**Noname manuscript No.**
(will be inserted by the editor)

# VNE Solution for Network Differentiated QoS and Security Requirements: From the Perspective of Deep Reinforcement Learning

**Chao Wang · Ranbir Singh Batth · Peiying Zhang · Gagangeet Singh Aujla · Youxiang Duan · Lihua Ren**

Received: date / Accepted: date

**Abstract** The rapid development and deployment of network services has brought a series of challenges to researchers. On the one hand, the needs of Internet end users/applications reflect the characteristics of travel alienation, and they pursue different perspectives of service quality. On the other hand, with the explosive growth of information in the era of big data, a lot of private information is stored in the network. End users/applications naturally start to pay attention to network security. In order to solve the requirements of differentiated quality of service (QoS) and security, this paper proposes a virtual network embedding (VNE) algorithm based on deep reinforcement learning (DRL), aiming at the CPU, bandwidth, delay and security attributes of substrate network. DRL agent is trained in the network environment constructed by the above attributes. The purpose is to deduce the mapping probability of each substrate node and map the virtual node according to this probability. Finally, the breadth first strategy (BFS) is used to map the virtual links. In the experimental stage, the algorithm based on DRL is compared with other representative algorithms in three aspects: long term average revenue, long term revenue consumption ratio and acceptance rate. The results show that the algorithm proposed in this paper has achieved good experimental results, which proves that the algorithm can be effectively applied to solve the end user/application differentiated QoS and security requirements.

**Keywords** End User/Application · Quality of Service · Security Requirement · Virtual Network Embedding

## 1 Introduction

The rapid development of network technology and the blowout growth of end users/applications have brought greater opportunities and challenges to infrastructure providers (InPs) and service providers (SPs) [1,2]. With the 5G era, many network applications are based on virtual network architecture. In this architecture, the implementation of network functions no longer depends on specific hardware facilities, but on the way of software programming to achieve flexible deployment of virtual functions [3–5]. For example, the realization of intelligent applications, such as the Internet of vehicles (IoV), intelligent medical, military unmanned aerial vehicle (UAV), cannot do without a strong virtual network infrastructure as support [6–8]. It cannot

C. Wang
China University of Petroleum (East China), Qingdao, China.
E-mail: wangch_upc@qq.com

R. S. Batth (✉)
Lovely Professional University, Phagwara, Punjab, India.
E-mail: ranbir.21123@lpu.co.in

P. Zhang (✉)
China University of Petroleum (East China), Qingdao, China.
E-mail: zhangpeiying@upc.edu.cn

G. S. Aujla
Durham University, Durham, United Kingdom.
E-mail: gagi_aujla82@yahoo.com and gagangeet.s.aujla@durham.ac.uk

Y. Duan · L. Ren
China University of Petroleum (East China), Qingdao, China.

Y. Duan
E-mail: yxduan@upc.edu.cn

L. Ren
E-mail: renlh@upc.edu.cn

be ignored that there are a series of challenges in using network virtualization (NV) technology to provide services for these intelligent applications [9, 10].

On the one hand, the QoS is closely related to the needs of network end users/applications. With the growth of network end users/applications and the expansion of network services, the QoS requirements of users/applications show the characteristics of differentiation [11]. For the users of the IoV or UAVs, their primary demand is the real-time and accurate control of the vehicles or UAVs [12]. Driverless cars need to judge the road conditions and make accurate decisions in time to avoid traffic accidents [13, 14]. UAV needs timely command to strike the target accurately. So they need InPs to provide low latency network services. For the network video live users, they need to use the network to carry out live activities. This type of application requires the network to provide a large amount of bandwidth in a short period of time to ensure the smoothness of the video picture, so it puts forward a high bandwidth service demand for the InPs. In addition, it also includes low cost, low memory consumption and other different QoS requirements. Therefore, large-scale end users/applications put forward differentiated QoS requirements for InPs [15–18].

On the other hand, with the mutual penetration of network technology and daily life, people's demand of using network to store personal information is more and more intense [19–23]. For bank accounts, health monitoring and electronic payment, these important private information is not expected to be exposed or stolen, and this information is the target of malicious devices or malware attacks [24, 25]. Therefore, the security of network services should be considered when satisfying the differentiated QoS requirements of network end users/applications [26–29].

In the NV environment, the requirements of network end users/applications are presented in the form of virtual network requests (VNRs). The SP is responsible for sending the VNR of the network end user/application. The purpose is to hope that the InP can allocate sufficient underlying network resources to meet the demand, that is, the problem of VNE [30–32]. This paper focuses on the problem of VNE with differentiated QoS requirements and security. According to the functional requirements of different users, this paper focuses on the design of differentiated QoS VNE algorithm from three aspects of VNE cost, network bandwidth and delay. In view of the security problems exposed in the process of VNE, a reasonable security level is set for VNR and substrate network. A VNE algorithm is designed to meet the requirements of differentiated QoS and security.

In order to improve the decision-making and optimization ability of VNE algorithm, deep learning method and reinforcement learning method are applied. The deep learning method is mainly used to solve the decision-making problem in high-dimensional space. By imitating the biological neural network to establish the network model, the problem that needs to be decided is input into the neural network and finally the optimal solution of the problem can be obtained [33, 34]. The reinforcement learning method mainly emphasizes the learning and training of agents in the interaction with the environment, and realizes the optimization of decision-making by using the evaluation feedback signal. In reinforcement learning method, training set data is usually used to train the agent and the agent adjusts the action through the size of feedback signal, so as to achieve the effect of continuous optimization. Finally, the optimal result can be obtained in the test data set [35–37]. The embedded problem of virtual network has been proved to be NP-hard [38], so deep reinforcement learning can be used to find the optimal solution for the embedded problem of virtual network to meet the differentiated QoS and security needs of network end users/applications.

The main work of this paper is as follows:

(1) Aiming at the differentiated QoS and security requirements of network end users/applications, this paper applies the representative DRL algorithm to the VNE problem. Through the efficient training results of agents, the differentiated QoS and security requirements of network end users/applications can be effectively solved.

(2) In the DRL algorithm, we extract four important attributes for the substrate network: CPU, bandwidth, delay and security. Using the custom policy network as the agent, the feature matrix is used as the input of the policy network for training. In this way, agents can be trained in a more realistic network environment, and the experimental results are also optimal. Finally, the mapping probability of each substrate node can be obtained.

(3) In order to prove the effectiveness of the algorithm, the VNE algorithm based on DRL is compared with other representative algorithms in three aspects of long term average revenue, long term revenue consumption ratio and acceptance rate. The experimental results show that the algorithm achieves good results and proves the effectiveness of the algorithm.

The rest of this paper is organized as follows: The second part describes the research status of VNE algorithms for differentiated QoS and security. The third part describes the problem of VNE and establishes the network model. The fourth part describes the implementation process of VNE algorithm based on differentiated QoS and security requirements. The fifth part introduces the setting of simulation experiment, then shows the experimental results and analyzes them. The last part summarizes the whole paper.

## 2 Related Work

### 2.1 VNE Algorithms Based on Differentiated QoS

In reference [39], a dynamic heuristic algorithm is proposed, which focuses on receiving as many VNRs as possible instead of optimizing the QoS performance of each VNR. When the QoS of VNR is not satisfied, the algorithm will drive the re-embedding scheme of heuristic algorithm to meet the given QoS requirements. Reference [40] considers that the existing solutions only aim at the congestion control problem of single objective VNE, and proposes a multi-objective VNE solution. Aiming at energy saving, energy sensing, avoiding network congestion and other service indicators, the embedding process of virtual network is

completed by combining the heuristic solution method based on SDN. In reference [41], a dynamic network resource allocation method based on load balancing and QoS is proposed. In this paper, the author proposes a QoS based scheduling mechanism for VNRs, which can reasonably rank incoming services by calculating the priority of VNRs. At the same time, this method uses a resource allocation mechanism based on load balancing to avoid the imbalance of resource consumption. In reference [42], an intelligent delay aware VNE scheme iVNE is proposed. This scheme focuses on the problem that the existing embedding algorithm of virtual network is not necessarily the optimal algorithm of industrial wireless network, but also lacks the QoS capacity. It provides delay guarantee for various industrial virtual networks, including static embedding process and dynamic forwarding process. Finally, the VNE algorithm achieves good load balancing ability. Reference [43] pays attention to the resource constraints and service quality issues of the IoV scenario. Based on artificial intelligence and machine learning, the author pushes cache and communication resources to the edge of smart cars, and jointly realizes the offloading of roadside units (RSU). The author uses a mixed integer nonlinear programming (MINLP) model to reduce the total network delay. The final experimental results prove that this method is effective in reducing user communication, computing, network congestion and content download delays.

### 2.2 VNE Algorithms Based on Security

In reference [44], network function virtualization (NFV) technology is applied to the field of network security. The network services processed by virtual network security function chain may be sensitive to specific network requirements, such as maximum bandwidth or minimum delay. The author proposes a gradual security service embedding scheme, which can optimize the resource utilization and deploy the virtual security function chain efficiently according to the security requirements of a single application and the strategy of the operator. In reference [45], a heuristic security aware VNE algorithm (SA-VNE) is proposed. The algorithm uses TOPSIS to sort the importance of the base nodes and select the most suitable base nodes. Finally, the shortest path algorithm is used to complete the link mapping process. In reference [46], trust relationship and trust degree are introduced into the problem of VNE, and the security problems in NV environment are analyzed quantitatively. This paper proposes a trust aware security VNE algorithm, which considers the local and global importance of nodes in the mapping process, and uses the approximate ideal ranking method to sort the substrate nodes. Finally, the k-shortest path method is used to complete the link mapping. In reference [47], the mapping process of security virtual network is modeled as a multi-objective mixed integer linear programming model, and a mapping algorithm of security virtual network based on multi-attribute comprehensive evaluation of nodes and path optimization is proposed. In this algorithm, the resource richness, security attributes and topological proximity of nodes are regarded as the criteria of node selection. In the link mapping phase, the available bandwidth and the number of path hops are used as the evaluation objects to select the mapping link. Finally the whole VNE process is completed.

From the above VNE algorithm based on differentiated QoS requirements and the security VNE algorithm, the existing VNE algorithm has done more perfect work. It cannot be ignored that they still have the following problems. First of all, in the research of VNE for QoS requirements, the author does not clearly point out which specific QoS indicators are. They regard the whole QoS as an evaluation standard, which does not reflect the characteristics of differentiation. There are no specific examples of differentiated QoS that need to be solved in the existing research, lacking of universal practical significance. Secondly, the research of security VNE algorithm only uses the traditional heuristic method. With the rapid development of intelligent learning method, it is of great significance to apply it to the problem of VNE. The intelligent learning method can allocate the network resources satisfying the security characteristics of VNRs, which has obvious advantages over the traditional heuristic method. Finally, there is no research on VNE algorithm which combines differentiated QoS requirements and security. In this paper, we will combine the differentiated QoS requirements and security issues, and apply DRL method to study the VNE algorithm.

## 3 Description and Model Establishment of VNE Problem with Differentiated QoS and Security

### 3.1 Description of VNE Problem with Differentiated QoS and Security

The different network functional requirements of network end users/applications are multiple heterogeneous VNRs. To allocate the underlying network resources reasonably for these VNRs, this process is called VNE. The embedding of virtual network can be divided into two parts: node embedding and link embedding [48, 49].

In the node embedding stage, virtual nodes need to find substrate nodes to meet their resource requirements. In order to better solve the problem of VNE with differentiated QoS requirements, we will focus on the cost of CPU resource consumption and the impact of node delay on QoS. For the security problem, the security requirement level attribute will be set for each virtual node, and each virtual node can only be embedded in the substrate node that meets its security requirements. Therefore, CPU resource, delay and security are three important indexes in the node attribute setting.

In the stage of link embedding, virtual link needs to find a substrate link to meet its resource requirements. A virtual link can be embedded in one substrate link or multiple substrate links through path segmentation. In order to better solve the problem of VNE with differentiated QoS requirements, we will focus on the cost of bandwidth consumption and the impact of link delay on QoS. Therefore, bandwidth resource and delay are two important indexes in link attribute setting.

In the whole process of VNE, CPU, bandwidth, delay and security are taken as the starting point to solve the problem of VNE with differentiated QoS requirements and security. By setting reasonable attributes for nodes and links, a reliable solution is provided for the problem of VNE facing differentiated QoS requirements and security.

### 3.2 Network Models

The undirected weighted graph $G^V = \{N^V, L^V, A_N^V, A_L^V\}$ is used to model the virtual network. $G^V$ represents a separate VNR. $N^V$ represents the collection of virtual nodes in the VNR, and $n^v$ represents a certain virtual node. $L^V$ represents the virtual link set in the VNR, and $l^s$ represents one of the determined virtual links. $A_N^V$ represents the attribute set of virtual node. For a specific virtual node $n^v$, its attributes include CPU resource requirement $CPU(n^v)$, delay level requirement $DELAY(n^v)$ and security level requirement $SR(n^v)$. $A_L^V$ represents the attribute set of virtual link. For a specific virtual link $l^v$, its attributes include bandwidth resource requirement $BW(l^v)$ and delay level requirement $DELAY(l^v)$.

The undirected weighted graph $G^S = \{N^S, L^S, A_N^S, A_L^S\}$ is used to build the mathematical model for the substrate network. $G^S$ represents the entire substrate network. $N^S$ represents the node set of the substrate network, and $n^s$ represents a certain substrate node. $L^S$ represents the set of links in the substrate network, and $l^s$ represents a certain substrate link. $A_N^S$ represents the attribute set of substrate nodes. For a specific substrate node $n^s$, its attributes include available CPU resource $CPU(n^s)$, delay level $DELAY(n^s)$ and security level $SL(n^s)$. $A_L^S$ represents the attribute set of the substrate link. For a specific physical link $l^s$, its attributes include the available bandwidth resource $BW(l^s)$ and delay level $DELAY(l^s)$.

Fig. 1 shows the topology of a virtual network and a substrate network. The virtual network consists of two virtual nodes and one virtual link. The three numbers next to the virtual node represent the CPU resource demand, delay demand level and security demand level of the virtual node. The two numbers on the virtual link represent the bandwidth resource demand and delay demand level of the virtual link. For a substrate network, the three numbers next to each substrate node represent the current available CPU resources, delay level and security level of the substrate node. Two numbers on each substrate link represent the amount of bandwidth resources and delay level that the substrate link can provide.

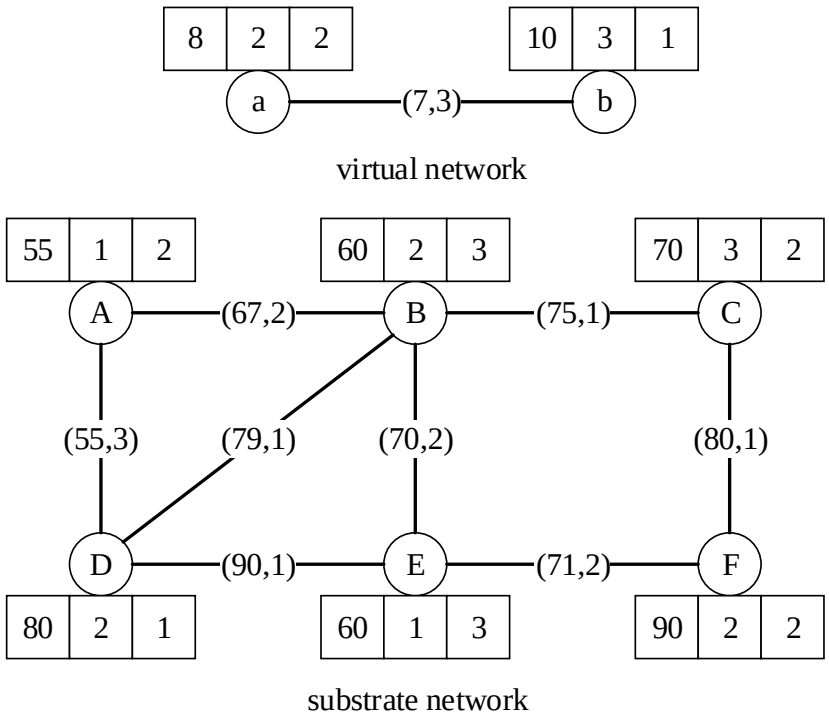



**Fig. 1** Diagram of virtual network and substrate network.

### 3.3 Constraints

The embedding of VNRs is limited by the number of underlying network resources. It is impossible for virtual network to be embedded on the underlying network without limitation. The successful embedding of every VNR consumes a certain amount

of network resources, mainly including the CPU resources of nodes and the bandwidth resources of links. In addition, due to the need for collaborative consideration of differentiated QoS and security of VNE, we stipulate that virtual nodes and virtual links can only be mapped to substrate nodes and substrate links whose delay level is not greater than its delay demand level. At the same time, it is necessary to ensure that the virtual node maps to a substrate node whose security level is not less than its security requirement level. With the above constraints, we can ensure that the VNE meets the differentiated QoS requirements and security requirements of CPU, bandwidth and delay. We will be involved in the formulation of constraints.

The current available CPU resources of substrate node $n^s$ can be represented by the remaining CPU resources:

$$CPU(n^s) = CPU_{initial}(n^s) - \sum_{all(n^v \uparrow n^s)} CPU(n^s). \tag{1}$$

$CPU(n^s)$ represents the current remaining CPU resources of substrate node $n^s$. $CPU_{initial}(n^s)$ represents the initial total CPU resources of substrate node $n^s$. $\sum_{all(n^v \uparrow n^s)} CPU(n^s)$ represents the CPU resources consumed by all virtual nodes mapped to substrate node $n^s$. The symbol $n^v \uparrow n^s$ indicates that virtual node $n^v$ is mapped to substrate node $n^s$.

The currently available bandwidth resources of substrate link $l^s$ can be represented by the remaining bandwidth resources:

$$BW(l^s) = BW_{initial}(l^s) - \sum_{all(l^v \uparrow l^s)} BW(n^s). \tag{2}$$

$BW(l^s)$ represents the current remaining bandwidth resources of substrate link $l^s$. $BW_{initial}(l^s)$ represents the initial total bandwidth resources of substrate link $l^s$. $\sum_{all(l^v \uparrow l^s)} BW(n^s)$ represents the bandwidth resources consumed by all virtual links mapped to substrate link $l^s$. The symbol $l^v \uparrow l^s$ indicates that virtual link $l^v$ is mapped to physical link $l^s$.

$$if\ n^v \uparrow n^s, CPU(n^v) \leq CPU(n^s), \tag{3}$$

$$if\ l^v \uparrow l^s, BW(l^v) \leq BW(l^s). \tag{4}$$

Formula (3) and formula (4) respectively represent the CPU resource constraint and bandwidth resource constraint embedded in the virtual network.

$$if\ n^v \uparrow n^s, DELAY(n^v) \geq DELAY(n^s), \tag{5}$$

$$if\ l^v \uparrow l^s, DELAY(l^v) \geq DELAY(l^s). \tag{6}$$

Formula (5) and formula (6) respectively represent the node delay constraint and link delay constraint embedded in the virtual network. Virtual node $n^v$ can only be mapped to substrate node $n^s$ which is not greater than its delay requirement level. Virtual link $l^v$ can only be mapped to substrate link $l^s$ which is not greater than its delay requirement level.

$$if\ n^v \uparrow n^s, SR(n^v) \leq SL(n^s). \tag{7}$$

Formula (7) represents the security constraints embedded in the virtual network. We set security requirement level for each virtual node and security level for each substrate node. Virtual node $n^v$ can only be mapped to substrate node $n^s$ which is not less than its security requirement level.

### 3.4 Evaluating Indicators

We take the long-term average revenue, acceptance rate and long-term revenue consumption ratio of VNE as the indexes to evaluate the performance of the algorithm. Because these three indicators can reflect the differentiated QoS requirements of CPU resource consumption, bandwidth resource consumption, delay constraint and security constraint to a certain extent.

The revenue of VNE is as follows:

$$R(G^V, t) = \sum_{n^v \in N^V} CPU(n^v) + \sum_{l^v \in L^V} BW(l^v), \tag{8}$$

where $R(G^V, t)$ represents the revenue embedded in the virtual network within the time period $t$ of the arrival of the VNR. $\sum_{n^v \in N^V} CPU(n^v)$ represents the CPU resource revenue obtained from the CPU resource consumed by virtual node $n^v$. $\sum_{l^v \in L^V} BW(l^v)$ represents the bandwidth resource revenue corresponding to the bandwidth resource consumed by the virtual link $l^v$. The revenue of VNE are determined by the sum of CPU resources and bandwidth resources consumed by VNRs.

The long-term average revenue of embedded virtual network is as follows:

$$R = \lim_{T \to \infty} \frac{\sum_{t=0}^{T} R(G^V, t)}{T}. \tag{9}$$

The consumption of VNE is expressed as follows:

$$C(G^V, t) = \sum_{n^v \in N^V} CPU(n^v) + \sum_{l^v \in L^V} BW(l^v) \times hops(l^v), \tag{10}$$

where $C(G^V,t)$ represents the consumption of VNE within the time period $t$ of VNR arrival. $\sum_{n^v \in N^V} CPU(n^v)$ represents the CPU resources consumed by virtual node $n^v$. $\sum_{l^v \in L^V} BW(l^v) \times hops(l^v)$ represents the bandwidth resources consumed by the virtual link $l^v$. Since a virtual link may be divided into multiple substrate links, $hops(l^v)$ represents the number of hops of the virtual link $l^v$. The consumption of VNE is determined by the sum of CPU resources and bandwidth resources consumed by VNRs.

The ratio of long-term revenue consumption embedded in virtual network is expressed as follows:

$$R/C = \lim_{T\to\infty} \frac{\sum_{t=0}^{T} R(G^V,t)}{\sum_{t=0}^{T} C(G^V,t)}. \tag{11}$$

The acceptance rate of VNE is expressed as follows:

$$ACC = \lim_{T\to\infty} \frac{\sum_{t=0}^{T} num(VNR_{acc})}{\sum_{t=0}^{T} num(VNR_{arr})}, \tag{12}$$

where $\sum_{t=0}^{T} num(VNR_{acc})$ represents the number of VNRs successfully embedded in the time range $t$. $\sum_{t=0}^{T} num(VNR_{arr})$ represents the total number of VNRs that arrive in the time range $t$.

### 3.5 An Example

Fig. 2 shows the different situations of two kinds of VNE. In case 1, the node mapping relationship is $a \uparrow A$, $b \uparrow B$, and the virtual link does not have path segmentation at this time. In case 2, the node mapping relationship is $a \uparrow F$, $b \uparrow D$, and the virtual link has path splitting. In both cases, the successful embedding of VNRs consumes the corresponding CPU resources and bandwidth resources. The delay constraints and security constraints are also satisfied. But in both cases, the ratio of revenue to consumption is different.

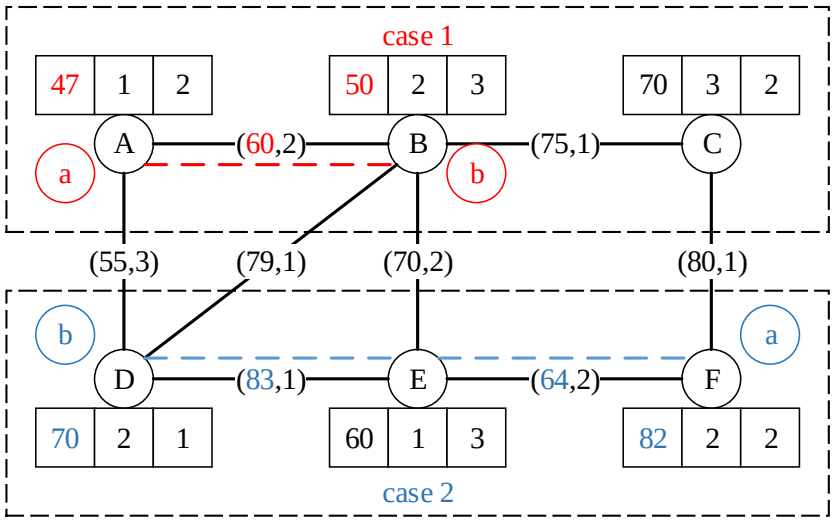



**Fig. 2** Example of virtual network embedding.

In case 1, the revenue of VNE is:

$$\begin{aligned} R(G^V,t) &= \sum_{n^v \in N^V} CPU(n^v) + \sum_{l^v \in L^V} BW(l^v) \\ &= CPU(n_a^v) + CPU(n_b^v) + BW(l_{ab}^v) \\ &= 8 + 10 + 7 \\ &= 25 \end{aligned} \tag{13}$$

The consumption of VNE is as follows:

$$\begin{aligned} C(G^V,t) &= \sum_{n^v \in N^V} CPU(n^v) + \sum_{l^v \in L^V} BW(l^v) \times hops(l^v) \\ &= CPU(n_a^v) + CPU(n_b^v) + BW(l_{ab}^v) \times hops(l_{ab}^v) \\ &= 8 + 10 + 7 \times 1 \\ &= 25 \end{aligned} \tag{14}$$

So the ratio of revenue to consumption in case 1 is 1.

In case 2, the revenue of VNE is:

$$\begin{aligned} R(G^V,t) &= \sum_{n^v \in N^V} CPU(n^v) + \sum_{l^v \in L^V} BW(l^v) \\ &= CPU(n_a^v) + CPU(n_b^v) + BW(l_{ab}^v) \\ &= 8 + 10 + 7 \\ &= 25 \end{aligned} \tag{15}$$

The consumption of VNE is as follows:

$$\begin{aligned} C(G^V,t) &= \sum_{n^v \in N^V} CPU(n^v) + \sum_{l^v \in L^V} BW(l^v) \times hops(l^v) \\ &= CPU(n_a^v) + CPU(n_b^v) + BW(l_{ab}^v) \times hops(l_{ab}^v) \\ &= 8 + 10 + 7 \times 2 \\ &= 32 \end{aligned} \tag{16}$$

So the ratio of revenue to consumption in case 2 is $\frac{25}{32}$.

It can be seen that path segmentation will result in greater cost of resource consumption. Therefore, in the design of VNE algorithm, it should try to avoid the generation of path segmentation.

## 4 Implementation of VNE algorithm Based on Differentiated QoS and Security Requirements

### 4.1 The Framework of VNE algorithm Based on DRL

With the continuous improvement of intelligent learning algorithm performance and the expansion of its application range, the application of DRL method to VNE algorithm will become the mainstream to solve the problem of VNE. The key of using DRL method to solve the problem of VNE lies in which kind of neural network is used to train the agent and how to create a realistic network environment for the agent. The embedded algorithm framework of virtual network based on DRL is shown in Fig. 3.

In order to train the DRL agent in a more real network environment, we extract four important attributes of the substrate network according to the differentiated QoS requirements and security requirements, so as to build the agent training environment. Feature extraction is described in detail below. DRL combines the advantages of deep learning and reinforcement learning, and agents can be well trained in our own policy network. Finally, a good result of VNE can be obtained.

### 4.2 Network Feature Extraction

The purpose of feature extraction of substrate network is to create a more real network environment for DRL agent. The DRL agent can make the best decision only when it is trained in the substrate network environment. Under the NV architecture, the substrate network is very complex and the network features available for extraction are very rich. Considering the computing power of our own policy network, if we extract too many network features, it will cause great computational complexity and reduce the performance of the algorithm. For differentiated QoS requirements and security requirements, we extract the following four attributes for each physical node:

*(1)* CPU resources. CPU resource is one of the most important resources in the network environment, which is an important factor affecting the cost of VNE. The CPU resource calculation method of the substrate node is shown in formula (1).

*(2)* The sum of bandwidth. The bandwidth sum of all links connected to a substrate node, which can reflect the bandwidth resource requirements of users. The link bandwidth sum connected to the substrate node is expressed as:

$$SUM\_BW(n^s) = \sum_{l^s \in L_{n^s}^S} BW(l^s). \tag{17}$$

*(3)* Delay. We set the delay attribute for both the substrate node and the substrate link. Virtual nodes and links can only be mapped to substrate nodes and links that are no higher than their delay requirements [50]. The specific expression is shown in formula (5).

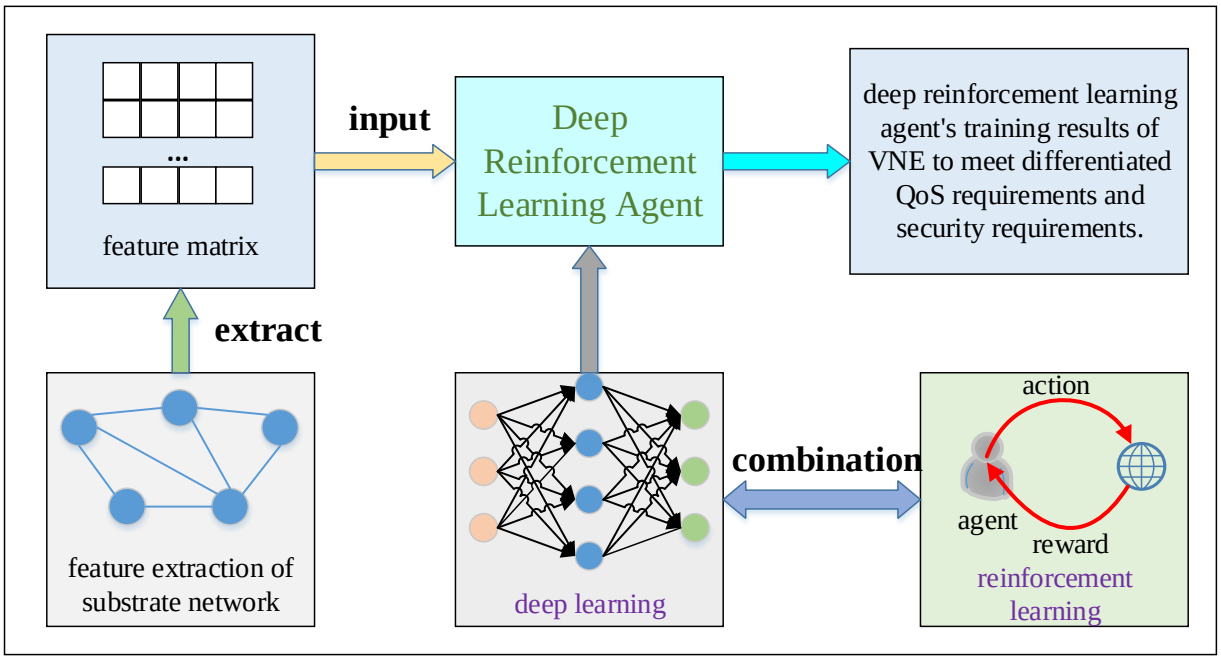



**Fig. 3** The framework of VNE based on DRL.

*(4)* Safety level. We set the security level for each substrate node. Virtual nodes can only be mapped to substrate nodes that are no lower than the security requirement level [51]. The specific expression is shown in formula (7).

CPU resources and bandwidth resources are the main resources consumed by VNRs. The evaluation criteria for evaluating the performance of VNE algorithm are also designed based on these two attributes. In order to meet the differentiated QoS requirements of network end-users/applications, it is necessary to consider the bandwidth resources and delay together. They deal with different QoS scenarios respectively. In addition, in view of the security problems in the network environment, innovatively considering the security attribute in the VNE problem can provide a new idea for the security of the VNE algorithm.

The extracted node attributes are concatenated into a four-dimensional feature vector. The feature vector of substrate node $n^s$ is expressed as:

$$v^{n^s} = \{CPU(n^s), BW(n^s), DELAY(n^s), RL(n^s)\}. \tag{18}$$

By combining the feature vectors of all substrate nodes, a four-dimensional feature matrix can be obtained. As the input of the policy network, the DRL agent is trained in the environment of the feature matrix. The feature matrix is expressed as:

$$M_f = \{v_1^{n^s}, v_2^{n^s}, ..., v_k^{n^s}\}. \tag{19}$$

### 4.3 Policy Network Construction

We use the basic elements of artificial neural network to build a simple policy network as a DRL agent. The extracted feature matrix is used as the input of the policy network, and the agent learns and trains in this environment, in order to deduce the probability of each substrate node being mapped. The resulting policy network is shown in Fig. 4. It mainly includes input layer, convolution layer, softmax layer and output layer.

The input layer is used to receive the feature matrix extracted from the substrate network, then use the policy network to evaluate the node attributes in the feature matrix. Convolution operation is performed on the feature vectors in the convolution layer to obtain the available resource vector $r_i$ of each eigenvector. The operation method is as follows:

$$r_i = \omega \cdot v_i + o, \tag{20}$$

where $r_i$ is the available resource vector of the ith feature vector. $\omega$ is the convolution kernel weight vector of the convolution layer. $v_i$ is the ith feature vector and $o$ is the offset.

In the softmax layer, a probability is generated for each node according to the available resource vector of each node, and the substrate nodes are ordered according to this probability. Softmax function is the extension of logical regression. It can convert n-dimensional vector into real value between 0 and 1 [52]. The calculation method is as follows:

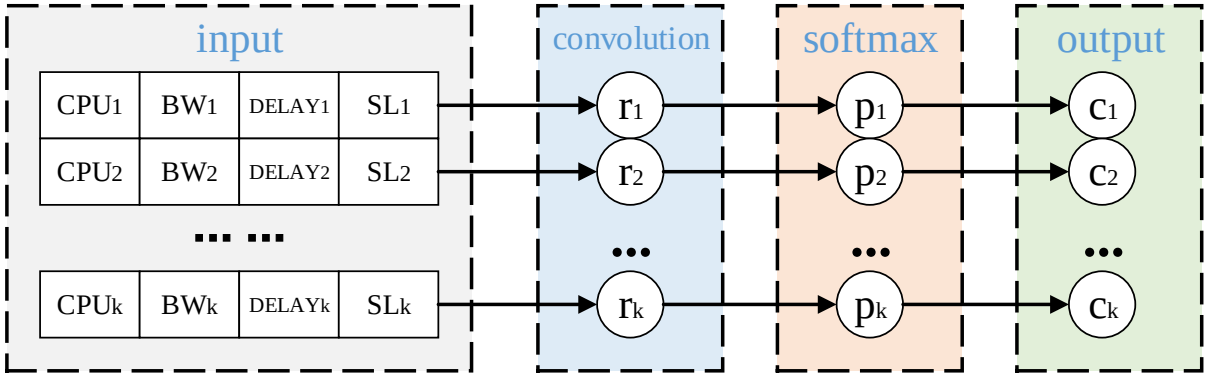



**Fig. 4** Policy network.

$$p_i = \frac{e^{r_i}}{\sum_k e^{r_k}}, \tag{21}$$

where $p_i$ represents the mapping probability of the ith substrate node. $k$ is the total number of feature vectors.

Finally, a set of available substrate nodes and their mapping probabilities are output at the output layer.

### 4.4 Training and Testing

In the training phase, whenever a VNR arrives, the policy network will extract a feature matrix from the substrate network as the input, so the feature matrix is dynamic. According to the input feature matrix, the agent can learn the most real situation of the substrate network and make the optimal decision. In reinforcement learning, in order to encourage the agent to make the best decision, a reward signal is usually set for the agent. The agent will decide which action to take according to the size of the reward signal. The size of the reward signal also depends on whether the agent's action is beneficial to it. The two are interactive. If an action taken by an agent receives a larger reward, the agent will take similar actions to accumulate the reward. In the problem of VNE, the revenue consumption ratio of VNE is usually used as the reward signal of agent. On the one hand, as an important evaluation index of VNE, the index has certain representativeness. On the other hand, the index reflects the utilization rate of the underlying network resources to a certain extent. There is a positive correlation between revenue consumption ratio and reward signal.

According to the method of supervised learning, a manual label is introduced for each feature vector in the policy network. Suppose that the manual label is introduced for the ith feature vector, then the label is 0 except that the ith position is 1. That is:

$$label_i = \{0, 0, ..., 1, ...0\}. \tag{22}$$

The cross entropy loss is calculated as follows:

$$Loss(label, c) = -\sum_i label_i \log(c_i), \tag{23}$$

where $label_i$ and $c_i$ are the ith element of label and the output of policy network respectively.

Using back propagation to calculate the gradient of parameters in the policy network:

$$g = \alpha \cdot r \cdot g_s, \tag{24}$$

where $\alpha$ is the size of training gradient. $r$ is the size of reward signal. $g_s$ is the stacking gradient.

The training process of DRL agent is shown in algorithm 1.

In the test phase, according to the node mapping probability obtained in the training phase, the virtual nodes are mapped in turn. Finally, the BFS is used to map the virtual links. The test process is shown in algorithm 2.

## 5 Experimental Setup and Result Analysis

In this part, we first introduce the setting of the experimental environment, then we will show the experimental results and analyze them.

**Algorithm 1** Training algorithm

**Input:** $epoch; \alpha; trainingset$;
**Output:** $trained\ parameters\ in\ policy\ network$;
$initialize\ parameters\ in\ policy\ network$;
**while** $iteration < epoch$ **do**
  **for** $request \in trainingset$ **do**
    $counter = 0$;
    **for** $node \in request$ **do**
      $M_f = getFeatureMatrix()$;
      $p = policyNetwork.output(M_f)$;
      $host = sample(p)$;
      $gradient(host)$;
    **end for**
    **if** $sn.cpu \geq vn.cpu\ and\ sl \geq sr\ and\ vn.delay \geq sn.delay$ **then**
      **if** $isMapped(\forall node \in request, \forall link \in request)$ **then**
        $reward = revToCost(request)$;
        $multiplyGradient(reward, \alpha)$;
      **else**
        $claergradients$;
      **end if**
    **end if**
    $counter++$;
    **if** $counter == batch\_size$ **then**
      $counter = 0$;
    **end if**
  **end for**
  $iteration++$;
**end while**
$return\ parameters$;

**Algorithm 2** Test algorithm

**Input:** $test\_set$;
**Output:** $long\ term\ average\ revenue, long\ term\ revenue\ consumption\ ratio, VNR\ acceptance\ rate$;
$initialize\ paraqmeters\ in\ policy\ network$;
**for** $request \in test\_set$ **do**
  **for** $node \in request$ **do**
    $M_f = getFeatureMatrix()$;
    $host = maxProbability(p)$;
  **end for**
  $BFSLinkMap(request)$;
  **if** $isMapped(\forall node \in request, \forall link \in request)$ **then**
    $return\ (success)$;
  **end if**
**end for**

### 5.1 Experimental Setup

We build a medium scale substrate network with 100 substrate nodes and 570 substrate links. The initial CPU resources of each substrate node are evenly distributed between 50 units and 100 units. In order to reflect the different requirements of users for delay characteristics, we set the delay level for each substrate node. The delay level is evenly distributed between 1 and 3. In order to reflect the user's demand for network security, we set the security level for each physical node. The security level is evenly distributed between 1 and 3. The initial bandwidth resources of each substrate link are evenly distributed between 50 units and 100 units, and the delay level is evenly distributed between 1 and 3.

We generated 2000 VNRs, of which the first 1000 were used as training sets and the last 1000 as test sets. Each VNR contains 2 to 10 different nodes, each node has a 50% probability of interconnection. The CPU resource demand of each virtual node is evenly distributed from 1 unit to 50 units. In order to reflect the different requirements of users for delay characteristics, we set the delay requirement level for each substrate node. The delay demand level is evenly distributed from 1 to 3. In order to reflect the user's demand for network security, we set the security requirement level for each virtual node. The security demand level is evenly distributed from 1 to 3. The bandwidth resource demand of each virtual link is evenly distributed between 1 unit and 50 units, and the delay demand level is evenly distributed between 1 and 3.

The arrival process of VNRs is simulated by Poisson distribution. The average arrival time of every 100 time units reaches 4 VNRs and the duration of each request follows the exponential distribution. We trained 100 epoch agents with gradient descent method, and the learning rate was set to 0.005.

The above experimental data are summarized in Table 1.

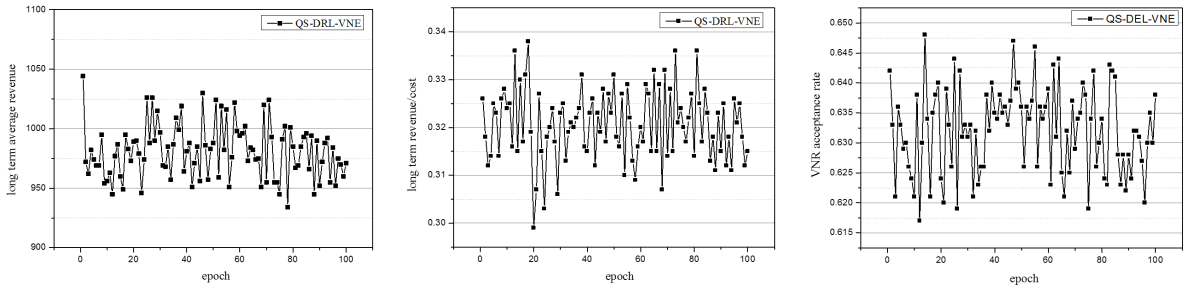
**Table 1** Parameter setting

| Parameter name | Parameter value |
|---|---|
| number of substrate nodes | 100 |
| number of substrate links | 570 |
| initial CPU resources | U[50, 100] |
| delay level of nodes | U[1, 3] |
| safety level of nodes | U[1, 3] |
| initial bandwidth resources | U[50, 100] |
| delay level of links | U[1, 3] |
| number of nodes per VNR | U[2, 10] |
| node connection rate | 50% |
| CPU resource demand | U[0, 50] |
| delay demand level of nodes | U[1,3] |
| safety demand level of nodes | U[1,3] |
| bandwidth resource demand | U[0, 50] |
| delay demand level of links | U[1,3] |

## 5.2 Results and Analysis

### *5.2.1 Training Results and Analysis*

Because the problem of VNE has been proved to be NP-hard and uncertain, the ultimate goal is to find an optimal solution. In order to achieve good results, we put the DRL agent on 100 epoch training sets. An epoch refers to the process of sending all the data into the network to complete a complete training. Through training, we get the stability degree of agent in three indexes: long term average revenue, long term revenue consumption ratio and VNR acceptance rate. The results are shown in Fig. 5.




(a) long term average revenue (b) long term revenue/cost (c) VNR acceptance rate

**Fig. 5** Results on training set.

In the initial stage of training, since the parameters of the policy network are randomly initialized, the agent will randomly take actions to explore the possibility of achieving good results, and the stability is poor at this time. In the middle of the training stage, as the agent becomes more and more familiar with the network environment, the agent will continue to find good solutions. At this time, the agent will get a larger reward signal. Meanwhile, the stability of the agent begins to improve gradually. In the later stage of training, the learning ability of the agent is limited by the performance of the policy network. At this time, the agent accumulates a certain degree of rewards, and the actions taken gradually tend to be stable, so the fluctuation range is small. The gradual stability of the curve proves the effectiveness of agent training, which lays a good foundation for the application of DRL agent in test set.

Fig. 6 shows the change of cross entropy loss during the training phase. It can be seen from the figure that the loss value of training keeps decreasing, which also proves that the training of policy network is effective.

It can be seen from the figure that the long term average revenue embedded in the virtual network and the acceptance rate of VNRs are decreasing with the increase of time, because these two indicators are limited by the number of network resources. When the underlying network resources are consumed, the number of VNRs it can carry will continue to decrease. Therefore, the long term average revenue and acceptance rate will show a downward trend. The change of the revenue consumption ratio

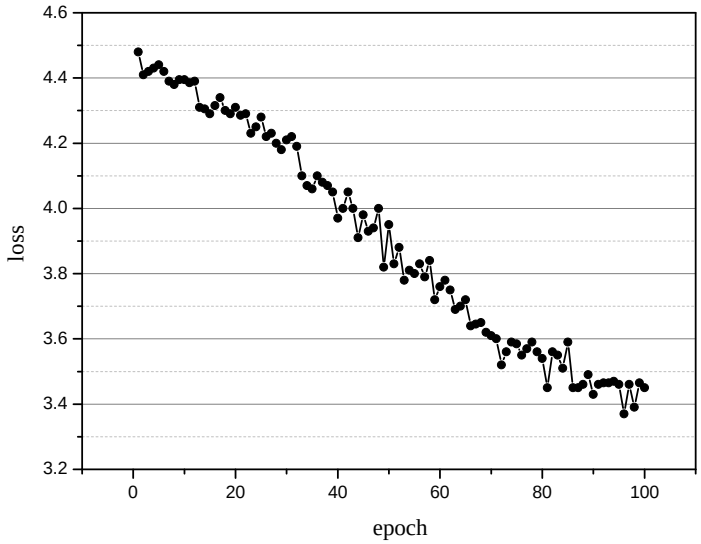



**Fig. 6** Cross entropy loss.

**Table 2** Parameter setting

| Algorithm | Content |
|---|---|
| QS-DRL-VNE | Focus on CPU, bandwidth, delay and security differentiated QoS algorithm. By extracting the features of the substrate network to form the feature matrix, the DRL agent is trained in this environment, and finally the node mapping probability is deduced. Use the BFS to complete the link mapping. |
| BASELINE | The formula $H(n^s) = CPU(n^s)\sum_{l^s \in L(n^s)} BW(L^S)$ is used to sort the substrate nodes, and the BFS is used to complete the link mapping. |
| BL-VNE | Using greedy strategy to complete node mapping, using shortest path algorithm to complete link mapping heuristic VNE algorithm. |
| CNL-VNE | VNE algorithm based on D-ViNE. In the process of node mapping, security constraints are added and the revenue and costs of VNE are modified according to security constraints. |

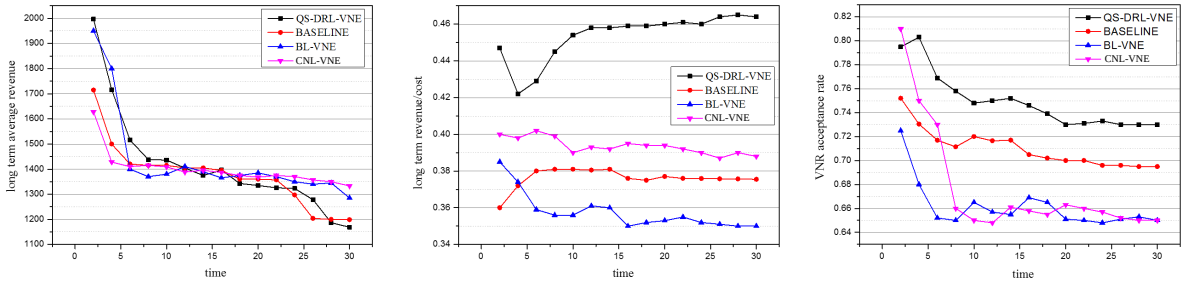
embedded in the virtual network has nothing to do with the quantity of the underlying network resources, so the index does not show a downward trend.

### *5.2.2 Test Results and Analysis*

After the training, the DRL agent is put in the test set to test, so as to prove the effectiveness of the algorithm. Since the mapping probability of each substrate node is obtained in the training phase, the mapping of virtual nodes is directly based on the probability in the test phase.

We compare the DRL-VNE algorithm based on differentiated QoS and security requirements (QS-DRL-VNE) with BASELINE algorithm [53], BL-VNE algorithm [45] and CNL-VNE algorithm [54]. BASELINE algorithm is a typical VNE algorithm based on intelligent learning method. BL-VNE algorithm is based on the mapping cost of virtual network. CNL-VNE algorithm is a security VNE algorithm. The core ideas of several algorithms are listed in Table 2. The experimental results show the performance of the algorithm in three aspects: long term average revenue, long term revenue consumption ratio and VNR acceptance rate, as shown in Fig. 7.

According to the comparison results of the three indexes, the DRL-VNE algorithm based on differentiated QoS and security requirements performs better than the other three algorithms, for two main reasons. First, based on differentiated QoS and security requirements, the DRL-VNE algorithm adopts efficient intelligent learning method to serve the problem of VNE, and efficiently solves the decision-making and optimization process of VNE. Compared with BL-VNE algorithm and CNL-VNE algorithm, intelligent learning algorithm has more advantages. Secondly, the DRL-VNE algorithm based on differentiated QoS and security requirements reasonably extracts the substrate network features, so that the DRL agent can be trained in a more real network environment, and finally achieves better results in the test set compared with the BASELINE algorithm. The above results show that the DRL-VNE algorithm based on differentiated QoS and security requirements is effective.




(a) long term average revenue (b) long term revenue/cost (c) VNR acceptance rate

**Fig. 7** Results on test set.

## 6 Conclusion

This paper combines the DRL algorithm with the VNE algorithm, and creatively solves the differentiated QoS and security requirements of network end users/ applications. In the framework of NV, the QoS and security requirements of network end users/applications are ultimately the problem of VNE. Using DRL method can improve the decision-making and optimization ability of VNE algorithm.

We attribute the QoS and security requirements of network end users/applications to four network indicators: CPU, bandwidth, delay and security. The DRL agent is trained in the network environment composed of these four attributes and finally the mapping probability of each substrate node is obtained. The experimental results show that it is feasible to solve the problem of VNE by this method and good results have been achieved. Therefore, it is of great practical significance to apply the algorithm to solve the differentiated QoS and security requirements of network end users/applications.

**Acknowledgements** This work is partially supported by the Major Scientific and Technological Projects of CNPC under Grant ZD2019-183-006, partially supported by the Project "Research on multi domain virtual network mapping algorithm for differentiated QoS requirements" supported by Shandong Provincial Natural Science Foundation, and partially supported by "the Fundamental Research Funds for the Central Universities" of China University of Petroleum (East China) under Grant 20CX05017A.

## References

1. P. Zhang, H. Yao, C. Qiu and Y. Liu, "Virtual network embedding using node multiple metrics based on simplified ELECTRE method," *IEEE Access*, vol. 6, pp. 37314-37327, 2018.
2. N. Kumar, G. S. Aujla, S. Garg, K. Kaur, R. Ranjan and S. K. Garg, "Renewable Energy-Based Multi-Indexed Job Classification and Container Management Scheme for Sustainability of Cloud Data Centers," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 5, pp. 2947-2957, 2019.
3. Z. Ning, P. Dong, X. Wang, et al., "When Deep Reinforcement Learning Meets 5G-enabled Vehicular Networks: A Distributed Offloading Framework for Traffic Big Data," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 2, pp. 1352-1361, 2020.
4. J. Du, C. Jiang, H. Zhang, Y. Ren and M. Guizani, "Auction Design and Analysis for SDN-based Traffic Offloading in Hybrid Satellite-Terrestrial Networks," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 10, pp. 2202-2217, 2018.
5. R. Munoz et al., "Integrated SDN/NFV management and orchestration architecture for dynamic deployment of virtual SDN control instances for virtual tenant networks [invited]," *IEEE/OSA Journal of Optical Communications and Networking*, vol. 7, no. 11, pp. B62-B70, 2015.
6. G. S. Aujla, R. Chaudhary, N. Kumar, J. J. P. C. Rodrigues and A. Vinel, "Data Offloading in 5G-Enabled Software-Defined Vehicular Networks: A Stackelberg-Game-Based Approach," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 100-108, 2017.
7. R. S. Batth, A. Nayyar and A. Nagpal, "Internet of Robotic Things: Driving Intelligent Robotics of Future - Concept, Architecture, Applications and Technologies," *2018 4th International Conference on Computing Sciences (ICCS), Jalandhar*, pp. 151-160, 2018.
8. G. Martins, L. F. Kopp and J. Genta et al. "A Prediction-Based Multisensor Heuristic for the Internet of Things," *the 15th ACM International Symposium on QoS and Security for Wireless and Mobile Networks*, pp, 71-78, 2019.
9. J. Du, E. Gelenbe, C. Jiang, H. Zhang and Y. Ren, "Contract Design for Traffic Offloading and Resource Allocation in Software Defined Ultra-Dense Networks," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2457-2467, 2017.
10. P. Zhang, H. Yao and Y. Liu, "Virtual Network Embedding Based on the Degree and Clustering Coefficient Information," *IEEE Access*, vol. 4, pp. 8572-8580, 2016.
11. A. K. Sandhu, R. Singh Batth and A. Nagpal, "Improved QoS Using Novel Fault Tolerant Shortest Path Algorithm in Virtual Software Defined Network (VSDN)," *2019 International Conference on Automation, Computational and Technology Management (ICACTM), London, United Kingdom*, pp. 383-388, 2019.
12. Z. Ning, K. Zhang, X. Wang and M. S. Obaidat et al., "Joint Computing and Caching in 5G-Envisioned Internet of Vehicles: A Deep Reinforcement Learning-Based Traffic Control System," *IEEE Transactions on Intelligent Transportation Systems, doi: 10.1109/TITS.2020.2970276*, pp. 1-12, 2020.
13. A. Jindal, G. S. Aujla, N. Kumar, R. Chaudhary, M. S. Obaidat and I. You, "SeDaTiVe: SDN-Enabled Deep Learning Architecture for Network Traffic Control in Vehicular Cyber-Physical Systems," *IEEE Network*, vol. 32, no. 6, pp. 66-73, 2018.
14. N. Aljeri and A. Boukerche, "An efficient handover trigger scheme for vehicular networks using recurrent neural networks," *the 15th ACM International Symposium on QoS and Security for Wireless and Mobile Networks*, pp. 85-91, 2019.
15. Y. Zhang, S. Q. Ren, S. B. Chen, B. Tan, E. S. Lim and K. L. Yong, "DifferCloudStor: Differentiated Quality of Service for Cloud Storage," *IEEE Transactions on Magnetics*, vol. 49, no. 6, pp. 2451-2458, 2013.
16. Xiong, Bing, Kun Yang, Jinyuan Zhao, Wei Li, and Keqin Li. "Performance evaluation of OpenFlow-based software-defined networks based on queueing model." Computer Networks 102 (2016): 172-185.

17. Zhang, Jianming, Wei Wang, Chaoquan Lu, Jin Wang, and Arun Kumar Sangaiah. "Lightweight deep network for traffic sign classification." Annals of Telecommunications 75, no. 7 (2020): 369-379.
18. Zhao, Jinyuan, Zhigang Hu, Bing Xiong, and Keqin Li. "Accelerating packet classification with counting bloom filters for virtual openflow switching." China Communications 15, no. 10 (2018): 117-128.
19. Aujla, G. S., Jindal, A., and Kumar, N. (2018). EVaaS: Electric vehicle-as-a-service for energy trading in SDN-enabled smart transportation system. Computer Networks, 143, 247-262.
20. N. S. Vishnu, R. Singh Batth and G. Singh, "Denial of Service: Types, Techniques, Defence Mechanisms and Safe Guards," *2019 International Conference on Computational Intelligence and Knowledge Economy (ICCIKE), Dubai, United Arab Emirates*, pp. 695-700, 2019.
21. Wang, Jin, Yu Gao, Wei Liu, Wenbing Wu, and Se-Jung Lim. "An asynchronous clustering and mobile data gathering schema based on timer mechanism in wireless sensor networks." Comput. Mater. Contin 58, no. 3 (2019): 711-725.
22. Wang, Jin, Yu Gao, Xiang Yin, Feng Li, and Hye-Jin Kim. "An enhanced PEGASIS algorithm with mobile sink support for wireless sensor networks." Wireless Communications and Mobile Computing 2018 (2018).
23. Ju, Chunwei, Yu Gao, Arun Kumar Sangaiah, and Gwang-jun Kim. "A PSO based energy efficient coverage control algorithm for wireless sensor networks." Computers, Materials and Continua 56, no. 3 (2018): 433-446.
24. N. V. Abhishek, T. J. Lim, A. Tandon and B. Sikdar, "Detecting forwarding misbehavior in clustered IoT networks," *the 14th ACM International Symposium on QoS and Security for Wireless and Mobile Networks*, pp. 1-6, 2018.
25. N. Ouferhat and A. Mellouk, "QoS dynamic routing for wireless sensor networks," *the 2nd ACM international workshop on Quality of service & security for wireless and mobile networks*, pp. 45-50, 2006.
26. I. Ahmad, S. Namal, M. Ylianttila and A. Gurtov, "Security in Software Defined Networks: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2317-2346, 2015.
27. V. Varadharajan, K. Karmakar, U. Tupakula and M. Hitchens, "A Policy-Based Security Architecture for Software-Defined Networks," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 4, pp. 897-912, 2019.
28. G. S. Aujla, R. Chaudhary, K. Kaur, S. Garg, N. Kumar and R. Ranjan, "SAFE: SDN-Assisted Framework for Edge–Cloud Interplay in Secure Healthcare Ecosystem," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 1, pp. 469-480, 2019.
29. S. Ziane and A. Mellouk, "A swarm intelligent multi-path routing for multimedia traffic over mobile ad hoc networks," *First ACM International Workshop on Quality of Service and Security in Wireless and Mobile Networks*, pp. 55-62, 2005.
30. P. Zhang, H. Yao and Y. Liu, "Virtual Network Embedding Based on Computing, Network, and Storage Resource Constraints," *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 3298-3304, 2018.
31. Aujla, G. S., Chaudhary, R., Kumar, N., Kumar, R., and Rodrigues, J. J. (2018, May). An ensembled scheme for QoS-aware traffic flow management in software defined networks. In 2018 IEEE International Conference on Communications (ICC) (pp. 1-7). IEEE.
32. O. S. Parra, G. Garica and B. Reyes, "Traffic forecasting using a multi layer perceptron model," *10th ACM Symposium on QoS and Security for Wireless and Mobile Networks*, pp. 133-136, 2014.
33. K. Kaur, S. Garg, G. S. Aujla, N. Kumar, J. J. P. C. Rodrigues and M. Guizani, "Edge Computing in the Industrial Internet of Things Environment: Software-Defined-Networks-Based Edge-Cloud Interplay," *IEEE Communications Magazine*, vol. 56, no. 2, pp. 44-51, 2018.
34. Z. Ning, P. Dong, X. Wang, J. Rodrigues and F. Xia, "Deep Reinforcement Learning for Vehicular Edge Computing: An Intelligent Offloading System," *ACM Transactions on Intelligent Systems and Technology*, vol. 10, no. 6, pp. 60, 2019.
35. C. Jiang, H. Zhang, Y. Ren, Z. Han, K. Chen and L. Hanzo, "Machine Learning Paradigms for Next-Generation Wireless Networks," *IEEE Wireless Communications*, vol. 24, no. 2, pp. 98-105, 2017.
36. J. Wang, C. Jiang, H. Zhang, Y. Ren, K-C. Chen and L. Hanzo, "Thirty Years of Machine Learning: The Road to Pareto-Optimal Wireless Networks," *IEEE Communications Surveys & Tutorials (DOI:10.1109/COMST.2020.2965856), Early Access*, pp. 1-1, 2020.
37. G. S. Aujla, R. Chaudhary, N. Kumar, A. K. Das and J. J. P. C. Rodrigues, "SecSVA: Secure Storage, Verification, and Auditing of Big Data in the Cloud Environment," *IEEE Communications Magazine*, vol. 56, no. 1, pp. 78-85, 2018.
38. P. Zhang, H. Yao, M. Li and Y. Liu. "Virtual network embedding based on modified genetic algorithm," *Peer-to-Peer Networking and Applications*, vol. 2, pp. 1-12, 2017.
39. H. Cao, S. Wu, G. S. Aujla, Q. Wang, L. Yang and H. Zhu, "Dynamic Embedding and Quality of Service-Driven Adjustment for Cloud Networks," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 2, pp. 1406-1416, 2020.
40. M. Pham, D. B. Hoang and Z. Chaczko, "Congestion-Aware and Energy-Aware Virtual Network Embedding," *IEEE/ACM Transactions on Networking*, vol. 28, no. 1, pp. 210-223, 2020.
41. S. Xu et al., "Load-Balancing and QoS Based Dynamic Resource Allocation Method for Smart Gird Fiber-Wireless Networks," *Chinese Journal of Electronics*, vol. 28, no. 6, pp. 1234-1243, 2019.
42. M. Li, C. Chen, C. Hua and X. Guan, "Intelligent Latency-Aware Virtual Network Embedding for Industrial Wireless Networks," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 7484-7496, 2019.
43. Z. Ning, K. Zhang, X. Wang and L. Guo et al, "Intelligent Edge Computing in Internet of Vehicles: A Joint Computation Offloading and Caching Solution," *IEEE Transactions on Intelligent Transportation Systems, doi: 10.1109/TITS.2020.2997832*, pp. 1-14, 2020.
44. R. Doriguzzi-Corin, S. Scott-Hayward, D. Siracusa, M. Savi and E. Salvadori, "Dynamic and Application-Aware Provisioning of Chained Virtual Security Network Functions," *IEEE Transactions on Network and Service Management*, vol. 17, no. 1, pp. 294-307, 2020.
45. P. Zhang, H. Li, Y. Ni, F. Gong, M. Li and F. Wang, "Security Aware Virtual Network Embedding Algorithm Using Information Entropy TOPSIS," *Journal of Network and Systems Management*, vol. 5, pp. 1-23, 2019.
46. S. Gong, J. Chen, C. Huang and Q. Zhu, "Trust-aware secure virtual network embedding algorithm," *Journal on Communications*, vol. 36, no. 11, pp. 1-10, 2015.
47. X. Liu, B. Wang, S. Liu, Z. Yang and Z. Zhao, "Heuristic algorithm for secure virtual network embedding," *Systems Engineering and Electronics*, vol. 40, no. 3, pp. 1-6, 2018.
48. J. Du, C. Jiang, Z. Han, H. Zhang, S. Mumtaz, and Y. Ren, "Contract Mechanism and Performance Analysis for Data Transaction in Mobile Social Networks," *IEEE Transactions on Network Science and Engineering*, vol. 6, no. 2, pp. 103-115, 2019.
49. P. Zhang, "Incorporating energy and load balance into virtual network embedding process," *Computer Communications*, vol. 129, pp. 80-88, 2018.
50. G. S. Aujla, A. Singh and N. Kumar, "AdaptFlow: Adaptive Flow Forwarding Scheme for Software-Defined Industrial Networks," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 5843-5851, 2020.
51. G. S. Aujla, A. Singh, M. Singh, S. Sharma, N. Kumar and K. R. Choo, "BloCkEd: Blockchain-Based Secure Data Processing Framework in Edge Envisioned V2X Environment," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 6, pp. 5850-5863, 2020.
52. P. Reverdy and N. E. Leonard, "Parameter Estimation in Softmax Decision-Making Models With Linear Objective Functions," *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 1, pp. 54-67, 2016.
53. M. Yu, Y. Yi, J. Rexford and M. Chiang, "Rethinking virtual network embedding: substrate support for path splitting and migration," *ACM SIGCOMM Computer Communication Review*, vol. 38, pp. 17-29, 2008.
54. N. M. M. K. Chowdhury, M. R. Rahman, R. Boutaba. "Virtual network embedding with coordinated node and link mapping," *Proceedings of the IEEE INFOCOM[C]. Rio de Janeiro*, pp. 783-791, 2009.