the traffic data can be uniquely identified as $s(i, t)$. Given service i, the numerical difference between two adjacent time slots, i.e. t and $t - \tau$ ($0 < \tau < t$), can be quantitatively denoted as

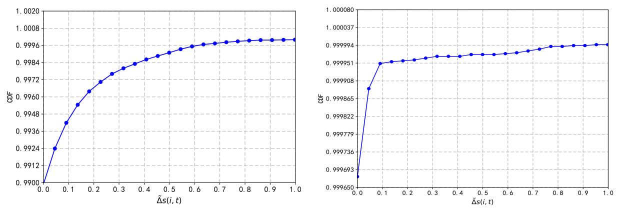
$$\Delta s(i, t) = |s(i, t) - s(i, t - \tau)|, \tag{1}$$

where $\Delta s(i, t)$ denotes the change of traffic $s(i, t)$ at t with the time interval of τ. Compared to the overall fluctuation of the traffic traffic of service i, if $\Delta s(i, t)$ is small, the change of traffic around timestamp t can be considered stable. Thus, the normalized difference between two adjacent time slots of service i can be presented as

$$\tilde{\Delta} s(i, t) = \frac{|s(i, t) - s(i, t - \tau)|}{\max\limits_{t} |s(i, t) - s(i, t - \tau)|}, \tag{2}$$

where $\max\limits_{t} |s(i, t) - s(i, t - \tau)|$ is the maximal difference, representing the largest instantaneous fluctuation in datasets.

The normalized difference $\tilde{\Delta} s(i, t)$ of the traffic data in datasets [32] and [33] can be further plotted in the form of Cumulative Distribution Function (CDF) illustrated in Figs. 1(a) and 1(b), respectively, where τ is set as 1. It can be seen from Fig. 1 that the probabilities that $\tilde{\Delta} s(i, t)$ is less than 0.1 are around 99.4% and 99.993%, respectively, which greatly indicates that the traffic data, concerned with network resource requests, is characterized by temporal stability.



(a) CDF of $\tilde{\Delta} s(i, t)$ in data [32] (b) CDF of $\tilde{\Delta} s(i, t)$ in data [33]

Fig. 1. Temporal stability of traffic data in different datasets.

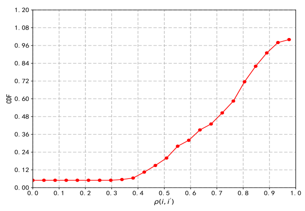
B. Service Correlation

Service correlation refers to the correlation and similarity degree of different types of services, e.g., services i and i', in the whole time dimension, which is quantitatively measured by means of the Normalized Cross Correlation (NCC) coefficient as

$$\rho(i, i') = \frac{\sum\limits_{t=1}^{T} \left[\frac{|s(i,t) - \bar{s}(i)|}{\max\limits_{2 \le t \le T} |s(i,t) - \bar{s}(i)|} \times \frac{|s(i',t) - \bar{s}(i')|}{\max\limits_{2 \le t \le T} |s(i',t) - \bar{s}(i')|} \right]}{\sqrt{\sum\limits_{t=1}^{T} \frac{[s(i,t) - \bar{s}(i)]^2}{\max\limits_{2 \le t \le T} [s(i,t) - \bar{s}(i)]^2}} \sqrt{\sum\limits_{t=1}^{T} \frac{[s(i',t) - \bar{s}(i')]^2}{\max\limits_{2 \le t \le T} [s(i',t) - \bar{s}(i')]^2}}}. \tag{3}$$

Where both $\bar{s}(i) = \frac{1}{T} \sum_{t=1}^{T} s(i, t)$ and $\bar{s}(i') = \frac{1}{T} \sum_{t=1}^{T} s(i', t)$ are the mean values of $s(i, t)$ and $s(i', t)$ over all ranges of times, respectively. According to the Cauchy-Schwarz inequality, $\rho(i,i^{'})$ ranges from 0 to 1, where $s(i,t)$ is related to $s(i',t)$ if $\rho(i,i^{'})=1$ while uncorrelated if $\rho(i,i^{'})=0$.

The CDFs of NCC coefficients of traffic data between any two services in datasets [32] and [33] are shown in Figs. 2(a) and 2(b). It can be seen that only around 10% of data owns an NCC value of less than 0.4 for both datasets, indicating that the traffic data, concerned with network resource requests, performs well in the service correlation.


(a) CDF of $\rho(i,i^{'})$ in data [32]

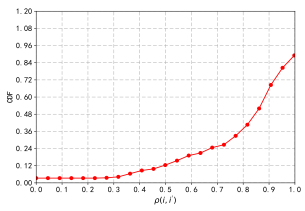

(b) CDF of $\rho(i,i^{'})$ in data [33]

Fig. 2. Service correlation of traffic data in different datasets.

C. Traffic Periodicity

Traffic periodicity is referred to as the traffic similarity, e.g., appearing peaks or valleys at approximately the same time period, due to the same network behaviors of users. Therefore, the gap between data, collected at the same time of two consecutive days for service i, is used for the analysis of traffic periodicity. Notice that the traffic data of [32] and [33] is measured every an hour and a minute, respectively, we add up the data in one hour in [33] as the total traffic data for this hour, and thus there are 24 time intervals in one day for both two datasets. Therefore, the data collected at the same time of two consecutive days can be denoted as $s(i,t)$ and $s(i,t+24)$, with the gap as

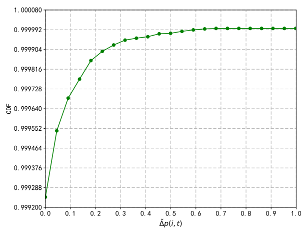
$$p(i,t)=|s(i,t)-s(i,t+24)|. \tag{4}$$

Furthermore, the normalized value of $p(i,t)$ is equal to

$$\tilde{\Delta}p(i,t)=\frac{|s(i,t)-s(i,t+24)|}{\max\limits_{1\leq t\leq T-24}|s(i,t)-s(i,t+24)|}, \tag{5}$$

where $\max\limits_{1\leq t\leq T-24}|s(i,t)-s(i,t+24)|$ is the maximal gap of the traffic data of service i, recorded at the same time between any two adjacent days. The smaller $\tilde{\Delta}p(i,t)$ is, the more likely that the traffic data will own the periodicity.

The CDFs of $\tilde{\Delta}p(i,t)$ derived from data [32] and [33] have been given in Figs. 3(a) and 3(b), respectively. It can be seen that almost all values of $\tilde{\Delta}p(i,t)$ (more than 99.9%) are around 0 for both two datasets, indicating the existing traffic periodicity.


(a) CDF of $\tilde{\Delta}p(i,t)$ in data [32]

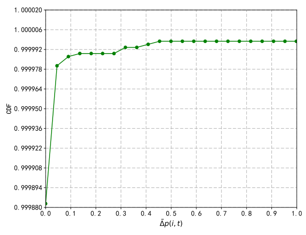

(b) CDF of $\tilde{\Delta}p(i,t)$ in data [33]

Fig. 3. Traffic periodicity of traffic data in different datasets.

IV. Problem Statement

This section first presents the network model of virtual network resources, followed by the resource grading model, then formulates the problem of required virtual resource prediction.

A. Network Model

It is considered that there are multiple NFV services originated from different users. Each VNF will consume different amount of resources in computing, storage and bandwidth, depending on their roles in the whole operations [9], [1]. Suppose that network operators can monitor the resource requirements of existing VNFs at regular intervals. Following the traffic features obtained from our observations, the required virtual resources of various VNFs can be counted at discrete time intervals, and denoted as a matrix. The required resources of VNF f_i at time slot t_j are recorded as a vector $(d(f_i,t_j,1),d(f_i,t_j,2),d(f_i,t_j,3))$, and can be simplified as $d(f_i,t_j,l)$, where l ($1\leq l\leq 3$) represents the type of resources, i.e., computing, storage and bandwidth. Thus, the required resources of m kinds of VNFs f_1, f_2,...,f_m at n time slots t_1, t_2,...,t_n can be modeled as an $m\times n$ matrix, expressed as

$$\begin{bmatrix} d(f_m,t_1,l) & .. & d(f_m,t_j,l) & .. & d(f_m,t_n,l) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ d(f_i,t_1,l) & \cdots & d(f_i,t_j,l) & .. & d(f_i,t_n,l) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ d(f_1,t_1,l) & .. & d(f_1,t_j,l) & .. & d(f_1,t_n,l) \end{bmatrix}, \tag{6}$$

where $1\leq i\leq m$ and $1\leq j\leq n$. In this way, the required resources of f_i at time slots t_1, t_2,...,t_n are recorded in the i-th row in $m\times n$ matrix, and the j-th column in $m\times n$ matrix contains the required resources of various VNFs at time slot t_j.

B. Resource Grading Model

Notice that each VNF in a number of network applications consumes the necessary resources in computing, storage and bandwidth in the different approximately constant scale to steer the workloads, the required resource $d(f_i,t_j,l)$, an array of three data, of VNF f_i at time t_j is converted to the graded level $d(f_i,t_j)$, described as follows.

Firstly, the demand of each kind of resource of f_i is normalized, denoted as

$$\overline{d(f_i,t_j,l)}=\begin{cases}\dfrac{d(f_i,t_j,l)}{\max\limits_{1\leq j\leq n}d(f_i,t_j,l)}, & d(f_i,t_j,l)\neq 0,\\ 0, & otherwise.\end{cases} \tag{7}$$

where $\max\limits_{1\leq j\leq n}d(f_i,t_j,l)$ is the maximum value of the demand for the l-th resource of f_i at all time slots. Furthermore, the weight of the l-th resource of f_i, denoted by $w(f_i,l)$, can be measured as

$$w(f_i,l)=\frac{\sum_{j=1}^{n}d(f_i,t_j,l)/s_l}{\max\limits_{1\leq i\leq m}[\sum_{j=1}^{n}d(f_i,t_j,l)/s_l]}, \tag{8}$$

where $\sum_{j=1}^{n}d(f_i,t_j,l)$ is the sum of the l-th resource demand of f_i at all time slots, s_l is the number of none zero values of the l-th resource, and the maximum value of which in all kinds of VNFs is $\max_{1\leq i\leq m}[\sum_{j=1}^{n} d(f_i,t_j,l)]$. On this basis, the resource demand of f_i at t_j, $d(f_i,t_j,l)$, is graded as

$$d(f_i,t_j) = \sum_{l=1}^{3} w(f_i,l) * \overline{d(f_i,t_j,l)}. \tag{9}$$

Built on Eq. (9), the resource matrix D can be simplified as

$$D = \begin{bmatrix} d(f_m,t_1) & \vdots & d(f_m,t_j) & \vdots & d(f_m,t_n) \\ \cdots & \ddots & \cdots & \ddots & \cdots \\ d(f_i,t_1) & \vdots & d(f_i,t_j) & \vdots & d(f_i,t_n) \\ \cdots & \ddots & \cdots & \ddots & \cdots \\ d(f_1,t_1) & \vdots & d(f_1,t_j) & \vdots & d(f_1,t_n) \end{bmatrix}, \tag{10}$$

which is further used for the virtual resource prediction.

Each predicted result of three resources derived from the predicted graded value $\hat{d}(f_i,t_n)$ changes in a range, which is determined by the upper bound $\hat{d}_{max}(f_i,t_n,l)$ and the lower bound $\hat{d}_{min}(f_i,t_n,l)$ as

$$\begin{cases} \hat{d}_{max}(f_i,t_n,l) = \dfrac{\hat{d}(f_i,t_n) * \alpha^{+}(f_i,l)}{\sum_{l=1}^{3} w(f_i,l) * \alpha(f_i,l)}, \\ \hat{d}_{min}(f_i,t_n,l) = \dfrac{\hat{d}(f_i,t_n) * \alpha^{-}(f_i,l)}{\sum_{l=1}^{3} w(f_i,l) * \alpha(f_i,l)}. \end{cases} \tag{11}$$

Here, $\alpha(f_i,l)$, $\alpha^{+}(f_i,l)$ and $\alpha^{-}(f_i,l)$ are the average, maximal and minimal proportion of the l-th resource in all resources of f_i, respectively, which are calculated based on existing resource request data as

$$\begin{cases} \alpha(f_i,l) = \dfrac{\sum_{j=1}^{n} d(f_i,t_j,l)}{\sum_{l=1}^{3}\sum_{j=1}^{n} d(f_i,t_j,l)}, \\ \alpha^{+}(f_i,l) = \max_{1\leq j\leq n} \dfrac{d(f_i,t_j,l)}{\sum_{j=1}^{n} d(f_i,t_j,l)}, \\ \alpha^{-}(f_i,l) = \min_{1\leq j\leq n} \dfrac{d(f_i,t_j,l)}{\sum_{j=1}^{n} d(f_i,t_j,l)}. \end{cases} \tag{12}$$

Built upon this, the predicted range for the l-th resource of f_i at time t_j, corresponding to the predicted resource graded value $\hat{d}(f_i,t_n)$, is $[d_{min}(f_i,t_n,l), d_{max}(f_i,t_n,l)]$, which helps network operators provide better services.

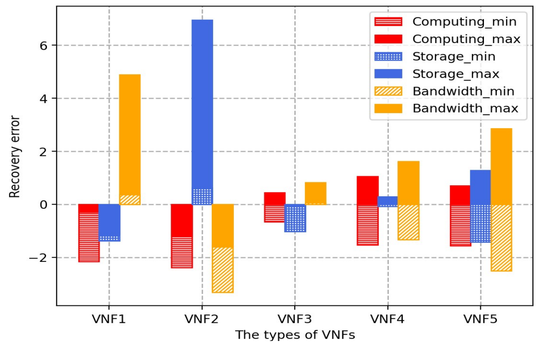


Fig. 4. Recovery errors between the original consumed resources and its recovered resources with the graded level $d(f_i,t_j)$ for hosting k (k=5) VNFs.

In order to estimate the effectiveness of the resource grading and recovery in a quantitative manner, Fig. 4 illustrates the recovery error between the originally required resources $d(f_i,t_j,l)$, concerned with the computing, storage and bandwidth resources, and the range of three resources recovered by the graded level $d(f_i,t_j)$, i.e., $[\hat{d}_{min}(f_i,t_j,l), \hat{d}_{max}(f_i,t_j,l)]$. There are diverse $d(f_i,t_j,l)$ ($1\leq i\leq 5$) of 5 VNFs graded and then recovered, where the difference between $d(f_i,t_j,l)$ and $\hat{d}_{min}(f_i,t_j,l)$ as well as that between $d(f_i,t_j,l)$ and $\hat{d}_{max}(f_i,t_j,l)$ are calculated for l=1, 2, 3 and then shown in Fig. 4, respectively. It can be seen that the two differences are respectively above and below the zero axis for various resources of VNF3, VNF4 and VNF5, indicating that $d(f_i,t_j,l)$ of them is involved in the recovery range $[\hat{d}_{min}(f_i,t_j,l), \hat{d}_{max}(f_i,t_j,l)]$, which further proves that the resource grading and recovery process are effective.

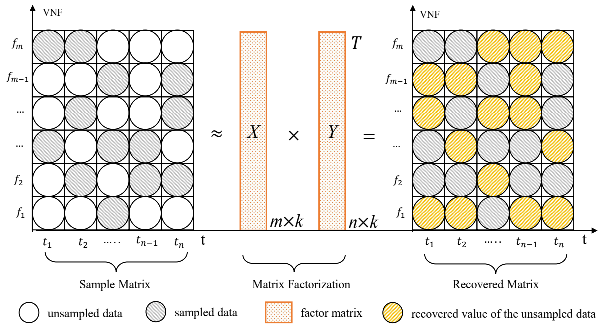


Fig. 5. Matrix-factorization-based resource prediction.

C. Problem Formulation

Given the above notations, the prediction of virtual network resources is conducted with the help of an $m\times n$ graded resource matrix D, where the data of the former $(n-1)$ columns is the existing graded observations as Eq. (10) and the data in n-th column, to-be-predicted resources of various VNFs at time slot t_n, is set to 0. The accurate prediction of virtual network resources can be formulated as the DRL-based matrix completion problem, which estimates the missing data of matrix D, i.e., the data in column n, with a number of known observations.

Our previous investigations in **Section III** illustrate that the resource-related data is much more characterized by temporal stability, service correlation and traffic periodicity. This means that any two columns $[d(f_1,t_i),...,d(f_k,t_i),..., d(f_n,t_i)]\in D$ and $[d(f_1,t_j),...,d(f_k,t_j),..., d(f_n,t_j)]\in D$ have distinct inherent similarities, and thus, the graded resource matrix D is low-rank or approximately low-rank. This can ensure that the $m\times n$ matrix D can be factorized via DRL into the product of two matrices X and Y^T as

$$D \simeq XY^T = \hat{D}, \tag{13}$$

where X and Y are $m\times k$ and $n\times k$ matrices, respectively, and $k \ll \min(m,n)$ is the rank of matrix D. Denote $x_i = [x_{i1},...,x_{ik}]$ and $y_j = [y_{j1},...,y_{jk}]$ included in X and Y, respectively, as shown in Fig. 5. Thus, the known and predicted resource $d(f_i,t_j)$ of f_i at time slot t_j can be estimated as

$$d(f_i,t_j) \approx \hat{d}(f_i,t_j) = x_i y_j^T. \tag{14}$$

In order to achieve accurate prediction, i.e., to make Eq. (14) true, the DRL-based matrix completion will be executed at an agent by intelligently introducing a number of rules specified by the existing matrix factorization solutions, so as to minimize the objective function $L(D,X,Y,W)$ between D and $\hat{D}$, which can be defined as

$$\begin{aligned} L(D,X,Y,W) =&\|W\cdot(D-\hat{D})\|^2 \\ =&\{\sum_{i,j=1}^{m,n} w_{ij}[d(f_i,t_j)-x_i {y_j}^T]\}^2 \\ =&\{\sum_{i,j=1}^{m,n} w_{ij}[d(f_i,t_j)-\sum_{p=1}^{k} x_{ip}y_{jp}]\}^2. \end{aligned} \tag{15}$$

Here, the parameter W denotes a weight matrix with w_{ij} as a binary variable to represent whether $d(f_i,t_j)$ is contained in D, while $x_{ip}{\in}x_i$ and $y_{jp}{\in}y_j$ are the elements of X and Y, respectively.

With matrix factorization rules in mind, it is fairly easy to minimize $L(D,X,Y,W)$ along the negative gradient direction of all $d(f_i,t_j)$ and $x_i y_j^T$, while with the possible overfitting due to the non-uniqueness of the factorization. Therefore, the regularization items along with the bias have been added into $L(D,X,Y,W)$, which can be further represented as

$$\begin{aligned} &L(D,X,Y,W,\lambda)=\sum_{i,j=1}^{m,n} w_{ij} l[d(f_i,t_j),x_i y_j^T)]+ \\ &\frac{\lambda}{2}[\sum_{i=1}^{m}(x_i x_i^T+b_{x_i}b_{x_i}^T)+\sum_{j=1}^{n}(y_j y_j^T+b_{y_j}b_{y_j}^T)] \\ &=\sum_{i,j=1}^{m,n} w_{ij}[d(f_i,t_j)-\sum_{p=1}^{k}x_{ip}y_{jp}-\hat{d}-b_{x_i}-b_{y_j}]^2+ \\ &\frac{\lambda}{2}(\sum_{i,p=1}^{m,k}x_{ip}^2+\sum_{j,p=1}^{n,k}y_{jp}^2+\sum_{i=1}^{m}b_{x_i}^2+\sum_{j=1}^{n}b_{y_j}^2), \end{aligned} \tag{16}$$

where λ is the regularization coefficient that controls the degree of regularization. There are three kinds of bias terms, i.e., the overall offset of matrix $\hat{D}$ decided by the mean of all the non-zero elements in $\hat{D}$ as $\hat{d}{=}mean(\hat{D})$, the overall offset of X as b_{x_i}, and that of Y as b_{y_j}, adopted to minimize the impact of sample characteristics.

V. DRL-based Virtual Resource Prediction

In this section, we present in detail the proposed DRL-based virtual resource prediction. Firstly, we propose the framework of resource prediction built on our previous problem formulation, and then describe the establishment of resource matrix and Markov Decision Process (MDP)-based four-tuple, followed by the DRL-based matrix factorization and its recovery for resource prediction.

A. Resource Prediction Framework

The operations of APRR illustrated in Fig. 6 work as follows. Firstly, the required resources of VNFs $f_1,\cdots,f_m$, whose features have been extracted, will be collected. Then, such data is divided into n time slots $t_1,...,t_n$ and further converted to the graded level to establish $m{\times}n$ resource matrix D, where the empty data in the n-th column refers to the resources to be predicted at time slot t_n. After that, the virtual resource prediction is modeled as a matrix completion problem, based on the grading on D, which completes D as the prediction matrix $\hat{D}$, aiming to minimize the objective function between $\hat{D}$ and D. Then, prediction matrix $\hat{D}$ is replaced by the product of two factor matrices XY^T, which are factorized from $\hat{D}$ and relevant to existing data of D, with the DRL-based matrix factorization rules, using the available solutions like SGD, Momentum and AdaDelta, to minimize the objective function between D and XY^T defined in Eq. (16). After adopting the optimal matrix factorization, we will acquire the factor matrices XY^T along with prediction matrix $\hat{D}$ where the data of the last column, $\hat{d}(f_i,t_n)$, denotes the predicted resources at time t_n, which is further recovered into prediction intervals of three kinds of resources, i.e., $\hat{d}(f_i,t_n,l)$. The details are presented below.

B. Resource Matrix Establishment

The $m{\times}n$ matrix D, which includes the required virtual resources of various VNFs at time slots t_1,...,t_n, is established through the monitoring of resource request traffic conducted by network service providers. The amount of consumed resource-related data of m kinds of VNFs will be firstly collected and its features will be extracted from temporal stability, service correlation and traffic periodicity, defined in Eqs. (2), (3) and (5). Note that the distributions of consumed resources of such data may skew with large variances, which violates the low-rank features of matrix factorization, the Box-Cox data transformation [14] has been introduced, and defined as follows:

$$boxcox[d(f_i,t_j)]=\begin{cases}\dfrac{d(f_i,t_j)^{\vartheta}-1}{\vartheta}, & if\ \vartheta\neq 0,\\ \lg[d(f_i,t_j)], & otherwise,\end{cases} \tag{17}$$

where $boxcox[d(f_i,t_j)]$ denotes the transformation value of the raw consumed resource $d(f_i,t_j)$ of VNF f_i at time t_j, and ϑ is used to control the extent of the transformation. Then, the consumed resources of such data will be divided into n time slots t_1,...,t_n, where $\|t_i\|$ may not be equal to $\|t_j\|(i{\neq}j)$, depending on their features. For VNF f_i, its required resources, i.e., computing, storage and bandwidth resources, are respectively recorded as $d(f_i,t_j,l)(l=1,2,3)$ at regular time intervals across several days, which are further integrated into an array and filled in the i-th row and j-th column of matrix D.

In this way, the resource demands of m kinds of VNFs at a total time of $\sum_{i=1}^{n-1}t_i$ are recorded and graded, and $m\times n$ matrix D can thus be established as expressed in Eq. (10), where the empty data in the n-th column is the resource to be predicted at time t_n. It's worth noting that there may be some missing data in D, which have no effects on the calculation of objective function as Eq. (16). This is because our created resource matrix D follows the low-rank characteristic by collecting as much as possible data of resource consumption in previous $n-1$ time intervals. The

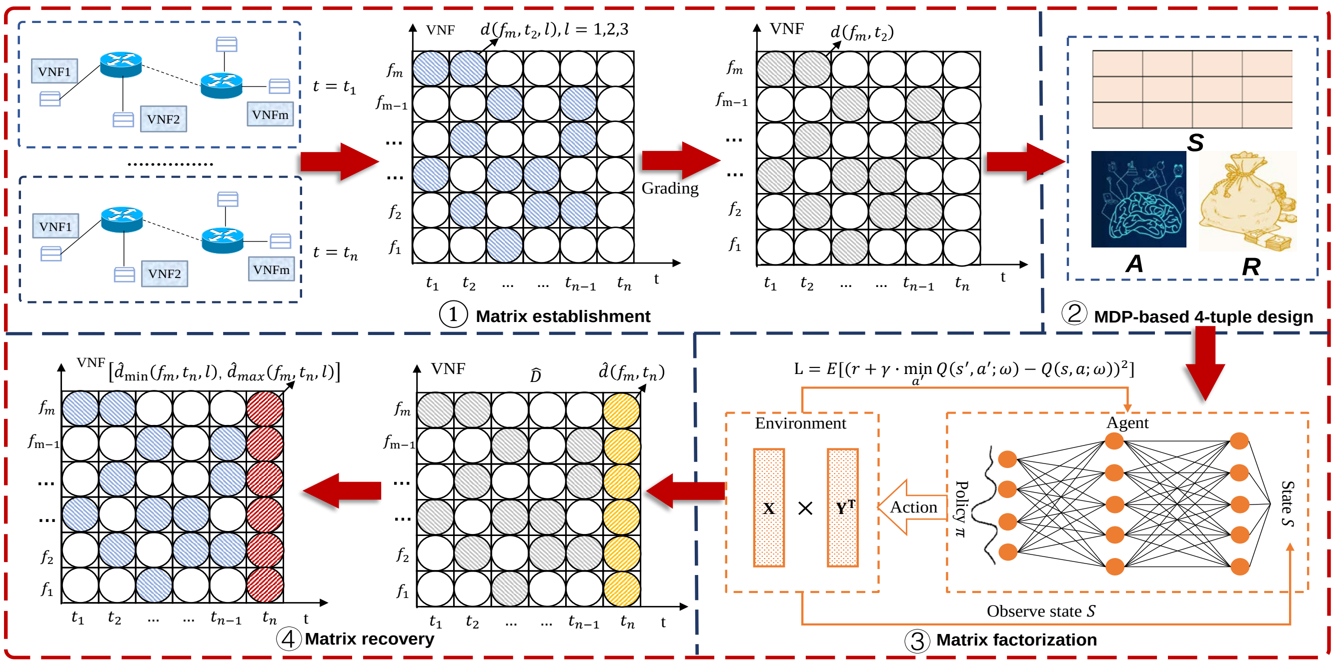


Fig. 6. The framework of required resource prediction, which includes matrix establishment, MDP-based four-tuple design, DRL-based matrix factorization and matrix recovery.

value of each interval is set depending on the features of such data in temporal stability, service correlation and traffic periodicity.

C. MDP-based 4-Tuple Design

The DRL-based virtual resource prediction is executed step by step by an agent, which controls the whole operations of the framework, to achieve the desired objective defined in Eq. (16). Essentially, it is an MDP-based iteration process with Markov features to implement iterative updates of a four-tuple $M = <\mathcal{S}, \mathcal{A}, \mathcal{R}, \gamma>$, which includes the sets of states, actions, rewards and discount factor. More specifically, $\mathcal{S}$ represents states of matrices in each step of matrix factorization, which will be changed by actions $\mathcal{A}$. Meanwhile, rewards $\mathcal{R}$ will be produced after taking actions, and γ is the discount factor satisfying $0<\gamma\leq 1$. The details of MDP-based four-tuple are defined as follows.

1) State Space: $\mathcal{S}$ contains the state of $m\times n$ graded resource matrix D with data in column n to be predicted, the state of $m \times n$ virtual resource prediction matrix $\hat{D}$, the states of factor matrices X of size $m \times k$ and Y of size $n \times k$, and iteration rule Ψ, in matrix factorization. The prediction matrix $\hat{D}$ denotes the product of X and Y^T at each iteration, and will gradually be closer to D, i.e, $\hat{D}=XY^T\approx D$. Therefore, the state $S_{\tilde{t}}$ at time slot $\tilde{t}$ can be expressed as

$$S_{\tilde{t}} = \{D, \hat{D}_{\tilde{t}}, X_{\tilde{t}}, Y_{\tilde{t}}, \Psi\}, \tag{18}$$

which is perceived by the agent, and will change to $S_{\tilde{t}+1}$ at time $\tilde{t}+1$ after the agent takes action $A_{\tilde{t}}$. Moreover, Ψ can be further denoted as

$$\Psi = \{\psi_1, \psi_2, \ldots, \psi_\kappa\}, \tag{19}$$

which includes κ alternative rules, specified by the important factorization solutions like SGD, Momentum and Adadelta. Each of them guides the matrix factorization from $\hat{D}$ to factor matrices X and Y i.e., $\hat{D} \xrightarrow{\Psi} XY^T$. The details of Ψ will be further explained in action space.

2) Action Space: $\mathcal{A}$ represents the set of taken actions to factorize the predicted matrix $\hat{D}$ at each step, which follows the iteration rules specified by Ψ. Therefore, the action $A_{\tilde{t}}$ executed at time slot $\tilde{t}$ can be defined as

$$A_{\tilde{t}} = \{a_1, a_2, \cdots, a_\kappa\}, \tag{20}$$

where $a_i (i\in[1,\kappa])$ stands for an action to implement matrix factorization, following the rule ψ_i which has been used in the current matrix factorization.

In order to facilitate the understanding, three rules ψ_1, ψ_2, and ψ_3, used in SGD, Momentum and Adadelta, respectively, associated with actions a_1, a_2 and a_3 to obtain factor matrices X and Y and update the parameters of the objective function, can be described as follows.

SGD-based rule. This rule used to iteratively update multiple variables, i.e., x_{ip}, y_{jp}, b_{x_i} and b_{y_j}, defined in Eq. (16), is denoted as

$$\begin{cases} x'_{ip} = x_{ip} - \alpha\dfrac{\partial L}{\partial x_{ip}} = x_{ip} + \alpha[\displaystyle\sum_{j=1}^{T} 2e_{ij}y_{jp} - \lambda x_{ip}], \\ y'_{jp} = y_{jp} - \alpha\dfrac{\partial L}{\partial y_{jp}} = y_{jp} + \alpha[\displaystyle\sum_{n=1}^{M} 2e_{ij}x_{ip} - \lambda y_{jp}], \\ b'_{x_i} = b_{x_i} - \alpha\dfrac{\partial L}{\partial b_{x_i}} = b_{x_i} + \alpha[\displaystyle\sum_{j=1}^{T} 2e_{ij} - \lambda b_{x_i}], \\ b'_{y_j} = b_{y_j} - \alpha\dfrac{\partial L}{\partial b_{y_j}} = b_{y_j} + \alpha[\displaystyle\sum_{i=1}^{m} 2e_{ij} - \lambda b_{y_j}], \end{cases} \tag{21}$$

where $e_{ij} = d(f_i, t_j) - \sum_{p=1}^{k} x_{ip}y_{jp} - b - b_{x_i} - b_{y_j}$.

Momentum-based rule. This rule is built on SGD by dynamically adjusting the learning rate with the accumulation of all gradients of the objective function. The update rule of the variables x_{ip}, y_{jp}, b_{x_i} and b_{y_j} is conducted as

$$\begin{cases} x_{ip}^{'} = x_{ip} - \alpha \sum_{i=1}^{\tilde{t}} \delta^{\tilde{t}-i} \frac{\partial L}{\partial x_{ip}}, \\ y_{jp}^{'} = y_{jp} - \alpha \sum_{i=1}^{\tilde{t}} \delta^{\tilde{t}-i} \frac{\partial L}{\partial y_{jp}}, \\ b_{x_i}^{'} = b_{x_i} - \alpha \sum_{i=1}^{\tilde{t}} \delta^{\tilde{t}-i} \frac{\partial L}{\partial b_{x_i}}, \\ b_{y_j}^{'} = b_{y_j} - \alpha \sum_{i=1}^{\tilde{t}} \delta^{\tilde{t}-i} \frac{\partial L}{\partial b_{y_j}}, \end{cases} \tag{22}$$

where δ is the constraint factor, while $\frac{\partial L}{\partial x_{ip}}$, $\frac{\partial L}{\partial y_{jp}}$, $\frac{\partial L}{\partial b_{x_i}}$ and $\frac{\partial L}{\partial b_{y_j}}$ represent the descending gradients of x_{ip}, y_{jp}, b_{x_i} and b_{y_j} at time slot t_i, which can be obtained from Eq. (21).

AdaDelta-based rule. This rule introduces a constraint into learning rate α to improve the accuracy of matrix factorization, as $\alpha^{'} = \alpha \frac{1}{\sqrt{v_{\tilde{t}} + \epsilon}}$, where ϵ is a small constant to avoid the denominator being 0. $v_{\tilde{t}}$ denotes the constraint which is iterated following $v_{\tilde{t}} = (1-\beta)\sum_{i=1}^{\tilde{t}} \beta^{\tilde{t}-i} g_i^2$, with β as a constant and $g_{\tilde{t}}$ as the gradient of descent at time $\tilde{t}$. The update rule of the variables x_{ip}, y_{jp}, b_{x_i} and b_{y_j} is conducted as

$$\begin{cases} x_{ip}^{'} = x_{ip} - \frac{\alpha}{\sqrt{(1-\beta)\sum_{i=1}^{\tilde{t}} \beta^{\tilde{t}-i} \frac{\partial L}{\partial x_{ip}}^2 + \epsilon}} \frac{\partial L}{\partial x_{ip}}, \\ y_{jp}^{'} = y_{jp} - \frac{\alpha}{\sqrt{(1-\beta)\sum_{i=1}^{\tilde{t}} \beta^{\tilde{t}-i} \frac{\partial L}{\partial y_{jp}}^2 + \epsilon}} \frac{\partial L}{\partial y_{jp}}, \\ b_{x_i}^{'} = b_{x_i} - \frac{\alpha}{\sqrt{(1-\beta)\sum_{i=1}^{\tilde{t}} \beta^{\tilde{t}-i} \frac{\partial L}{\partial b_{x_i}}^2 + \epsilon}} \frac{\partial L}{\partial b_{x_i}}, \\ b_{y_j}^{'} = b_{y_j} - \frac{\alpha}{\sqrt{(1-\beta)\sum_{i=1}^{\tilde{t}} \beta^{\tilde{t}-i} \frac{\partial L}{\partial b_{y_j}}^2 + \epsilon}} \frac{\partial L}{\partial b_{y_j}}. \end{cases} \tag{23}$$

Only one rule can be selected by agent in each iteration of matrix factorization, that is, only one of $a_1, a_2, \cdots, a_\kappa$ can be implemented while the others are ignored. This means that, for each selected action a_i, there are $m \times k$ entries of X and $k \times n$ entries of Y to be updated for building an $m \times n$ temporary matrix $\hat{D}$, which has $m \times n$ new entries closer to resource matrix D. $A_{\tilde{t}}$ is executed at time slot $\tilde{t}$ to interact with the environment and thus assist it to enter the next state $S_{\tilde{t}+1}$.

3) Reward Space: $\mathcal{R}$ is the set of instant rewards $R_{\tilde{t}}$ which is obtained after the agent has taken action $A_{\tilde{t}}$ at state $S_{\tilde{t}}$. Consistent with the modified optimization goal defined in Eq. (16), $R_{\tilde{t}}$ is set to minimize the objective function between resource matrix D and prediction matrix $\hat{D}$ that is factorized as XY^T, i.e., maximizing the negative of $L(D, X, Y, W, \lambda)$ defined in Eq. (16), so as to improve the accuracy of prediction. Therefore, $R_{\tilde{t}}$ is denoted as

$$R_{\tilde{t}} = -L(D, X, Y, W, \lambda). \tag{24}$$

D. DRL-based Matrix Factorization

The DRL-based matrix factorization, from predicted matrix $\hat{D}$ to the product of factor matrices XY^T, is conducted through combing DRL with κ optional rules, specified by like SGD, Momentum and Adadelta, to minimize the objective function between D and XY^T defined in Eq. (16). The agent installed Deep Q-networks (DQN) will dynamically choose one of the above κ rules, depending on their rewards, to update the iteration rule of matrix factorization. The optimization goal of training is to learn optimal policy $\pi^\star$ so as to maximize the cumulative discount return over several iterations, which is built on the action-value function $Q_\pi(S_{\tilde{t}}, A_{\tilde{t}})$ as

$$Q_\pi(S_{\tilde{t}}, A_{\tilde{t}}) = \mathbb{E}_\pi[\sum_{i=\tilde{t}}^{\infty} \gamma^{i-\tilde{t}} R_{\tilde{t}} | S_{\tilde{t}} = s, A_{\tilde{t}} = a], \tag{25}$$

where $\gamma (0 < \gamma < 1)$ is the discount factor to be used as the coefficient for each $R_{\tilde{t}}$. The optimal policy $\pi^\star$ will be used to guide the agent to obtain a greater $Q(S_{\tilde{t}}, A_{\tilde{t}})$ than other policies, thus $\pi^\star$ can be denoted as

$$\pi^\star = \arg\max_\pi Q_\pi(S_{\tilde{t}}, A_{\tilde{t}}), \tag{26}$$

with the optimal action-value function $Q^\star(S_{\tilde{t}}, A_{\tilde{t}}) = \max_\pi Q_\pi(S_{\tilde{t}}, A_{\tilde{t}})$.

The DQN-based architecture for matrix factorization includes experiential replay buffer, experimental environment, Q-network and target Q-network. Among them, the experience replay buffer is responsible for the store of previous training experience in the form of $(S_{\tilde{t}}, A_{\tilde{t}}, R_{\tilde{t}}, S_{\tilde{t}+1})$. The environment mainly interacts with the agent, executes actions imputed by agent and then returns the new state and obtained reward. With the help of them, the Q-network with the built-in parameter θ is used to fit the state-value function $Q^\star(S_{\tilde{t}}, A_{\tilde{t}}; \theta)$ for each pair of $S_{\tilde{t}}$ and $A_{\tilde{t}}$, while the target Q-network with parameter θ^- aims to fit the optimal target Q value as a reference for Q-network to be updated.

With $\epsilon-greedy$ policy in mind, the agent will first randomly select an action, with a probability of ϵ ($\epsilon_{rd} \leq \epsilon < 1$), from action space $\mathcal{A}$, where there are κ optional actions $a_1, \cdots, a_\kappa$, or greedily take the optimal action that maximizes $Q(S_{\tilde{t}}, A_{\tilde{t}}; \theta)$ if $0 < \epsilon < \epsilon_{rd}$, which can be denoted as

$$A_{\tilde{t}} = \begin{cases} \arg\max_A Q(S_{\tilde{t}}, A; \theta), & 0 < \epsilon < \epsilon_{rd}, \\ random_{A \in \mathcal{A}} \quad A, & \epsilon_{rd} \leq \epsilon < 1. \end{cases} \tag{27}$$

Here, $\epsilon_{rd} \in [0, 1]$ is an exploration probability parameter which is initially set to a larger value and decreases gradually over the iteration, approaching zero finally. Each $x_{ip} \in X$ and $y_{jp} \in Y$ ($1 \leq i \leq m$, $1 \leq p \leq k$, and $1 \leq j \leq n$) will follow the same rules, which have been partially defined in Eq. (21), Eq. (22) or Eq. (23), depending on their selected actions, to be updated. Meanwhile, the agent will execute action $A_{\tilde{t}}$, with reward $R_{\tilde{t}}$ following Eq. (24), and then changes into the new state $S_{\tilde{t}+1}$. One training experience $(S_{\tilde{t}}, A_{\tilde{t}}, R_{\tilde{t}}, S_{\tilde{t}+1})$ is thus generated, which is further stored in experiential replay buffer B and continuously tracked during the iteration.

In order to improve prediction accuracy, a mini-batch of $\mathbb{M}$ transition samples with less factorization errors will be selected from B for the training of Q-network and target Q-network. Given k-th transition sample, the factorization error χ_k of it can be expressed as

$$\chi_k = \sum_{i=1,j=1}^{m,n} \|d(f_i,t_j) - x_i y_j^T\|. \tag{28}$$

All transition samples included in mini-batch are stored in the form of $(S_{\tilde{t}}, A_{\tilde{t}}, R_{\tilde{t}}, S_{\tilde{t}+1})$. To eliminate the correlation of samples, each of them will be randomly selected to train Q-network and target Q-network.

On this basis, the update of Q-network θ and target Q-network θ^- are conducted to fit the optimal state-value function $Q^\star(S_j,A_j;\theta)$ and target $\hat{Q}^\star(S_j,A_j;\theta^-)$. The structure of two neural networks as well as the initial value of θ and θ^- are set to be the same, respectively. Q-network is updated at each step of iteration, based on the loss function $L(\theta)$ as

$$L(\theta) = \mathbb{E}\left[\frac{1}{2}(\varrho_j - Q(S_j,A_j;\theta))^2\right], \tag{29}$$

where $Q(S_j,A_j;\theta)$ is the Q value predicted by the current Q-network and ϱ_j is the target for update of θ, determined by

$$\varrho_j = \begin{cases} R_j, & \text{episode ends at } j{+}1, \\ R_j + \gamma \max\limits_{A'} \hat{Q}(S_j^{'}, A^{'};\theta^-), & \text{otherwise.} \end{cases} \tag{30}$$

Here, $\hat{Q}(S_j^{'}, A^{'};\theta^-)$ denotes the Q value predicted by target Q-network with the state $S_j^{'}$ at the next moment, and possible action $A^{'}$ at this state. In this way, to precisely learn $Q^\star(S_j,A_j;\theta)$, gradient descent is applied to minimize the loss function $L(\theta)$, defined in Eq. (29), so as to update the Q-network following

$$\begin{aligned} \theta' =& \theta - \alpha^\star \nabla_\theta L(\theta) \\ =& \theta - \alpha^\star \mathbb{E}[R_j + \gamma \max_{A'} \hat{Q}(S_j^{'}, A^{'};\theta^-) \\ & - Q(S_j,A_j;\theta)]^2 \nabla_\theta Q(S_j,A_j;\theta). \end{aligned} \tag{31}$$

Different from Q-network, target Q-network is only updated each C steps through copying the value of θ and remains constant between individual updates.

The pseudocode of the DQN-based matrix factorization training has been listed in **Algorithm 1**. First of all, the initialization is conducted as shown in Lines 1-4. In Lines 6-9, the experience transitions are sampled based on $\epsilon-greedy$, where $A_{\tilde{t}}$ is selected and executed in Line 7 while $R_{\tilde{t}}$ and $S_{\tilde{t}+1}$ are observed from the environment in Line 8, which are later jointly stored into the replay buffer B as $(S_{\tilde{t}}, A_{\tilde{t}}, R_{\tilde{t}}, S_{\tilde{t}+1})$ in Line 9. On this basis, an error-prioritized mini-batch of transitions $(S_j, A_j, R_j, S_j^{'})$ are sampled from B in Line 10. Lines 11-13 conduct the update of Q-network with loss function defined in Line 12 and the gradient-descent-based iterations of parameter executed in Line 13. After experiencing C steps of training, the target Q-network is also updated in Line 15.

E. Matrix Recorvery

After the training of DQN-based matrix factorization has been carried out as mentioned above, the optimal matrix factorization along with the optimal prediction matrix $\hat{D}$ and factor

Algorithm 1: DQN-based matrix factorization

```
1  Initialize replay buffer $B$
2  Initialize $Q$ and $\hat{Q}$ with random weights $\theta$ and $\theta^- = \theta$
3  foreach episode $E$ do
4      Initialize the initial state $s_0$
5      for $t$= 1 to max-episode-length do
6          Generate a radom value for $\epsilon$
7          Select and execute
           $A_{\tilde{t}} = \begin{cases} \arg\max_A Q(S_{\tilde{t}}, A;\theta), & 0 < \epsilon < \epsilon_{rd}, \\ random_{A \in \mathcal{A}} \quad A, & \epsilon_{rd} \leq \epsilon < 1. \end{cases}$
8          Observe reward $R_{\tilde{t}}$ and state $S_{\tilde{t}+1}$
9          Store transition $(S_{\tilde{t}}, A_{\tilde{t}}, R_{\tilde{t}}, S_{\tilde{t}+1})$ into $B$
10         Sample error-prioritized mini-batch of
             transitions $(S_j, A_j, R_j, S_j^{'})$ from $B$
11         Set
           $\varrho_j = \begin{cases} R_j, & E \text{ ends at } j{+}1 \\ R_j + \gamma \max_{A'} \hat{Q}(S_j^{'}, A^{'};\theta^-), & \text{otherwise} \end{cases}$
12         $L(\theta) \leftarrow \mathbb{E}\left[\frac{1}{2}(\varrho_j - Q(S_j,A_j;\theta))^2\right]$
13         Update Q-network as $\theta' \leftarrow \theta - \alpha^\star \nabla_\theta L(\theta)$
14         if $t$ mod $C$ then
15             Update target Q-network as $\theta^- \leftarrow \theta$
```

matrices XY^T are obtained. Therefore, the predicted resource of various VNFs at time t_n can be thus acquired from the data in the last column of $\hat{D}$, i.e., $\hat{d}(f_i,t_n)(1 \leq i \leq m)$. It is worth noting that the prediction resource obtained is a grading value, based on our proposed resource grading model. Therefore, $\hat{d}(f_i,t_n)$ is further recovered into prediction intervals of three kinds of resources, i.e., $\hat{d}(f_i,t_n,l)(l=1,2,3)$ which belongs to the prediction intervals $[\hat{d}_{min}(f_i,t_n,l), \hat{d}_{max}(f_i,t_n,l)]$ following Eqs. (11) and (12) in Section IV.

VI. Performance Evaluation

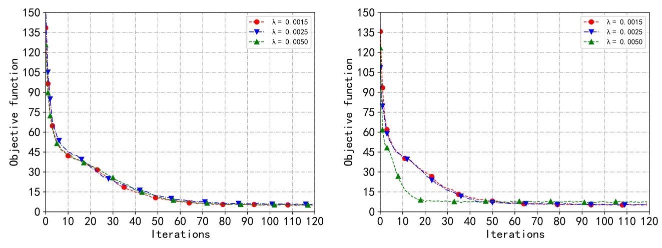
In this section, comprehensive simulation experiments, running on a server equipped with an Intel(R) Xeon(R) CPU E5-2637 v4@3.50GHz with 64G RAM, have been implemented to estimate the performance of APRR. First, the simulation setup is introduced, followed by the evaluation metrics. Then, the performance results of APRR, compared with state-of-the-art approaches, i.e, SGD, Momentum, AdaDelta, AMF [14], and LPAW* [15] for resource prediction, are presented. AMF is a learning-based solution built on SGD to perform online QoS prediction for candidate services. Due to limited measurement data used in resource prediction, the top-sparse auto-encoder designed to compress the original high-dimensional workload in L-PAW [15] has not been considered, thus we abbreviate it to LPAW*. Essentially, LPAW* is a DL-based solution, which exploits LSTM-based neural networks to predict the required resources.

A. Simulation Setup

The simulation experiments are executed in Python 3.6.12 with Pytorch-based Q-network and target Q-network. The Q-network is composed of five fully connected layers, i.e., one input layer, three hidden layers and one output layer with $(m\times n + m\times k + k\times n)$, 4800, 480, 48 and 3 neurons, respectively. The number of neurons in the input layer is related to the dimensions of matrices D, X and Y on account of its connection to the current state $S_{\tilde{t}}$, and the output layer indicates optional actions, i.e., κ matrix factorization rules. The target Q-network owns the same structure as Q-network, and is updated by copying its weights every 4 steps of training.

The dataset used for the performance evaluation is chosen from [32], where two-day resource-consumption-related traffic data is selected and expanded into two resource matrices by adding random noise. One matrix is considered as a training set for resource prediction, and the other is a test set. Furthermore, there are graded resource requirements of 400 VNFs at 480 time slots in each of matrix, which are recorded every 60 minutes for twenty days. In order to simulate the failure that may occur in the actual data acquisition process, we randomly zero 10% of data in each column of resource matrix. In this way, the resource prediction problem of 400 VNFs at time 480 is transformed into a matrix-factorization-based matrix completion problem of 400×480 matrix. Three important rules associated with taken actions, used in SGD, Momentum, and AdaDelta (κ=3), are introduced into agents in our simulations for DRL-based matrix factorization.

Unless specially noted, the learning rate $\alpha^{\star}$ and discount factor γ of DRL are set to 0.005 and 0.9, respectively, and the regulatory factor ϵ is set to $1e-8$ for AdaDelta to avoid its denominator being 0. Both δ and β are set to 0.0015. The parameters of $\epsilon-greedy$ rule are set to ϵ_{rd}= 0.3. The capability of replay buffer B is 1600, where a set of experiences with mini-batch size of $\mathbb{M}$=400 will be sampled for further training at each episode. The rank of X and Y is set to 10. The iteration of matrix factorization will terminate if the error $\sum^{(i,j)}\sqrt{w_{ij}(d_{f_i,t_j}-\hat{d}_{f_i,t_j})^2/(m\times n)}(w_{ij}\in\{1,0\})$ between resource matrix D and prediction matrix $\hat{D}$ is less than the given threshold, set as 10^{-4}, in two consecutive iterations, or the number of iterations C reaches the upper limit, set as 120, during the simulations.



(a) $\alpha = 0.0025$ (b) $\alpha = 0.0030$

Fig. 7. Convergence of APRR in term of objective function with different learning rate α and regularization coefficient λ.

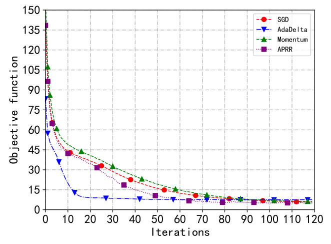
B. Evaluation Metrics

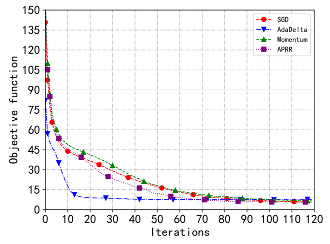
There are six metrics presented to investigate the performance of APRR, i.e., objective function, fitting errors, fitting error ratios, prediction errors, prediction error ratios and computing cost efficiency, which are introduced in details as follows.

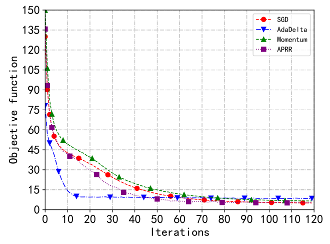
- **Objective function.** The objective function between prediction matrix $\hat{D}$ and resource matrix D, defined as Eq. (16) with the regularization and bias items, is counted after each iteration, which indicates the convergence of APRR and other approaches during the matrix factorization process. The more stable the objective function is with a small value, the better the convergence of APRR will be, which further illustrates that it can find a stable solution for the optimization problem in **Section IV-C**.
- **Fitting errors.** The metric evaluates the fitting accuracy between known resource data in the first $(n-1)$ columns of prediction matrix $\hat{D}$ and original resource matrix D, which is defined as $\sum^{(i,j)}\sqrt{w_{ij}(d_{f_i,t_j}-\hat{d}_{f_i,t_j})^2/[m\times(n-1)]}$, where $w_{ij}\in\{1,0\}$, $i\in[1,m]$, and $j\in[1,n-1]$. For given D, the smaller the fitting error is, the better the closeness of fit is on the known resource data.
- **Fitting error ratios.** It is the ratio of fitting errors to their corresponding real resource values, defined as $\sqrt{\sum^{(i,j)}(d_{f_i,t_j}-\hat{d}_{f_i,t_j})^2/\sum^{(i,j)}{d_{f_i,t_j}}^2}$, where $i\in[1,m]$, and $j\in[1,n-1]$. The smaller the fitting error ratio is, the more accurate the fitting is on the known resource data.
- **Prediction errors.** This metric is referred to as the prediction accuracy of the $n-th$ column of $\hat{D}$. Given resource matrix D, the metric can be expressed as $\sum^{(i)}\sqrt{w_{in}(d_{f_i,t_n}-\hat{d}_{f_i,t_n})^2/m}$, where $w_{in}\in\{1,0\}$, and $i\in[1,m]$. The prediction accuracy is negatively correlated with the prediction error, a smaller prediction error means better performance on prediction.
- **Prediction error ratios.** It evaluates the relative value of prediction errors to the predicted resource values, defined as $\sqrt{\sum^{(i)}(d_{f_i,t_n}-\hat{d}_{f_i,t_n})^2/\sum^{(i)}{d_{f_i,t_n}}^2}$, where $i\in[1,m]$. A smaller prediction error ratio means better performance on prediction.
- **Computing cost efficiency.** This metric is defined as the quotient of the inverse prediction errors to the computing costs, which represents the efficiency of the prediction in matrix factorization. Since the computing resources are consumed in each iteration, its cost is proportional to the number of iterations. Lower prediction errors with smaller iteration times represent higher efficiency.

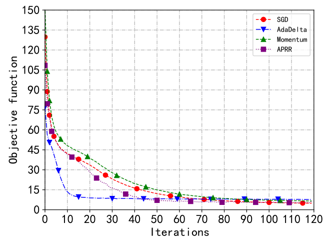
C. Simulation Results

1) Convergence of APRR: In Fig. 7, the convergence of APRR is evaluated in the form of objective function during the matrix factorization process over approximately 120 rounds of iterations. The matrix factorization learning rate α is set as 0.0025 and 0.0030, while the regularization coefficient λ is 0.0015, 0.0025 and 0.0050. On the whole, it can be seen from Fig. 7 that the objective functions gradually decrease to 0 regardless of the values of training parameters, which greatly illustrates the effectiveness of APRR and proves that it can find a stable solution for the resource prediction problem. Furthermore, when α and λ are 0.0030 and 0.0050, respectively, as









(a) $\alpha = 0.0025$ and $\lambda = 0.0015$ (b) $\alpha = 0.0025$ and $\lambda = 0.0025$ (c) $\alpha = 0.0030$ and $\lambda = 0.0015$ (d) $\alpha = 0.0030$ and $\lambda = 0.0025$

Fig. 8. Convergence of APRR, SGD, Momentum and AdaDelta in terms of objective function with different learning rate α and regularization coefficient λ.

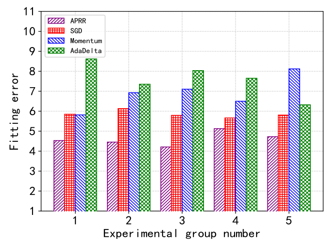
shown in Fig. 7(b), the final objective function is not small enough although it converges the fastest. Therefore, in order to achieve accurate resource prediction, we will not consider the case where λ is 0.0050 in subsequent discussions.

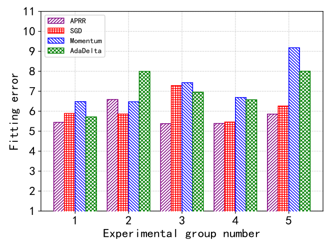
The convergence of APRR along with SGD, Momentum and AdaDelta is illustrated in Fig. 8, which is evaluated in the form of objective function. The settings of α are equal to 0.0025 and 0.0030, while λ is set to 0.0015 and 0.0025, respectively. Simulation results indicate that all of them tend to converge during the iteration progress and finally learn to decompose matrix $\hat{D}$ in a high-precision manner with low objective function. Furthermore, it can be seen from Fig. 8 that the objective functions of SGD, Momentum and APRR decrease slower obviously than that of AdaDelta, and APRR is slightly faster than SGD and Momentum. APRR is not as sharp as AdaDelta in convergence, but can get to the minimum objective function as 5.496 while the final objective function of AdaDelta is 7.939, which greatly illustrates that APRR can achieve resource prediction more accurately.

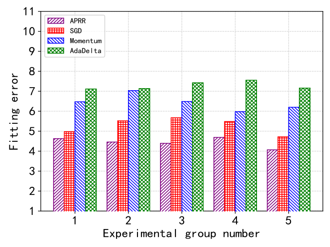
It is because that AdaDelta is easy to be trapped into the local optimal solution and halrdly declines in the late period of convergence, while APRR can alleviate this issue faster by jointly using SGD and Momentum and reach a better solution in a more flexible manner. By combining the advantages of SGD, Momentum and AdaDelta, APRR is proved to achieve an equilibrium between convergence rate and prediction accuracy, so as to fit the real data D more quickly and more accurately.

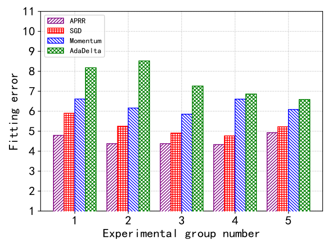
2) Comparisons in fitting accuracy: The fitting errors for APRR, SGD, Momentum and AdaDelta to predict the required resources are illustrated in Fig. 9, where the matrix factorizations are implemented in five different experimental groups with the same matrices to be predicted. The matrix factorization learning rate α is set to 0.0025 and 0.0030 and the regularization coefficient λ is 0.0015 and 0.0025. It is shown in Fig. 9(c) that all of them can achieve the minimum fitting errors when α=0.0030 and λ = 0.0015, in such simulation scenarios compared with other parameter settings. Furthermore, it can be seen from Fig. 9 that APRR performs the best on average fitting errors, with reductions of 21.20%, 39.32% and 33.13% on average in Fig. 9(a), 6.85%, 18.71% and 20.96% in Fig. 9(b), 15.70%, 38.90% and 30.88% in Fig. 9(c) as well as 12.47%, 39.07% and 27.21% in Fig. 9(d), compared with SGD, Momentum and AdaDelta, respectively. Such achievements illustrate that predicted matrix $\hat{D}$ obtained by APRR keeps most similarities with the original D in term of known data.

In order to reduce the impact of some errors with large values of their own on the overall errors, the fitting error ratios of APRR, SGD, Momentum and AdaDelta are illustrated in Fig. 10. The settings of α and λ are the same as Fig. 9. Regardless of the experimental parameter settings, it can be seen that APRR can still achieve the best fitting accuracy with the minimum fitting error ratios, compared with SGD,



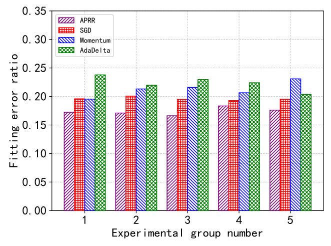


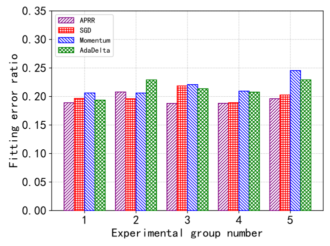


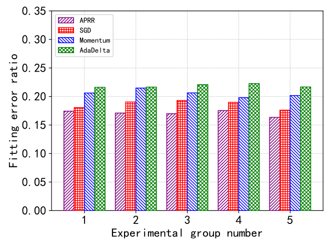


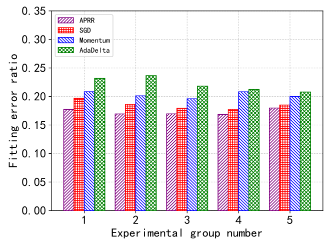
(a) $\alpha = 0.0025$ and $\lambda = 0.0015$ (b) $\alpha = 0.0025$ and $\lambda = 0.0025$ (c) $\alpha = 0.0030$ and $\lambda = 0.0015$ (d) $\alpha = 0.0030$ and $\lambda = 0.0025$

Fig. 9. Fitting errors of APRR, SGD, Momentum and AdaDelta with different learning rate α and regularization coefficient λ.



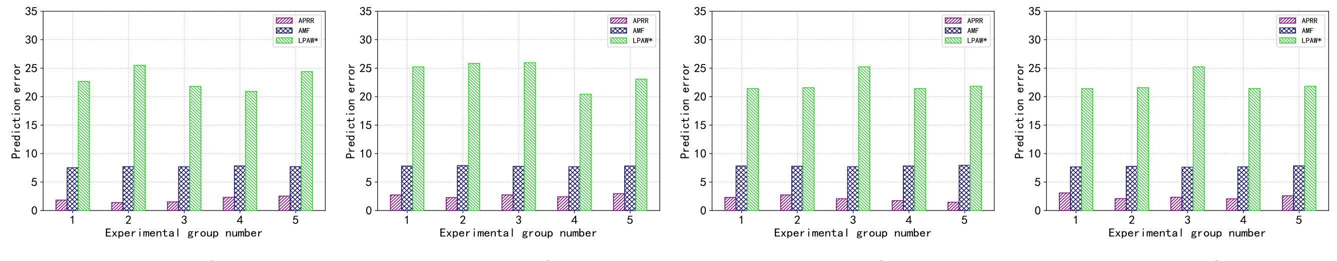






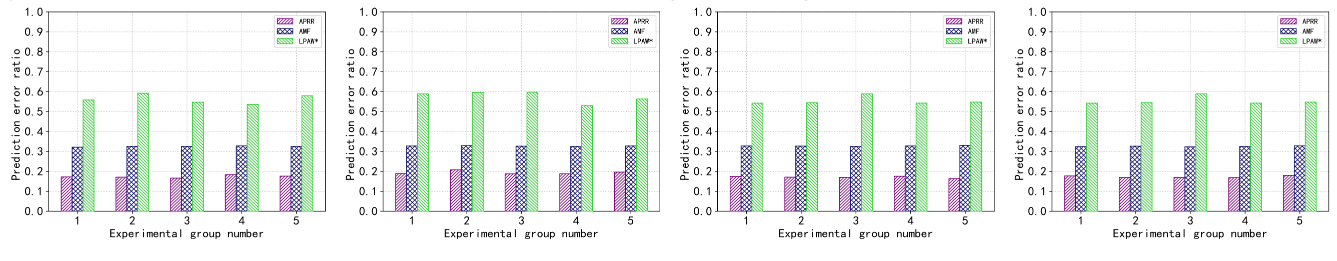
(a) $\alpha = 0.0025$ and $\lambda = 0.0015$ (b) $\alpha = 0.0025$ and $\lambda = 0.0025$ (c) $\alpha = 0.0030$ and $\lambda = 0.0015$ (d) $\alpha = 0.0030$ and $\lambda = 0.0025$

Fig. 10. Fitting error ratios of APRR, SGD, Momentum and AdaDelta with different learning rate α and regularization coefficient λ.



(a) $\alpha = 0.0025$ and $\lambda = 0.0015$ (b) $\alpha = 0.0025$ and $\lambda = 0.0025$ (c) $\alpha = 0.0030$ and $\lambda = 0.0015$ (d) $\alpha = 0.0030$ and $\lambda = 0.0025$

Fig. 11. Prediction errors of APRR, AMF and LPAW* with different learning rate α and regularization coefficient λ.



(a) $\alpha = 0.0025$ and $\lambda = 0.0015$ (b) $\alpha = 0.0025$ and $\lambda = 0.0025$ (c) $\alpha = 0.0030$ and $\lambda = 0.0015$ (d) $\alpha = 0.0030$ and $\lambda = 0.0025$

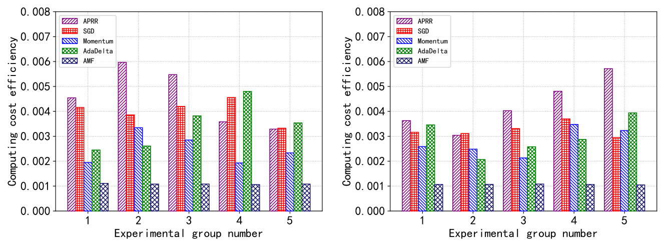
Fig. 12. Prediction error ratios of APRR, AMF and LPAW* with different learning rate α and regularization coefficient λ.

Momentum and AdaDelta, which further indicates that APRR can achieve the most accurate prediction of resources in future.

Furthermore, it is worth noting that the predicted resources of various VNFs in our simulations are represented with the graded values based on Eq. (10), which are further recovered to scopes of resource requirements, i.e., $[\hat{d}_{min}(f_i, t_n, l), \hat{d}_{max}(f_i, t_n, l)]$ defined in Eq. (11), for computing, storage and bandwidth resources with $l = 1, 2, 3$. The prediction accuracy verified in this section is represented in the form of total errors between graded $d(f_i, t_n)$ and $\hat{d}(f_i, t_n)$, while the recovery accuracy is verified in Fig. 4.

3) Comparisons in prediction accuracy: To validate the efficiency of our design, we compare the prediction errors of APRR with that of AMF and LPAW*, where α is set as 0.0025 and 0.0030 while λ is equal to 0.0015 and 0.0025 in Figs. 11(a)-(d). It can be seen that APRR achieves prediction errors as 1.923 in Fig. 11(a), 2.627 in Fig. 11(b), 2.060 in Fig. 11(c), and 2.433 in Fig. 11(d) on average, respectively, while the prediction errors of AMF and LPAW* are 7.728 and 22.935 on average. Therefore, APRR is proved to reduce prediction errors by 70.74% and 90.14%, compared with AMF and LPAW*, which greatly shows that APRR performs better predictions of unknown resources. Furthermore, compared with fitting errors shown in Fig. 9, APRR owns smaller prediction errors, indicating that it has great generalization performance and its fitted model can perform resource prediction effectively.

In addition, the prediction error ratios of APRR, AMF and LPAW* are illustrated in Fig. 12. The settings of α and λ are the same as Fig. 11. Regardless of the experimental parameter settings, it is clear that APRR still holds the smallest prediction error ratios, with reductions of 46.52% and 69.10% on average in Fig. 12(a), 41.60% and 67.00% in Fig. 12(b), 47.86% and 69.14% in Fig. 12(c) as well as 43.98% and 67.04% in Fig. 12(d), compared with AMF and LPAW*. The results indicate that APRR is effective to predict the required resources using DRL-based matrix completion, especially when the amount of data in a given test set is small.



(a) $\alpha = 0.0025$ and $\lambda = 0.0015$ (b) $\alpha = 0.0030$ and $\lambda = 0.0015$

Fig. 13. Computing cost efficiency of APRR, SGD, Momentum, AdaDelta and AMF with different learning rate α.

4) Comparison in computing cost efficiency: Fig. 13 evaluates the computing cost efficiency of APRR, SGD, Momentum, AdaDelta and AMF in different experiment groups. It is worth noting that LPAW* is not a matrix-factorization-based solution of resource prediction, thus we set the number of its operations as 120, ensuring the required resources can be effectively inferred. Differently, the iterations of APRR, SGD, Momentum, AdaDelta and AMF always vary due to additional action exploration. Therefore, we introduce this metric rather than only using iteration steps to estimate the efficiency of computing cost. The settings of α are 0.0025 and 0.0030, and λ is equal to 0.0015, in which the prediction error of APRR is the lowest. In Fig. 13, APRR almost always performs the best in computing cost efficiency, even with achievements of 289.7% compared with AMF on average. This is because the prediction error of APRR keeps to a minimum for the most parts while its computing cost is the lowest.

5) Time Complexity: The time complexity of APRR, SGD, Momentum, AdaDelta, AMF and LPAW* has been elaborated in TABLE III.

The time complexity of APRR is mainly determined by matrix factorization and training process. Specifically, the time complexity of matrix factorization depends on the shape of $M \times N$ matrix and the steps Q to factorize such a matrix, given by $O(QMN)$. As for the training process, the time complexity is dependent on the design of training looping and neural net-

TABLE III
TIME COMPLEXITY

Solutions	Time complexity
APRR	$O[QMNEP(MK{+}KN{+}MN)\|F\|\|G\|\|\mathcal{A}\|]$
Momentum	$O(QMN)$
SGD	$O(QMN)$
AdaDelta	$O(QMN)$
AMF	$O(QMN)$
LPAW*	$O(EM^2NH)$

works. The training looping consists of two-layer looping, i.e., outer looping and inner looping, determined by the episode E and the epoch design P, respectively, with the time complexity of $O(EP)$. Let $|F|$ and $|G|$ represent the number of two-column neurons of hidden layer in neural networks. The neural network are scaled to the size of action space $\mathcal{A}$ and state space $\mathcal{S}$ with $MK+KN+MN$ and $|\mathcal{A}|$ neurons in input layer and output layer, where K stands for the settled rank of matrix factorization. Therefore, the time complexity of APRR can be expressed as $O[QMNEP(MK+KN+MN)|F||G||\mathcal{A}|]$.

Being the important components of APRR, SGD, Momentum and AdaDelta are introduced to factorize $M\times N$ matrix. The time complexity of them is mainly decided by the size of $M\times N$ matrix and the steps Q to factorize matrix, and can be given by $O(QMN)$.

AMF introduces data transformation and SGD-based online learning to factorize $M\times N$ matrix. The time complexity of it is dependent on the size of $M\times N$ matrix and the steps Q to factorize this matrix, represented as $O(QMN)$.

LPAW* exploits LSTM-based and linear neural networks, which consist of LSTM layer and linear layer, respectively, to predict required resource. The time complexity in LSTM layer is scaled to the product of the hidden size H and in-out size of LSTM neural network, where its input size is equal to the shape of resource matrix MN and its output size equals the value of matrix rows M, also being equal to the number of output of linear layer. Let E represent the training episode. Then, the time complexity of LPAW* can be given by $O(EM^2NH)$.

VII. CONCLUSION

This paper has investigated the accurate DRL-based prediction of required virtual resources to fulfill the ever-increasing requirements of diversified network services. Based on our quantitative analyses, the important inherent features of network traffic, referring to temporal stability, service correlation and traffic periodicity, have been confirmed. Therefore, the required network resources have been modelled as a time-variant network matrix, which includes a number of elements, obtained from network measurement, and the missing elements needed to be inferred. In order to obtain accurate missing required resources, DRL-based matrix factorization with a set of important available rules has been introduced into APRR and intelligently executed to minimize the prediction errors. Furthermore, the prioritized learning experience samples, built on rewards, are used for model training with quicker convergence. Simulation results show that APRR achieves accurate prediction of the required resources compared with the state-of-the-art approaches.

REFERENCES

[1] R. Mijumbi, J. Serrat, J. L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network Function Virtualization: State-of-the-Art and Rresearch Challenges," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 236–262, 2015.

[2] H. Hawilo, M. Jammal, and A. Shami, "Network Function Virtualization-Aware Orchestrator for Service Function Chaining Placement in the Cloud," *IEEE Trans. Serv. Comput.*, vol. 37, no. 3, pp. 643–655, 2019.

[3] X. Wei, L. Li, X. Li, X. Wang, S. Gao, and H. Li, "Pec: Proactive Elastic Collaborative Resource Scheduling in Data Stream Processing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 7, pp. 1628–1642, 2019.

[4] H. Robbins and S. Monro, "A Stochastic Approximation Method," *The Annals of Mathematical Statistics*, pp. 400–407, 1951.

[5] M. D. Zeiler, "Adadelta: An Adaptive Learning Rate Method," 2012, *arXiv:1212.5701*. [Online]. Available: http://arxiv.org/abs/1212.5701

[6] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the Importance of Initialization and Momentum in Deep Learning," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2013, pp. 1139–1147.

[7] A. Michlmayr, F. Rosenberg, P. Leitner, and S. Dustdar, "End-to-End Support for QoS-Aware Service Selection, Binding, and Mediation in VRESCo," *IEEE Trans. Serv. Comput.*, vol. 3, no. 3, pp. 193–205, 2010.

[8] Z. Wang, M. M. Hayat, N. Ghani, and K. B. Shaban, "Optimizing Cloud-Service Performance: Efficient Resource Provisioning via Optimal Workload Allocation," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 6, pp. 1689–1702, 2017.

[9] H. Yu, J. Yang, and C. Fung, "Fine-Grained Cloud Resource Provisioning for Virtual Network Function," *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 3, pp. 1363–1376, 2020.

[10] Z. Xiao, W. Song, and Q. Chen, "Dynamic Resource Allocation Using Virtual Machines for Cloud Computing Environment," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 6, pp. 1107–1117, 2013.

[11] Y. Kwon *et al.*, "Mantis: Efficient Predictions of Execution Time, Energy Usage, Memory Usage and Network Usage on Smart Mobile Devices," *IEEE Trans. Mobile Comput.*, vol. 14, no. 10, pp. 2059–2072, 2015.

[12] S. Kianpisheh, M. Kargahi, and N. M. Charkari, "Resource Availability Prediction in Distributed Systems: An Approach for Modeling Non-Stationary Transition Probabilities," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 8, pp. 2357–2372, 2017.

[13] Q. Zhang, L. T. Yang, Z. Yan, Z. Chen, and P. Li, "An Efficient Deep Learning Model to Predict Cloud Workload for Industry Informatics," *IEEE Trans. Ind. Informat.*, vol. 14, no. 7, pp. 3170–3178, 2018.

[14] J. Zhu, P. He, Z. Zheng, and M. R. Lyu, "Online QoS Prediction for Runtime Service Adaptation via Adaptive Matrix Factorization," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 10, pp. 2911–2924, 2017.

[15] Z. Chen, J. Hu, G. Min, A. Y. Zomaya, and T. El-Ghazawi, "Towards Accurate Prediction for High-Dimensional and Highly-Variable Cloud Workloads with Deep Learning," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 4, pp. 923–934, 2020.

[16] W. Zhang, B. Li, D. Zhao, F. Gong, and Q. Lu, "Workload Prediction for Cloud Cluster Using a Recurrent Neural Network," in *Proc. IEEE Int. Conf. Identification Inf. Knowl. Internet Things (IIKI)*, 2016, pp. 104–109.

[17] H. Tang, D. Zhou, and D. Chen, "Dynamic Network Function Instance Scaling Based on Traffic Forecasting and VNF Placement in Operator Data Centers," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 3, pp. 530–543, 2019.

[18] A. K. Singh, D. Saxena, J. Kumar, and V. Gupta, "A Quantum Approach Towards the Adaptive Prediction of Cloud Workloads," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 12, pp. 2893–2905, 2021.

[19] L. Nie, X. Wang, S. Wang, Z. Ning, M. S. Obaidat, B. Sadoun, and S. Li, "Network Traffic Prediction in Industrial Internet of Things Backbone Networks: A Multitask Learning Mechanism," *IEEE Trans. Inf. Theory*, vol. 17, no. 10, pp. 7123–7132, 2021.

[20] "Riverbed Virtual SteelHead Series," https://www.wansolutionworks.com/Virtual-SteelHead.asp, accessed November 12, 2021.

[21] "Barracuda Web Application Firewall," https://www.barracuda.com/products/webapplicationfirewall/models/2, accessed November 12, 2021.

[22] "Cisco Virtual Wireless Controller Data Sheet," https://www.cisco.com/c/en/us/products/collateral/wireless/virtual-wireless-controller/data_sheet_c78-714543.html, accessed November 12, 2021.
[23] "Bro," https://www.bro.org/sphinx/cluster/index.html, accessed 2017.
[24] S. Baig, W. Iqbal, J. L. Berral, A. Erradi, and D. Carrera, "Adaptive Prediction Models for Data Center Resources Utilization Estimation," *IEEE Trans. Netw. Service Manag.*, vol. 16, no. 4, pp. 1681–1693, 2019.
[25] I. K. Kim, W. Wang, Y. Qi, and M. Humphrey, "Forecasting Cloud Application Workloads with CloudInsight for Predictive Resource Management," *IEEE Trans. on Cloud Comput.*, pp. 1–1, 2020.
[26] L. Hu, X. L. Che, and S. Q. Zheng, "Online System for Grid Resource Monitoring and Machine Learning-based Prediction," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 1, pp. 134–145, 2012.
[27] F. Xu, H. Zheng, H. Jiang, W. Shao, H. Liu, and Z. Zhou, "Cost-Effective Cloud Server Provisioning for Predictable Performance of Big Data Analytics," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 5, pp. 1036–1051, 2019.
[28] S. Gupta, A. D. Dileep, and T. A. Gonsalves, "Online Sparse BLSTM Models for Resource Usage Prediction in Cloud Datacentres," *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 4, pp. 2335–2349, 2020.
[29] Y. Jiang, H. Feng, F. Zheng, D. Niyato, and X. You, "Deep Learning-Based Edge Caching in Fog Radio Access Networks," *IEEE Trans. Wireless Commun.*, vol. 19, no. 12, pp. 8442–8454, 2020.
[30] H. Shi, M. Xu, and R. Li, "Deep Learning for Household Load Forecasting—A Novel Pooling Deep RNN," *IEEE Trans. Smart Grid*, vol. 9, no. 5, pp. 5271–5280, 2018.
[31] K. Xie *et al.*, "Accurate Recovery of Internet Traffic Data: A Sequential Tensor Completion Approach," *IEEE/ACM Trans. Netw.*, vol. 26, no. 2, pp. 793–806, 2018.
[32] N. Loi, "Predict Traffic of LTE Network," https://www.kaggle.com/naebolo/predict-traffic-of-lte-network/discussion, accessed July 12, 2021.
[33] M. Singh, M. Singh, and S. Kaur, "10 Days DNS Network Traffic," https://data.mendeley.com/datasets/zh3wnddzxy/1, accessed July 12, 2021.

Haojun Huang is an Associate Professor in the School of Electronic Information and Communications at Huazhong University of Science and Technology, China. He received his PhD degree in Communication and Information Engineering from the University of Electronic Science and Technology of China in 2012, and the BS degree in Computer Science from Wuhan University of Technology in 2005. His current research interests include Internet of Things, Network Function Virtualization, Software-Defined Networking, and Artificial Intelligence for networking.

Zhaoxi Li is currently pursuing a Master Degree in Information and Communication Engineering at Huazhong University of Science and Technology, China. She received the BS degree in Networking Engineering from China University of Geosciences, China, in 2020. Her research interests include Network Function Virtualization and Reinforcement Learning.

Jialin Tian is currently a PhD student in Computer Science at the University of Exeter. She received the ME degree in Information and Communication Engineering from Huazhong University of Science and Technology, China in 2022, and the BS degree in Communication Engineering from Northeastern University, China, in 2019. Her research interests include Network Function Virtualization and Artificial Intelligence for networks.

Geyong Min is a Professor of High Performance Computing and Networking in the Department of Computer Science within the College of Engineering, Mathematics and Physical Sciences at the University of Exeter, United Kingdom. He received the PhD degree in Computing Science from the University of Glasgow, United Kingdom, in 2003, and the BS degree in Computer Science from Huazhong University of Science and Technology, China, in 1995. His research interests include Computer Networks, Wireless Communications, Parallel and Distributed Computing, Ubiquitous Computing, Multimedia Systems, Modelling and Performance Engineering.

Wang Miao is currently a Postdoctoral Research Associate in the Department of Computer Science at the University of Exeter, United Kingdom. He received his PhD degree in Computer Science from the University of Exeter, United Kingdom in 2017. His research interests focus on Network Function Virtualization, Software-Defined Networking, Unmanned Aerial Networks, Wireless Communication Networks, Wireless Sensor Networks, and Edge Artifical Intelligence.

Dapeng Oliver Wu is currently a Chair Professor at the Department of Computer Science, City University of Hong Kong. He received the PhD degree in Electrical and Computer Engineering from Carnegie Mellon University, Pittsburgh, PA, in 2003, and BE degree in Electrical Engineering from Huazhong University of Science and Technology, Wuhan, China, in 1990. His research interests are in the areas of Networking, Communications, Signal Processing, Computer Vision, Machine Learning, Smart Grid, and Information and Network Security. He has served as an Editor in Chief of IEEE TNSE, Associate Editor for IEEE TCC, ToC, TWC and TVT, and a Guest-Editor for IEEE JSAC. He is an IEEE Fellow.