Information Sciences 498 (2019) 106–116

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Data-driven dynamic resource scheduling for network slicing: A Deep reinforcement learning approach

Haozhe Wang[a], Yulei Wu[a,*], Geyong Min[a,*], Jie Xu[b], Pengcheng Tang[b]

[a] Department of Computer Science, College of Engineering, Mathematics and Physical Sciences, University of Exeter, UK
[b] NFV Research Department, Huawei Technologies, China

ARTICLE INFO

Article history:
Received 19 March 2019
Revised 25 April 2019
Accepted 6 May 2019
Available online 16 May 2019

Keywords:
Data-driving
End-to-End
Deep reinforcement learning
Network slicing

ABSTRACT

Network slicing is designed to support a variety of emerging applications with diverse performance and flexibility requirements, by dividing the physical network into multiple logical networks. These applications along with a massive number of mobile phones produce large amounts of data, bringing tremendous challenges for network slicing performance. From another perspective, this huge amount of data also offers a new opportunity for the management of network slicing resources. Leveraging the knowledge and insights retrieved from the data, we develop a novel Machine Learning-based scheme for dynamic resource scheduling for networks slicing, aiming to achieve automatic and efficient resource optimisation and End-to-End (E2E) service reliability. However, it is difficult to obtain the user-related data, which is crucial to understand the user behaviour and requests, due to the privacy issue. Therefore, Deep Reinforcement Learning (DRL) is leveraged to extract knowledge from experience by interacting with the network and enable dynamic adjustment of the resources allocated to various slices in order to maximise the resource utilisation while guaranteeing the Quality-of-Service (QoS). The experiment results demonstrate that the proposed resource scheduling scheme can dynamically allocate resources for multiple slices and meet the corresponding QoS requirements.

1. Introduction

With the exponential growth of mobile devices, proliferation of various emerging applications, together with ever-increasing requirements for high Quality-of-Service (QoS), the fifth-generation (5G) communications system is envisioned to meet the unprecedented demands and support a variety of novel services and vertical industries, such as Industry 4.0, automotive communication, Virtual Reality (VR) and Augmented reality (AR), and remote healthcare [3]. The tremendous number of devices, various applications and complex network infrastructure produce a huge amount of data every second. This brings immense challenges for guaranteeing the performance of networks and satisfying the customer experience. Therefore, the management of network resources becomes a timely challenge in 5G networks.

From another perspective, the big data generated in the network offers a new opportunity to understand the behaviour of users, performance of applications and situation of networks [2,12]. Valuable insights and context-rich knowledge can be discovered from the big data, which in turn allows for intelligent and dynamic resource management.

* Corresponding authors.
E-mail addresses: h.wang3@exeter.ac.uk (H. Wang), y.l.wu@exeter.ac.uk (Y. Wu), g.min@exeter.ac.uk (G. Min), jay.xujie@huawei.com (J. Xu), pony.tang@huawei.com (P. Tang).

https://doi.org/10.1016/j.ins.2019.05.012
0020-0255/© 2019 Elsevier Inc. All rights reserved.

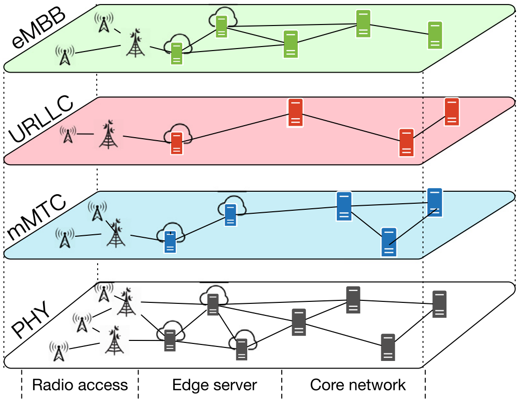


Fig. 1. An illustration of 5G network slicing running on a shared physical infrastructure. Each slice is isolated E2E virtual network and serves a particular application.

To efficiently accommodate a wide range of services in 5G and address the diverse QoS requirements, the concept of network slicing [27] has been proposed as a key enabler to realising the above vision and achieving flexibility and scalability in the management of network resources. A network slice is an isolated End-to-End (E2E) virtualised network across all the network domains running on a shared physical infrastructure and can be controlled and managed independently, as shown in Fig. 1. The realisation of network slices is through the combination of the emerging technologies, i.e., Software Defined Networking (SDN) and Network Function Virtualisation (NFV) [18]. Each slice is a collection of network resources in the form of multiple Virtual Network Functions (VNFs) that are network capabilities implemented as software instances running on commodity servers. Network slicing facilitates efficient utilisation of network resources to meet diverse service requirements by providing only the necessary resources on demand.

Resource scheduling and allocation are the fundamental functions of network slicing in 5G networks [1]. They are formulated as different NP-hard problems [7,26,34,36] and solved by various heuristic methods. Since most resource allocation problems are not convex, the optimal solutions are very difficult to obtain. The heuristic methods also allocate resources in a static manner, in which a fixed amount of resource is reserved for the slice. This would lead to the resource under utilisation and face great challenges to meet the dynamic QoS requirements of diverse mobile services. In our previous work [19], we developed an analytical model to evaluate the performance of NFV service chains. The accuracy of the model relies on the factual assumption of user requests. The user data is crucial to understand users' behaviour and requests. However, this kind of data always related to personal privacy and very difficult to collect. In this work, we design the method to schedule the slicing resource in an asymmetric information scenario, where the user-related data is not used, therefore, the user privacy is preserved.

The machine learning methods, especially deep learning, have gained popularity in a large variety of research areas [35]. The recent successful applications of machine learning in some challenging decision-making domains [6,9–11,17,21,22,31,37,38] imply that machine learning-based methods can be used for solving the NP-hard resource scheduling problems. In particular, Reinforcement Learning (RL), a branch of machine learning has gained tremendous attractions recently. In RL, an agent learns to select an optimal action directly from experience gained from interacting with the environment. The agent begins with knowing nothing about the task at hand and learns by reward signal that evaluates how the chosen action is doing on the task. Combining the use of deep learning with RL generates the concept of Deep Reinforcement Learning (DRL) [4], which enables RL to scale to the decision-making problems that were previously intractable, i.e., the settings with high-dimensional state and action spaces. Inspired by the recent advances of DRL in video games [21], Go [31], robotics [30] and cyber-security [16], we propose to design the 5G network slicing scheduler that learns to manage and allocate resources directly from experience, providing a viable alternative to human-generated heuristics for resource management and scheduling in 5G network slicing.

In this paper, a novel DRL-based scheduling scheme is designed to achieve the dynamic resource management for 5G network slicing. Instead of allocating a static amount of resource to a slice, the designed scheme learns to dynamically manage the resources of a slice depending on the perceived demands. The main contributions of this paper can be stated as follows:

- An E2E system model is designed to describe the service-specific performance and requirements of each slice, therefore the slicing resource scheduling treats all the slices in the network as a whole instead of chained individual VNFs. With the ensemble of slices, the scheduling decisions can achieve fairness and E2E performance at once.
- A novel model-free resource scheduling scheme is proposed for 5G network slicing, in which the user-related privacy data is not used for the decision making. Deep Convolutional Neural Network (CNN) is adopted to model the complex 5G network environment and solve the optimisation challenge caused by the lack of a precise model.

Table 1
Frequently used notations.

Parameter	Meaning
G_P	Substrate 5G network
N_P	Node set in the substrate network
N	Number of substrate node is the network
L_P	Link set in the substrate network
N_s	A substrate node providing computing resources
L_s	A link path providing network resources
$Res_k(N_d)$	Capacity of resource k on node N_s
S	Set of all running slices in the network
s_m	Service slice m
$v_{m,\ d}$	Requested resource of the VNF belonging to slice s_m on node N_d
$ob(s_m)$	Current observation of slice m state
C_m	Percentage of resource allocation to slice m
U_m	Percentage of usage of slice m to the allocated resource
I_m	Weighted percentage of allocated resource of s_m to the node resource
P_m	Percentage of available resources of substrate node
A	Action space set
a_c	Action chosen by the agent to adjust resource by a_c%
x	The control granularity for resource scheduling
z	Limit of resource adjustment in percentage
$P(s_m)$	Performance penalty of slice m
r_t	Reward returned at time t
$vio(s_m)$	Number of VNFs in s_m not allocated enough resources
$al_{m,d}$	Allocated resources to slice m on Node N_d
$u_{m,d}$	Resource usage of slice m at node N_d

- The developed scheme adopts the DRL framework for automatically decision-making to optimise the resource allocation to each slice. This is achieved by constantly learning through the observed current resource usage of network and new service requests.
- The features of substrate network are represented as a binary matrix to facilitate the DRL agent to discover the implicit relations between multiple slices; a practical action space is designed to solve the exponential order of actions in slicing scheduling.

The remainder of the paper is organised as follows. Section 2 provides the most relevant related work. Section 3 is devoted to the design of the system model. Section 4 proposes the resource scheduling scheme for 5G network slicing. The effectiveness of the proposed scheme is evaluated by the simulated experiments in Section 5. Finally, Section 6 concludes the paper.

2. Related work

As a key technology for 5G, network slicing has been gaining tremendous attention from both academia and industry [5,8,13,18,23,24,27,32,34]. Most of the existing studies focus on the resource allocation during the slicing deployment phase, which is similar to a Virtual Network Embedding (VNE) problem. The slicing placement has been formulated as an Integer Linear Program (ILP) optimisation problem extended from VNE in [34]. A game-based method was proposed in [5] to deal with the resource sharing among multiple slices as a Fisher market problem. The virtual machine placement problem was formulated as two ILP problems with two goals to minimise the cost and maximise the quality of experience for Content Delivery Network slicing in [23]. The authors in [13] proposed a two-tier priority based admission control strategy for multiple service slices and solved the problem using heuristics method. However, in practice, 5G networks are inherently dynamic [27]. The utilisation of services and network traffic flowing through slices fluctuate over time. As a result, the network provider needs to continuously optimise the slice resource allocation to meet the time-varying slicing requirements. There are a limited number of studies dealing with the dynamic resource scheduling for slicing. A dynamic resource algorithm has been proposed in [20], leveraging Q-learning to optimise the resource allocation to an individual node in VNE. However, the resource allocation in slicing is different from VNE. Dynamic resource scheduling in 5G presents additional challenges as we have to deal with interdependent VNFs with predefined orders and different resource requirements and isolated slices with various QoS requirements. Therefore, it is critical to design a dynamic /resource scheduling scheme for the distinct QoS requirements of different network slice services to effectively optimise service performance and resource utilisation.

3. The system model

In this section, we present our system model and formulate the resource scheduling problem for network slicing. The model consists of two main components: substrate network and service slice. Table 1 summarises the notations used in this paper.

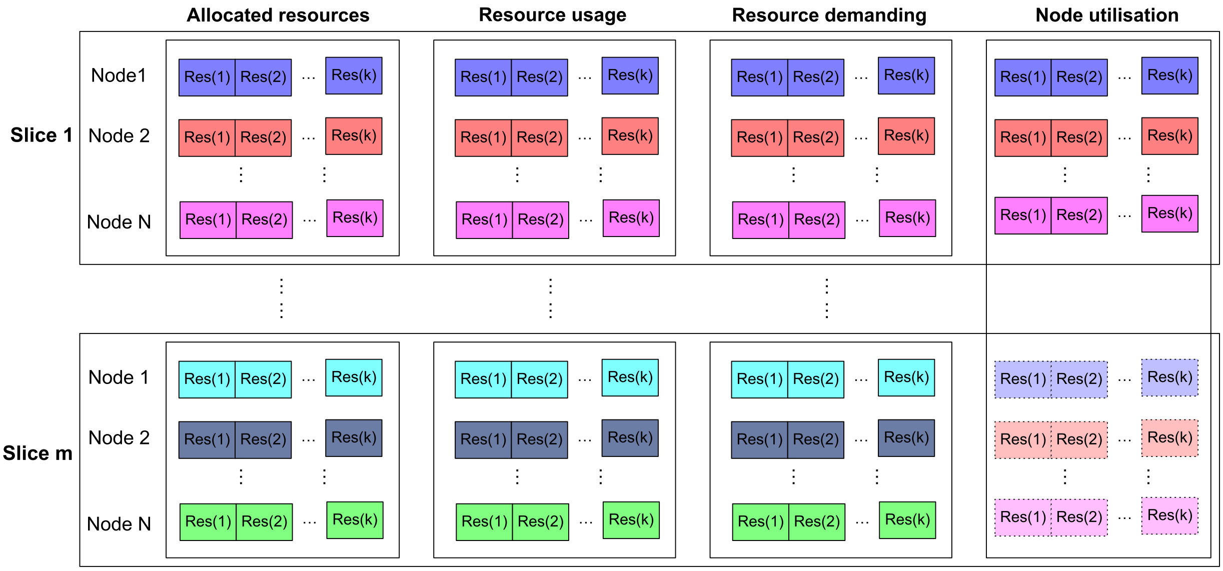


Fig. 2. The E2E slice profiles in a 5G network, with N substrate nodes and m slices in the network.

3.1. Network model

The substrate network refers to a general 5G network infrastructure that provides all kinds of resources. The substrate network is represented as an undirected graph and denoted as $G_P = (N_P, L_P)$, where N_P is the set of physical nodes built on common servers and L_P indicates the set of all substrate link connections. Each substrate node $N_s \in N_P$ is associated with k different types of resources (e.g., CPU, memory, Storage, GPU). The resources on N_s is modelled as a vector $Res(N_s) = [Res_1(N_s), Res_2(N_s), \ldots, Res_k(N_s)]$. In this paper, we consider each node in N_P has the same number of resource types, and thus the length of $Res(N_s)$ vector is fixed. Each link path in the substrate network $L_s(i, j) \in L_P$ connecting the nodes $N_s(i)$ and $N_s(j)$ is equipped with an amount of bandwidth. The total available bandwidth for link $L_s(i, j)$ is denoted as $L_s^{BW}(i, j)$.

The network slices present diverse services, such as high-definition video streaming, vehicular communication and Internet of Things (IoT), which require different types of resources. The 5G networks will support the SDN and NFV network techniques, in which each node N_s can create multiple Virtual Machines (VMs) to install the according VNFs, and each link L_s is divided into multiple virtual links. The created VMs and virtual links are chained to accommodate different slices. As a result, the slicing resource scheduling problem involves managing the resource allocated for the VMs and the links of slices to ensure the guaranteed QoS and optimal utilisation of networks.

A centralised controller is in charge of scheduling and fulfilling the requirement of slices. The slices serve the time-varying requests from the users. By monitoring the time-varying usage, the controller is able to dynamically allocate the resources to the slices. However, the allocations should be elaborately operated to meet the various requirements of slices. We indicate the set of slices in the substrate network as a set, $S = \{s_1, s_2, \ldots, s_M\}$. Each slice consists of multiple associated VNFs deployed on different physical nodes, and each VNF is characterised by the demands for each type of resources. The dth VNF belonging to slice s_m is represented as $v_{m,d} = [Res_1(m, d), Res_2(m, d), \ldots, Res_k(m, d)]$, where $Res_k(m, d)$ denotes the requested resources of type k for slice s_m at node N_d. The controller handles a queue of requests, each requesting a number of different resources for a time period (e.g. x units of Res_1 and y units of Res_2 for t time steps). The controller manages the resources based on scheduling-frame. At each discrete time step, new incoming slice service requests arrive at the network, and the scheduler in the controller dynamically adjusts the allocated resources.

3.2. End-to-end slicing model

An E2E slicing model is crucial for understanding the holistic network situation. In this work, we design the E2E slicing model to consider the slicing network as a whole, instead of individual VNFs or VMs. Obtaining the whole picture, we can then develop a resource scheduling scheme to achieve resource optimisation for all slices at once, which attains both the fairness resource distribution among slices and improving E2E performance.

The E2E slicing model is denoted by a four-dimensional tensor, as shown in Fig. 2. The first dimension represents the resource requirements of a VNF for each type of resources. The second dimension chains the multiple VNFs that belong to the same slice, forming many resource matrices as $Res(s_m) = [v_{m,1}, v_{m,2}, \ldots, v_{m,N}]$.

The third dimension defines the key performance index of slicing resource management. For each slice, we obtain four resource metrics, including the resource allocation status of the VNFs of slice on the physical nodes, the current resource usage of each VNFs, the scale of the slice and the resource utilisation of the physical node. The obtained metrics of slice s_m is represented as a vector, $ob(s_m) = (C_m, U_m, I_m, P_m)$, where C_m is also a vector with each element denoting the percentage of allocated resource of each VNF belonging to s_m, U_m is a vector of the percentage of resource usage of each VNF of s_m, I_m is a weighted sum of percentage of the resource taken by each VNF to the total resource of substrate node, and P_m is a vector of the percentage of resource utilisation of each physical node that the slice is running on. Since P_m is the same for all the slices running on the same node, we will neglect the index and use P for the simplicity. The purpose of using percentage instead of the real value is to normalise different states and accelerate the training process; a similar idea has been used in [20]. As a result, the E2E model for a single slice s_m is given by:

$$ob(s_m) = (C_m, U_m, I_m, P_m) \tag{1}$$

$$C_m = \left(\frac{al_{m,1}}{v_{m,1}}, \ldots, \frac{al_{m,d}}{v_{m,d}}\right)^T \tag{2}$$

$$U_m = \left(\frac{u_{m,1}}{al_{m,1}}, \ldots, \frac{u_{m,d}}{al_{m,d}}\right)^T \tag{3}$$

$$I_m = \omega_m\Big(\frac{al_{m,1}}{Res(N_1)}, \ldots, \frac{al_{m,d}}{Res(N_N)}^T, \sum_{s_m \in S} \omega_m = 1 \tag{4}$$

$$P_m = \left(\sum_{m}\sum_{Res}\frac{al_{m,1}}{Res(N_1)}, \ldots, \sum_{m}\sum_{Res}\frac{al_{m,d}}{Res(N_d)}\right)^T \tag{5}$$

where $al_{m,d}$ denotes the current allocated resource for the VNF of slice s_m at node N_d, and $u_{m,d}$ denotes the VNF resource utilisation rate of slice s_m on node N_d.

In the fourth dimension, we concatenate all the slice state vector to produce the E2E model of the whole network as $\mathbb{O}$ where each element indicates one slice:

$$\mathbb{O} = [ob(s_1), ob(s_2), \ldots, ob(s_M)]^T \tag{6}$$

In practice, each physical node would have a different number of VNFs. However, it is appropriate to maintain a fixed state representation so that it can be directly applied as input to the agent. Therefore, we set a maximum number M of VNFs that can be running on one physical node, and a maximum number of nodes that a slice can be distributed on as $N = |N_p|$. If the current node has fewer VNFs than the maximum number, or a slice does not deploy on certain nodes, we set the corresponding position value to 0.

Each physical node that may contain many servers. In this paper we treat it as a single collection of resources, ignoring the machine fragmentation effects. We use various colours to illustrate different slices. Slices are deployed on different nodes, so the nodes are shared by different set of slices.

4. A Dynamic 5G network slicing resource scheduling scheme

This section describes the design details of the proposed resource scheduling scheme, which generates resource allocation decisions using DRL-based dynamic resource scheduling policies. We start by describing the learning framework and policy network, and then describe the design of the training algorithm.

4.1. Learning algorithm

The learning-based slicing scheduling scheme adopts the DRL for dynamically adjusting the resource allocation. The state representation, action space and reward function are designed as follows.

4.1.1. State representation

The way to understanding and observing the slicing network system is crucial for the DRL agent to establish a thorough knowledge of the network status and generate efficient scheduling decisions. To this end, we firstly design the state representation that serves as the input to the scheme. The states of the current time step slicing system observed by the agent are represented as images based on the E2E model developed in Section 3.2

The resource status reveals the current resource allocation and usage of each resource that have been scheduled for the slices. However, the values of the high-dimensional tensor generated from the E2E model are decimal percentages. Directly feed these values to a deep neural network may result in a lousy accuracy. To transfer the state tensor into an image that can be fed into the deep neural network in the learning agent, we encode each percentage into a binary string. To this end, we define the length of the binary string related to the control precision. The control precision indicates the controlling granularity which is the unit of resource that can be adjusted for a VNF in the slice. In realistic slicing network, the control precision can be in the unit of CPU cores, a certain amount of memory size or even a VM. For slices support different applications, the control precision can be different according to the service types and specific requirements, which means the length of binary string may be different.

In the proposed scheme, we restrict the new resource requirements to be ranged from 0 to 200% of the currently allocated resource. When the percentage is more than 100% (i.e., $al_{m,d} < u_{m,d}$), it indicates that the allocated resource is less than the actual usage, and this is the only case that a percentage is larger than 100%. If the required resource becomes 0, the VNF will be removed from the slice; while if the new required resource is doubled, in order not to affect the other VNFs on the same node, the VNF will be migrated to another substrate node. For example, if the level of control precision is 10%, we will have 20 different states ranging of $[0, 10\%, \ldots 200\%]$, which require 5 bits to represent. In this case, for one type of resource in a slice, the state matrix will be N rows and 4×5 columns. Therefore, if there are k types of resources, the observed state matrix of slice s_m will have $|ob(s_m)| = 4 \times 5 \times k$ columns, and the state image of the substrate network for each time step will be a $[M \times N,\ 20 \times k]$ binary matrix, where M is the number of slices.

4.1.2. Action space

The output actions generated from the agent and executed by the scheduler indicate the adjustment of allocated resources of slices. At each point of time step, the scheduler may decide to increase or decrease the resources of any VNF. However, the adjustment amount of resources can be any integer value, and if we change the allocated resources of multiple slices at the same time, this could lead to a significant huge action space making the learning impossible. For example, for a network of m slices with n VNFs, even if the number of possible actions is only 10, the action space will be 10^{m*n} which increases exponentially with the network size.

To make the action space practical, the action space in our scheme is designed to have the same level of granularity as the state, and each action only affects one VNF at a time. To restrict the behaviour of the agent, the limit of resource adjustment is set between $-z$ and $+z$ to avoid significant changes, where $0 < z < 1$ is a proportion of the currently allocated resource, and a negative value means scale-down and a positive value means scale-up. In one time step, the agent keeps taking actions until the resources of all the VNFs of each slice is updated. If the granularity in the state observation is x, the total number of actions is $2z/x$, and the action space A can be given by

$$A = \left\{ a_c | a_c = z - 2z \cdot c \cdot x, c = 0, 1, \ldots, \frac{2z}{x} \right\} \tag{7}$$

where action a_c means that the agent chooses to increase the allocated resource of the current VNF by a_c%; and negative value denotes the decrease in allocated resources. The case $a_c = 0$ indicates that the agent does not change the scheduled resources in the current time step. When the resources of all the VNFs in the network have been scheduled, the time step moves to the next and new requests will be received from the controller. In this way, the agent can schedule multiple VNFs of different slices in a discrete manner at one time step while keeping the action space linear with $1/x$. The process is summarised in Algorithm 1.

Algorithm 1: Generation of action sets.

```
Input  : adjustment limit, control granularity $x$
Output: action set $A$
Initialise the adjustment range $r_a = 2 \cdot z$;
Calculation number of actions $r_a/x$;
for each action $a_c$ in $A$ do
    Set action indicting number $c$ to each action ;
    /* Generating resource adjustment value for $a_c$                */
    $a_c = z - 2z \cdot c \cdot x$ ;
end
```

4.1.3. Reward

Unlike supervised learning, in which each sample has a corresponding label indicating the preferred output of the learning model, the agent in RL relies on reward signal to evaluate the effectiveness of actions and further improve the policy. Therefore, the design of reward function is crucial in the training process of RL. In slicing resource scheduling problem, the overall objectives are to minimise the SLA violations of slices, guarantee the performance, while maximising the resource utilisation of physical nodes. Specifically, the reward function is crafted to relate with the performance of slices and the resource utilisation of the substrate network.

The performance of one slice depends on each VNF of the slice deploying on different nodes. This is different from traditional virtual network embedding, where the virtual nodes have no order requirement and are independent of each other. However, in network slicing, traffic has to flow through predefined ordered VNFs, and the performance degradation of any VNF would affect the overall slicing performance. To take this into consideration, the performance reward P of a slice s_m is defined to combine two parts: SLA violations and wasting of resource, and is given by

$$P(s_m) = \begin{cases} \sum_{d\in N_P} \eta \cdot e^{vio(s_m)}, & \text{if } u_{m,d} > al_{m,d} \\ \sum_{d\in N_P} \zeta \cdot \frac{al_{m,d}}{u_{m,d}}, & \text{otherwise} \end{cases} \tag{8}$$

Here, the first case indicates that the allocated resource to a VNF is not enough to satisfy the SLA requirement, and η is the penalty parameter for the SLA violation. $vio(s_m)$ denotes the number of SLA violation in one slice. For each type of resources allocated to a VNF, if the allocated resource is less than the required amount, it counts as one violation. Then $vio(s_m)$ is the sum of the violations on every VNF belonging to slice s_m. The exponent is used to model the impact of multiple VNFs with insufficient resource on the slice, since all the VNFs are correlated in one slice; and the second case indicates the inefficient utilisation of resources, which could lead to low revenue and energy waste, and ζ is the resource wasting penalty. The penalty constants η, ζ imply the weights of the optimal objectives, and can be adjusted for different services. By combining the resource utilisation of the substrate network with performance, the reward function is expressed by:

$$r_t = \sum_{s_m\in S} P(s_m) + \varphi \sum_{N_s\in N_P} \sum_{Res} \frac{al_{m,N_s}}{\sum_{Res} N_s} \tag{9}$$

where φ is the utilisation penalty constant and t denotes the current time step. Similar to Mnih et al. [22], the agent does not use any instant reward for the learning, but aim to maximise the cumulative discounted reward, since the reward getting from the adjustment of the allocated resource of one VNF does not represent the whole slice.

4.2. Policy network

The agent relies on the policy network to select actions for resource scheduling. The policy network takes the state image as input and outputs a probability distribution over action set. We construct a multi-layer CNNs [15] to represent the policy network in our design. CNNs have been used successfully to solve many problems in visual recognition and computer vision, and have been proven as an efficient tool in extracting features from images without hand craft efforts [14]. The built policy network is a sequence of layers. It consists of four convolutional layers, with two MaxPooling2D layers following the second and fourth convolutional layer. The first two convolutional layers are with 32 kernels. After the first pooling layer, the number of kernels are doubled. Then, the output of the second pooling layer is flattened to one dimensional vector and passed through a fully connected layer. ReLU activations are used for all layers except the output layer. The output layer uses a SoftMax activation for the purpose of probabilistic actions. At each time step, the scheduler uses the policy network to decide how to adjust the resources allocated to the slices.

4.3. Training algorithm

We train the policy network and optimise its performance in an episodic manner. In each episode, the agent will receive a sequence of resource requests and adjust the resources allocated to each VNF of every slice using the policy network. In each training iteration, we conduct multiple episodes for the resource requests sequence to explore the probabilistic space of possible actions using the current policy. The (state, action, reward) tuple is recorded at each time step to calculate the discounted cumulative reward. Then a baseline method [29] is used to construct the baseline b_t utilising the average of the returned discounted reward $\sum_{t=0}^{\infty} \gamma^t r_t$ across all episodes. Then policy gradient is leveraged to train the built network using the constructed baseline. The policy gradient can be expressed as

$$g = \mathbb{E}\left[\sum_{t=0}^{L_e} \left(\sum_{t'=t} \gamma^{t'-t} r_{t'} - b_t \right) \nabla_\theta log\pi_\theta(a_t|s_t) \right] \tag{10}$$

where L_e is the length of the episode, and $\pi_\theta(a_t|s_t)$ represents the current policy with parameters θ.

For updating the policy network parameters, we employ RMSProp [28], which is a gradient-based optimisation technique. The updating process can be derived as:

$$\begin{aligned} E[g^2]_t &= \rho E[g^2]_{t-1} + (1-\rho) g_t^2 \\ \theta_{t+1} &= \theta_t - \frac{lr}{\sqrt{E[g^2]_t + \epsilon}} g_t \end{aligned} \tag{11}$$

where g_t is the gradient at the current step, $E[g_t^2]$ is the averaged accumulated past squared gradients, ρ is the decaying factor that indicates how important is the new gradient, θ is the parameters, and ϵ is the smoothing constant to avoid dividing zero. Due to the complexity of the network, the resulting gradients may yield a high variance. The key idea of using RMSProp is to focusing on updating the learning rate lr based on the gradients for each step. The learning rate lr is divided by the Root Mean Squared (RMS) error, the root of an exponentially decaying average of squared gradients $\sqrt{E[g^2]_t + \epsilon}$. As a result, the updating method normalises the gradient to achieve adaptive learning rates. It decreases the rate for large gradient to avoid exploding, and increases the rate for small gradient to avoid vanishing.

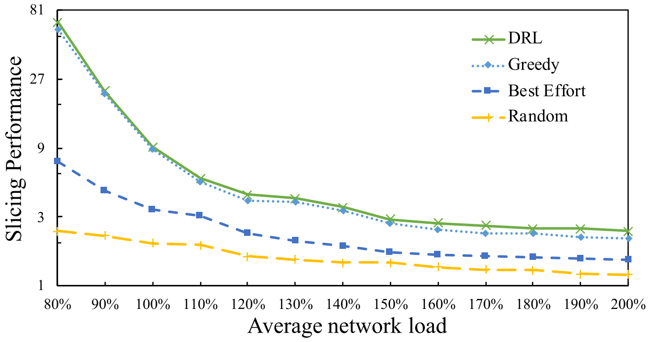


Fig. 3. Slicing performance under different levels of network load.

5. Performance evaluation

In this section, we present the numerical simulations to demonstrate the effectiveness of the proposed dynamic slicing resource scheduling scheme.

5.1. Experiment setup

To evaluate our method, a random topology with five physical nodes is generated. We assign three types of resource, naming CPU, storage and bandwidth capacities to each physical node with 50 resource units respectively. In each episode, 5 network slices with a total of 25 VNFs are deployed into the network using a random allocation. The requested resource for each VNF in a slice follows a uniform distribution between 1 and 20 resource units during each time-step of the service. Then 5 sequences of time-varying resource requests are generated and assigned to each slice. The new resource demands are uniformly chosen between increasing or decreasing between −0.5 to 0.5 of the currently allocated resources. The duration of the slice follows an exponential distribution with an average of 150 time steps. The training network of the scheme was built as CNN with 9 layers and a total of 382441 parameters. Theano [33] is employed as the backend to build the policy network. The parameters of the network are initiated following a uniform distribution.

In each training iteration, we run 300 epoch simulations to calculate the cumulated rewards, and then update the network. The discount factor is set as $\gamma = 0.995$. The policy network parameters are updated using the RMSProp algorithm with a learning rate of $\gamma = 0.0001$ and $\rho = 0.9$. Unless otherwise specified, the results below are from training for 500 training iterations. The proposed method is compared against three agents, a greedy agent, a static agent and a random agent respectively under four service load pressure. The *greedy* agent is implemented following by the method in [25]. It allocates resources as much as possible to satisfy the requested amount, increasing or decreasing the resources to the amount that is nearest to the demands for each slice. Therefore, the greedy agent focuses on optimising the resource utilisation rate of slices. The *best-effort* agent is a semi-static allocating method. If one slice requires more resources, it will try its best to satisfy the requirements or allocate as much resource as can be afforded from the available resources, but once allocated, the agent does not scale down the resources even the resource utilisation is low. For this manner, the best-effort agent focuses on reducing the SLA violation. The *random* agent allocates a random percentage of the demanding resource to each request.

5.2. Performance evaluation

We evaluate the performance and demonstrate the effectiveness of the proposed scheme. The agents are compared and measured in three aspects, the QoS of slices in terms of the performance, the average SLA violation, and the resource utilisation of the substrate network under different levels of network loading. The results are shown in Figs. 3–5.

When the required resource by all the slices are less than the resource provided by the substrate network, the scheduling decision making is easy for all methods, since they don't have to make balanced allocation between all the slices to ensure the E2E performance. Therefore, we choose high network loads scenarios that require the scheduling scheme to obtain a whole image of the network and make intelligent decisions to achieve both high E2E performance and low SLA violations.

The performance measurements of slices are shown in Fig. 3. To get a complete evaluation of the proposed scheme, we have carried out more experiments to show the performance of slices under light network loads between 80% to 100%, medium loads between 110% and 150% and high loads that are in the range of 160% to 200%. The E2E performance of slice is calculated by collectively considered the performance of all VNFs belongs to the same slice. Therefore, the E2E performance is determined by each VNF in that slice, which is measured by the allocated resources and the utilised resource of that VNF. The number of unmet VNF is counted and smoothed by a sigmoid function, and the performance for each slice is given by

$$\mathrm{Perform}(s_m) = \sum_{d \in N_p} \frac{\sum_{Res} N_d}{\mathrm{sigmoid}(e^{vio(s_m)})} \tag{12}$$

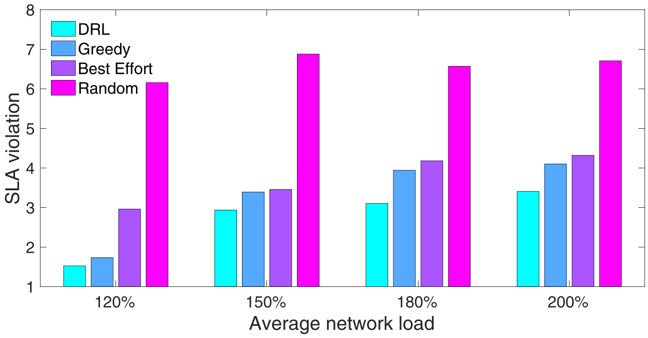


Fig. 4. Average SLA violation of four agents under different levels of network load.

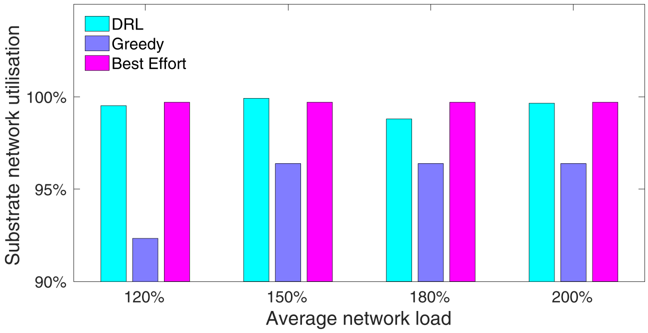


Fig. 5. Average substrate nodes resource utilisation under different levels of network load.

and then the overall slicing performance is calculated as the summary of each slice $\sum_{s_m \in S} \text{Perform}(s_m)$. As shown in the figure, the DRL agent performs better than the other agents under various levels of network loads. This can be attributed to the fact that the DRL agent leverages the E2E slicing model to discover the correlations between different slices and capture the holistic picture of the resource allocation and utilisation of the whole network, and therefore can achieve a globally optimal allocation rather than trying to satisfy all the demands of a single slice.

Fig. 4 depicts the comparison of average SLA violation. Similar to the slicing performance, the SLA of a slice depends on the allocated and used resources. But different from performance, SLA is used to measure the degree of how the resource demands of a slice are fulfilled. As a result, the SLA violation is derived as:

$$\text{Violation} = \frac{1}{|S|} \sum_{s_m \in S} \sum_{d \in N_P} vio(s_m) * \left(1 - \frac{al_{m,d}}{u_{m,d}}\right) \tag{13}$$

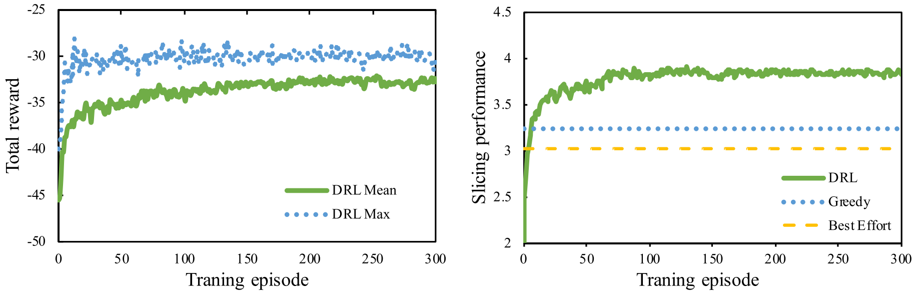
The result shows that the proposed DRL-based dynamic scheduling scheme has the smallest SLA violation among all the agents. The DRL agent balances the resource distribution not only between the VNFs in the same slice but also between the slices. As a result, the number of times of violation occurring on the VNF is less in the same slice, leading to a lower violation.

Fig. 5 illustrates the comparison of average resource utilisation of the agents. The utilisation of one substrate node is calculated as the total allocated resources to the VNFs of various slices of the whole capacity. The average utilisation of substrate nodes can be expressed as

$$\text{Utilisation} = \frac{1}{N} \sum_{d \in N_p} \frac{\sum_{s_m \in S} al_{m,d}}{\sum_{Res} N_d} \tag{14}$$

The random is not shown in the figure since with random actions, the average utilisation is always around 75%. From the figure, we can see that our scheme outperforms greedy and achieves a similar utilisation as static. Since the static agent aims to maximise the utilisation only, the results validate that our method can achieve high efficiency in resource utilisation while guaranteeing slicing performance.

Finally, we investigate the convergence behaviour of the proposed scheme. To understand the convergence of the scheme, we look deeper into one specific scenario, in which the scheme is trained to optimise the resource allocation under service load of 150% in Fig. 3. Fig. 6a plots the learning curve of total reward improved as training. The DRL Max is the maximum reward across all the epochs at each iteration. The DRL Mean is the averaged reward. As the training goes, both values increase due to the improvement of the scheduling scheme. The gap between max and mean values narrows, which shows



(a) Total reward by episode (b) Slicing Performance by episode

Fig. 6. The learning curve illustrating the training improves the total reward and slicing performance during the training.

the convergence of the model. Fig. 6b depicts the scheduled averaged slicing performance under the policy at each episode. As expected, we see that the proposed scheme improves the performance by each iteration. At the beginning, the policy is no better than neither best-effort or greedy allocation methods, but after 20 iterations the DRL is better than both methods. The results show a good convergence speed of the proposed scheme.

6. Conclusions

In this paper, we have solved the key challenges in designing the dynamic resource scheduling method for network slicing, and have proposed a data-driven resource scheduling scheme. The proposed E2E slicing model has treated the VNFs in one slice as a whole and have achieved a global optimisation. The developed DRL-based scheduling scheme has solved the slicing resource management challenge in an asymmetric information scenario without using the user-related data, due to the model-free and dynamic online learning features. CNN has been used to improve the feature extraction of the DRL to consider the network resources and slice requirements at the same time. The experiment results have shown that the proposed scheduler outperforms the heuristic, best-effort and random approaches and can improve the resource utilisation of the physical network while guaranteeing the QoS.

Conflicts of Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work was supported by Huawei Technologies under the HIRP project (Contract No.: HO2017050001C6).

References

[1] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, H. Flinck, Network slicing and softwarization: a survey on principles, enabling technologies, and solutions, IEEE Commun. Surv. Tutorials 20 (2018) 2429–2453.
[2] E. Ahmed, I. Yaqoob, I.A.T. Hashem, J. Shuja, M. Imran, N. Guizani, S.T. Bakhsh, Recent advances and challenges in mobile big data, IEEE Commun. Mag. 56 (2018) 102–108.
[3] N. Alliance, Next Generation Mobile Networks: 5g White Paper, 2015, Technical Report.
[4] K. Arulkumaran, M.P. Deisenroth, M. Brundage, A.A. Bharath, Deep reinforcement learning: a brief survey, IEEE Signal Process. Mag. 34 (2017) 26–38.
[5] P. Caballero, P. Caballero, A. Banchs, A. Banchs, G. De Veciana, G. De Veciana, X. Costa-Perez, X. Costa-Perez, Network slicing games: enabling customization in multi-tenant networks, in: IEEE INFOCOM 2017 - IEEE Conference on Computer Communications, IEEE, 2017, pp. 1–9.
[6] Z. Chen, Q. Yan, H. Han, S. Wang, L. Peng, L. Wang, B. Yang, Machine learning based mobile malware detection using highly imbalanced network traffic, Inf. Sci. 433–434 (2018) 346–364.
[7] X. Cheng, Y. Wu, G. Min, A.Y. Zomaya, Network function virtualization in dynamic networks: a stochastic perspective, IEEE J. Select. Areas Commun. (2018). Early Access.
[8] X. Foukas, G. Patounas, A. Elmokashfi, M.K. Marina, Network slicing in 5g: survey and challenges, IEEE Commun. Mag. 55 (2017) 94–100.
[9] K. Gai, M. Qiu, Optimal resource allocation using reinforcement learning for IoT content-centric services, Appl Soft Comput 70 (2018) 12–21, doi:10.1016/j.asoc.2018.03.056. http://www.sciencedirect.com/science/article/pii/S1568494618302540.
[10] K. Gai, M. Qiu, Reinforcement learning-based content-centric services in mobile sensing, IEEE Netw. 32 (2018) 34–39.
[11] K. Gai, M. Qiu, M. Liu, H. Zhao, Smart resource allocation using reinforcement learning in content-centric cyber-physical systems, in: M. Qiu (Ed.), Smart Computing and Communication, Springer International Publishing, Cham, 2018, pp. 39–52.
[12] H. Huang, H. Yin, G. Min, H. Jiang, J. Zhang, Y. Wu, Data-driven information plane in software-defined networking, IEEE Commun. Mag. 55 (2017) 218–224.
[13] M. Jiang, M. Condoluci, T. Mahmoodi, Network Slicing Management & Prioritization in 5G Mobile Systems, in: European Wireless, 2016, pp. 197–202.
[14] A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet classification with deep convolutional neural networks, Commun. ACM 60 (2017) 84–90.
[15] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (1998) 2278–2324.
[16] C. Li, M. Qiu, Reinforcement Learning for Cyber-Physical Systems: with Cybersecurity Case Studies, CRC Press, 2019.

[17] T.P. Lillicrap, J.J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, Continuous control with deep reinforcement learning, 2015, (pp. 1–14). arXiv:1509.02971.

[18] D. Lopez, J. Ordonez-Lucena, P. Ameigeiras, J.J. Ramos-Munoz, J. Lorca, J. Folgueira, Network slicing for 5g with SDN/NFV: concepts, architectures, and challenges, IEEE Commun. Mag. 55 (2017) 80–87.

[19] W. Miao, G. Min, Y. Wu, H. Huang, Z. Zhao, H. Wang, C. Luo, Stochastic performance analysis of network function virtualization in future internet, IEEE J. Select. Areas Commun. 37 (2019) 613–626.

[20] R. Mijumbi, J.L. Gorricho, J. Serrat, M. Claeys, F. De Turck, S. Latre, Design and Evaluation of Learning Algorithms for Dynamic Resource Management in Virtual Networks, in: NOMS 2014 - 2014 IEEE/IFIP Network Operations and Management Symposium, IEEE, 2014, pp. 1–9.

[21] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M. Riedmiller, Playing atari with deep reinforcement learning, 2013, arXiv:1312.5602.

[22] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, D. Hassabis, Human-level control through deep reinforcement learning, Nature 518 (2015) 529–533.

[23] S. Retal, M. Bagaa, T. Taleb, H. Flinck, Content delivery network slicing: qoe and cost awareness, in: ICC 2017 - 2017 IEEE International Conference on Communications, IEEE, 2017, pp. 1–6.

[24] M. Richart, J. Serrat, J. Baliosian, J.L. Gorricho, Resource slicing in virtual wireless networks: a survey, IEEE Trans. Netw. Serv. Manage. 13 (2016) 462–476.

[25] R. Riggio, T. Rasheed, R. Narayanan, Virtual network functions orchestration in enterprise WLANs, in: 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), IEEE, 2015, pp. 1220–1225.

[26] M. Rost, S. Schmid, Service chain and virtual network embeddings: approximations using randomized rounding, 2016. arXiv:1604.02180.

[27] P. Rost, C. Mannweiler, D.S. Michalopoulos, C. Sartori, V. Sciancalepore, N. Sastry, O. Holland, S. Tayade, B. Han, B. Han, D. Bega, D. Aziz, H. Bakker, Network slicing to enable scalability and flexibility in 5g mobile networks, IEEE Commun. Mag. 55 (2017) 72–79.

[28] S. Ruder, An overview of gradient descent optimization algorithms, 2016. arXiv:1609.04747.

[29] J. Schulman, S. Levine, P. Moritz, M.I. Jordan, P. Abbeel, Trust region policy optimization, 2015. arXiv:1502.05477.

[30] H. Shi, Z. Lin, S. Zhang, X. Li, K.S. Hwang, An adaptive decision-making method with fuzzy bayesian reinforcement learning for robot soccer, Inf. Sci. 436–437 (2018) 268–281.

[31] D. Silver, A. Huang, C.J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, D. Hassabis, Mastering the game of go with deep neural networks and tree search, Nature 529 (2016) 484–489.

[32] T. Taleb, A. Nakao, P. Du, Y. Kiriha, F. Granelli, A.A. Gebremariam, M. Bagaa, End-to-end network slicing for 5g mobile networks, J. Inf. Process. 25 (2017) 153–163.

[33] Theano Development Team, Theano: a Python framework for fast computation of mathematical expressions, 2016. arXiv:1605.02688.

[34] S. Vassilaras, L. Gkatzikis, N. Liakopoulos, I.N. Stiakogiannakis, M. Qi, L. Shi, L. Liu, M. Debbah, G.S. Paschos, The algorithmic aspects of network slicing, IEEE Commun. Mag. 55 (2017) 112–119.

[35] Y. Wu, F. Hu, G. Min, A.Y. Zomaya, Big Data and Computational Intelligence in Networking, CRC Press, 2017.

[36] Y. Xie, Z. Liu, S. Wang, Y. Wang, Service function chaining resource allocation: a survey, 2016, arXiv:1608.00095.

[37] Z. Yang, K. Merrick, L. Jin, H.A. Abbass, Hierarchical deep reinforcement learning for continuous action control, IEEE Trans. Neural Netw. Learn. Syst. (2018) 1–11.

[38] Y. Zuo, Y. Wu, G. Min, L. Cui, Learning-based network path planning for traffic engineering, Future Gener. Comput. Syst. 92 (2019) 59–67.