EURASIP Journal on Wireless Communications and Networking

RESEARCH **Open Access**

Dynamic handoff policy for RAN slicing by exploiting deep reinforcement learning

Yuansheng Wu[1*], Guanqun Zhao[2], Dadong Ni[1] and Junyi Du[1]

*Correspondence: wysefan@163.com
[1] Southwest China Institute of Electronic Technology, Chengdu 610036, China
Full list of author information is available at the end of the article

Abstract

It has been widely acknowledged that network slicing is a key architectural technology to accommodate diversified services for the next generation network (5G). By partitioning the underlying network into multiple dedicated logical networks, 5G can support a variety of extreme business service needs. As network slicing is implemented in radio access networks (RAN), user handoff becomes much more complicated than that in traditional mobile networks. As both physical resource constraints of base stations and logical connection constraints of network slices should be considered in handoff decision, an intelligent handoff policy becomes imperative. In this paper, we model the handoff in RAN slicing as a Markov decision process and resort to deep reinforcement learning to pursue long-term performance improvement in terms of user quality of service and network throughput. The effectiveness of our proposed handoff policy is validated via simulation experiments.

Keywords: RAN slicing, Dynamic handoff, MDP, Deep reinforcement learning

1 Introduction

It has been widely recognized that network slicing will play an important role in future mobile networks to support highly diverse quality of service (QoS) requirements from end users [1, 2]. Network slicing (NS) is defined as a technology to logically separate network functions and resources into multiple network slices (NSs) within a common physical infrastructure [3, 4]. Along with the benefits, the introduction of NS also brings many design challenges to the sliced radio access networks. Especially, handoff is crucial for keeping users connected while communication environment changes (e.g., user movement) [4], as it affects not only QoS of users but also network performance in terms of handoff rate, resource utilization, NS re-configuration frequency, etc. Considering the introduction of NS, conventional reference signal received power (RSRP)-based handoff schemes [5] are not applicable to RAN slicing. This happens because the target base station (BS) could not provide the required service for users if only considering RSRP when handoffs occur, and thus RSRP-based handoff scheme is unable to achieve the aim to provide guaranteed QoS for mobile users. Therefore, it is obligatory to design new handoff mechanisms dedicated for RAN slicing.

In slice-based mobile networks, the state of the network is constantly changing due to many factors, such as user arrival/departure, user mobility and randomness of access conditions. We need to make handoff decision according to the user's movement, as well as changes in the network environment and traffic load, in order to provide users with satisfactory access and transmission performance. In traditional mobile networks, the problem of handoff has attracted much research attentions. But very limited studies focus on the handoff problem for RAN slicing. In slice-based mobile networks, network slices and BS do not have a one-to-one correspondence. 3GPP standard TR.38.300 [5] states that the network slice is always composed of the RAN part and the core network (CN) part. Therefore, in the RAN slicing handoff problem, both the constraints of RAN and CN parts need to be considered. Moreover, in NS-based network architecture, in addition to the NS selection decision, the UE's BS association scheme needs to be considered. In other words, before the UE makes a handoff decision, we need to jointly consider BS association and NS selection.

Depending on the network status and user requirements, users may face the following scenarios when a handoff occurs: (1) switch to different slices on same BS; (2) switch to different base stations with the same slice; and (3) switch to different slices of different base stations. Thus RAN slicing handoff problem becomes much more complicated than the traditional BS handoff problem. Besides, the random nature of wireless channels, user mobility and the randomness of the access situation will cause the dynamics of the available resources in the network. All aforementioned factors may affect the user's handoff result.

In this paper, we propose an intelligent handoff strategy based on deep reinforcement learning for slice-based mobile networks. The main contributions of this work include:

1. In traditional mobile networks, handoff performs to achieve optimal matching between mobile users and access points. We proposed an intelligent handoff algorithm to obtain the optimal matching between mobile users, network slices and access points in slice-based mobile networks.
2. In this work, the user handoff problem in slice-based mobile network is modeled as an MDP. We thus resort to reinforcement learning theory [6, 7] and propose an intelligent handoff algorithm based on deep reinforcement learning. The users observe and independently learn the network states in order to select a suitable slice/base station pair to access.
3. The effectiveness of our proposed handoff algorithm is validated via simulation experiments. The numerical results show that compared with the typical handoff algorithm in traditional networks, the proposed algorithm can enable users to obtain better transmission performance.

The rest of the paper is organized as follows. In Section II, we present the related work. In section III, we describe the system model. Section IV formulates the handoff problem in slice-based mobile networks. We present the solution to the handoff problem in Section V. Section VI provides numerical results for the performance comparison. Finally, Section VII concludes the paper.

2 Related work

In this section, we overview the related work from two perspectives, handoff algorithm design for traditional cellular networks and handoff for sliced networks.

2.1 Handoff for traditional cellular networks

In recent years, existing handoff strategies in traditional networks usually take into account UE SINR, QoS, mobility and traffic load of the base stations [8, 9]. In [8], the authors design a handoff algorithm based on the estimated load of the cell and improve the system energy efficiency by combining the handoff strategy with the base station sleep strategy. The authors of [9] propose a new handoff algorithm that effectively controls the base station transmission power and reduces redundant handoff. In addition, some research work uses machine learning to solve the handoff problem [10–12]. The authors of [10] proposed a learning-based smart handoff strategy, which reduces the number of handoffs under the premise of guaranteeing user service quality. In LTE-based heterogeneous networks, the authors of [13] and [12] model the user handoff problem as a Markov decision process (MDP) problem and propose a handoff decision based on the value iterative algorithm. In [11], the authors aim to maximize the user's QoE (quality of experience) and propose a handoff strategy based on reinforcement learning.

3 Handoff for sliced networks

Very little attention is paid to the handoff problem in slice-based mobile network. Specifically, the authors of [14] proposed a multi-agent reinforcement learning-based smart handoff scheme to minimize handover cost while maintaining user QoS. Several different kinds of handoff type in sliced network have been considered in this work. In work [15], the authors proposed a device association scheme for RAN slicing by exploiting a hybrid FL reinforcement learning (HDRL) framework, to improve network throughput while enhancing data security.

Besides handoff algorithm design, a few works have studied user association problem in sliced networks, which can also give some insights for designing handoff algorithm in sliced networks. In work [16], the authors proposed a unified framework for user association in RAN slicing with aim of maximizing resource utilization while guaranteeing QoS of users. Both admission control and resource allocation in RAN slicing have been carefully investigated by the authors. In work [17], the authors investigated the optimal selection of end-to-end slices with the aim of improving network resources utilization while guaranteeing the QoS of users.

4 System model

We consider a slice-based mobile network model as shown in Fig. 1 [5, 18], which consists of M base stations and N network slices. Different slices share system transmission resources, including wireless transmission bandwidth and power at the RAN

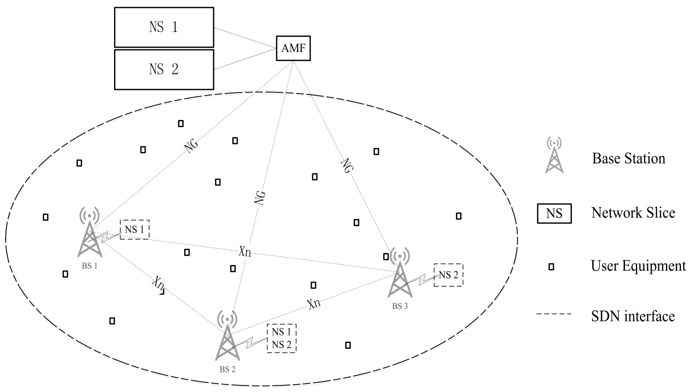

Fig. 1 Network slice-based network architecture. We consider a slice-based mobile network model as shown in this figure, which consists of M base stations and N network slices. Different slices share system transmission resources, including wireless transmission bandwidth and power at the RAN and slice capacity at the CN. A slice may cover multiple BSs, one BS may also be covered by multiple slices, and transmission resources in the BS are allocated to multiple slices

and slice capacity at the CN. A slice may cover multiple BSs, one BS may also be covered by multiple slices, and transmission resources in the BS are allocated to multiple slices. In slice-based networks, the base stations can be connected to each other through the Xn interface, and each base station is connected to the Access and Mobility Management Function (AMF) in the core network through the NG interface. Through the SDN interface, information exchange can be performed between the BS and the BS, and between the BS and the AMF. One UE may be in the coverage of multiple BSs. The slice information accessible on different BSs is learned through BS broadcast, and the appropriate slice access is selected therefrom. Through the RAN, the user can select the AMF that supports the accessed slice. Through the AMF, the user is assigned the corresponding core network resource.

In slice-based mobile networks, user mobility and dynamic network conditions may lead to different access conditions at different times for mobile users. At different times, users can adaptively adjust their access policies through dynamic handoff according to the perceived network environment and changes in service requirements. As shown in Fig. 1, a user may be within the coverage of multiple slice signals of multiple different BSs. It is assumed that the user can periodically receive network status information within the signal range, such as the available bandwidth of the slice and the average transmission delay. According to the dynamic changes of the network status, the user will decide whether to switch the access slice to obtain better access and transmission performance. Switching the access slice is a complicated process. The network needs to complete the conversion of the user access state through a series of signaling interactions, which brings additional processing cost and signaling overhead to the network. Therefore, when performing the handoff decision in slice-based mobile networks, it is necessary to consider how to improve the transmission performance of the user, and consider the impact of the additional processing cost and signaling overhead on the system performance.

5 Problem formulation

In this paper, we consider that the user will check the current network status periodically and decide whether to perform handoff decision to obtain better transmission performance according to the current network status. For users, the goal is to maximize their long-term transmission performance. In this section, we model the handoff problem into a Markov decision process.

Considering the system model of Fig. 1, at each decision point, the agent (UE) needs to make a decision to select the best slice for the user to access. This may change the state of the network, causing the network to transfer to another state. Based on this, we model the handoff problem into a Markov decision process. In order to solve the handoff problem using the relevant theory of MDP, we need to define the state S of the system, the action A and the reward function R of the action.

Considering the UE as the agent, the handoff problem in slice-based network can be described by the quaternion $M=(S, A, P, R)$:

State: Let $s \in S$ be the network state, where S is the set of all states. If there are N slices in the network, M base stations, $s = [I, B_{1,1}, \ldots, B_{j,k}, \ldots B_{M,N}]$ where $I=(j, k)$ represents the current access state, and $B_{j,k}$ represents the available bandwidth that the network can provide when user accesses slice j through base station k. According to the actual network situation, when the BS k is not associated with the slice j, we can remove the corresponding $B_{j,k}$ from the state vector to reduce our state space. Since network resources are limited, we use $b_{j,k}^{\max}$ to indicate the maximum bandwidth that the network can provide when accessing slice j through base station k.

Action: In the slice selection model, we define the action as the user's choice behavior for the slice. When a user needs to access the network, it has N slices to choose from, and each slice is associated with one or more BSs. Therefore, the selection of the BS is also required while selecting the slice. Let action $a=(j, k)$ denote that the UE accesses the slice j through the BS k, and $a \in A$ where $A = \{(j,k)|1 \leq j \leq N, 1 \leq k \leq M\}$ is action space. When the selected action is performed, the agent transfers to the next state with a certain transition probability.

Transition probability: $P = \{p_{s,s'}^{a}|s, s' \in S, a \in A\}$ is a state transition probability set. $p_{s,s'}^{a}$ is the probability that the state transfer from s to s' when performs action a. We assume that this probability is unknown.

Reward: There are two main factors that need to be considered when defining the immediate reward of action a at state s. One is the transmission rate obtained when the user takes action a. The second is whether action a causes the handoff to occur. There are many ways to quantify the user's reward function. We will take the reward obtained by taking action a in state s as:

$$r(s,a) = f(s,a) - g(s,a) \tag{1}$$

where $f(s,a)$ is the service rate obtained by the user, and $g(s,a)$ is the handoff overhead generated after the action is taken. When the UE transfers from $s = [I, B_{1,1}, \ldots, B_{j,k}, \ldots B_{M,N}]$ to $s' = [I', B'_{1,1}, \ldots, B'_{j,k}, \ldots B'_{M,N}]$ by performing action a, its handoff overhead function is defined as [12, 13]:

$$g(s,a) = \begin{cases} K_{I,I'}, & I \neq I' \\ 0, & I = I' \end{cases} \tag{2}$$

Here, $K_{I,I'}$ represents the handoff overhead when switching from the access state I to the access state I'. The value of $K_{I,I'}$ depends on the signaling overhead, processing delay, etc., caused by the handoff performed in the actual network.

6 Methods

6.1 Q-learning

Q-learning is a kind of temporal-difference learning algorithm. Its advantage is that it does not need to know the environment model and can be used for continuous tasks. It can be known from the definition of the action value function that the state value function $V^*(s)$ and the action value function $Q^*(s,a)$ corresponding to the optimal strategy π^* satisfy the following relationship:

$$V^*(s) = \max_a Q^*(s,a) \tag{3}$$

That is to say, if we have obtained the action value function of the system through learning, under the state s, the action corresponding to the maximum action value function is the optimal strategy. In Q-learning algorithm, the formula for updating the Q value is stated as follows [11]

$$Q(s,a) = Q(s,a) + \alpha_t[R + \gamma \max_{a_*} Q(\mathrm{s}', a_*) - Q(s,a)] \tag{4}$$

where α_t denotes the learning rate and γ denotes the discount factor. As can be seen from Eq. (4), in state s, action a is selected, and the action function value is updated by the obtained reward R and the largest $Q(\mathrm{s}', a)$.

In reinforcement learning problem, the agent not only needs to learn to get the optimal strategy, but also to make the decision by the learned strategy. This is also the problem of exploration and exploitation in the Q-learning algorithm. In this case, we adopt the ε - greedy strategy, that is, the agent randomly selects an action with the probability of ε and selects the best learning currently learned by the probability of $1-\varepsilon$. Then, by slowly reducing the value of ε, the algorithm is finally converged and the optimal strategy is obtained.

6.2 Deep Q-learning algorithm

In the previous section, the implementation of the Q-learning algorithm is based on a precondition: the state space and the action space are discrete, and the state space and the action space cannot be very large. It is worth noting that the value function is in a table. For state value functions, the index is the state. For action value functions, the index is a state-action pair. The iterative update of the value function is actually an iterative update of this table. Therefore, this form of reinforcement learning is also called tabular reinforcement learning.

However, in some scenarios, the state space could be very large. Even in some cases, we will face the problem of continuous state space. At this point, value iterative functions and tabular reinforcement learning algorithms such as Q-learning will no longer be applicable. We need to represent the value function using the value function approximation method. Common approximation methods include linear approximation, neural network approximation and so on. In the value function approximation method, the value function corresponds to an approximation function $\hat{Q}(s, a)$. When the approximating value function structure is determined, then the value function approximation is equivalent to the parameter approximation, and the update of the value function is equivalent to the parameter update.

The Deep Q-Learning (DQN) algorithm is a kind of end-to-end learning algorithm that combines deep learning and reinforcement learning. It is an effective way to solve the problem of continuous state space. Its modifications to Q-learning are mainly reflected in:

1. DQN uses a deep neural network to approximate the value function. As mentioned above, the method of value function approximation consists of linear value function approximation and nonlinear value function approximation. In practical problems, since it is difficult to guarantee the linear relationship between input and output, it is obvious that the method of linear approximation cannot obtain accurate value function approximation results. Therefore, the researchers propose to solve this problem by using nonlinear approximation. A commonly used method in nonlinear approximation is to use neural networks for value function approximation. When a neural network training for value function approximation is completed, the current state s and action a are input, and the output is the corresponding action value function $Q(s, a)$.
2. DQN uses experience replay training to enhance the learning process of learning. An important condition when training a neural network is that the data of the training set must satisfy the independent and identical distribution. However, there is a correlation between the training concentration data obtained by the reinforcement learning sampling. If these data are used for sequential training, it is easy to cause instability of the neural network. To solve this problem, DQN uses experience replay mechanism to break the correlation between data. The agent stores the data in an experience replay pool and then uses the uniform random sampling method to extract data from the experience replay pool for neural network training.
3. DQN independently sets the target network to handle the TD bias in the temporal-difference algorithm separately. Unlike the tabular reinforcement learning algorithm, when DQN uses the neural network to approximate the value function, it actually updates the parameter value θ. Since the parameter update method in the neural network is the gradient descent method, the update of the parameters in the DQN actually becomes an update process of supervised learning. The update strategy is

$$\theta_{t+1} = \theta_t + \alpha[r + \gamma \max_{a'} Q(s', a'; \theta) - Q(s, a; \theta)]\nabla Q(s, a; \theta) \tag{5}$$

where $r + \gamma \max_{a'} Q(s', a'; \theta)$ is called the TD target. From Eq. (5), we can see that the parameter θ used in calculating the TD target is the same as the parameter θ used in calculating the gradient. This easily leads to the correlation between the data, which makes the training results unstable. In order to solve this problem, the target network method is adopted in DQN to further reduce the correlation between data. The basic idea is: The network parameter for calculating the TD target is denoted as θ^-, and the network parameter for which the calculated value function is approximated is denoted as θ. In the process of executing the algorithm, the network parameters for the approximation of the action value function are updated every step. The parameters used to calculate the TD target network are updated every certain number of steps. Therefore, the parameter update strategy is

$$\theta_{t+1} = \theta_t + \alpha[r + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta)]\nabla Q(s, a; \theta) \tag{6}$$

6.3 DQN-based handoff decision algorithm

In this section, we apply the DQN algorithm to the handoff problem of network slices. Based on the theoretical model in Section III and the analysis in Section IV-B, we design the architecture diagram of the deep reinforcement learning algorithm shown in Fig. 2. During the execution of the algorithm, the physical network needs to provide information such as network status, user actions and reward functions. Using this information, the agent performs deep Q-learning algorithm as shown in algorithm 1 for training.

In this algorithm, we design the neural network used for the approximation of the value function as the three-layer neural network, namely the input layer, the hidden

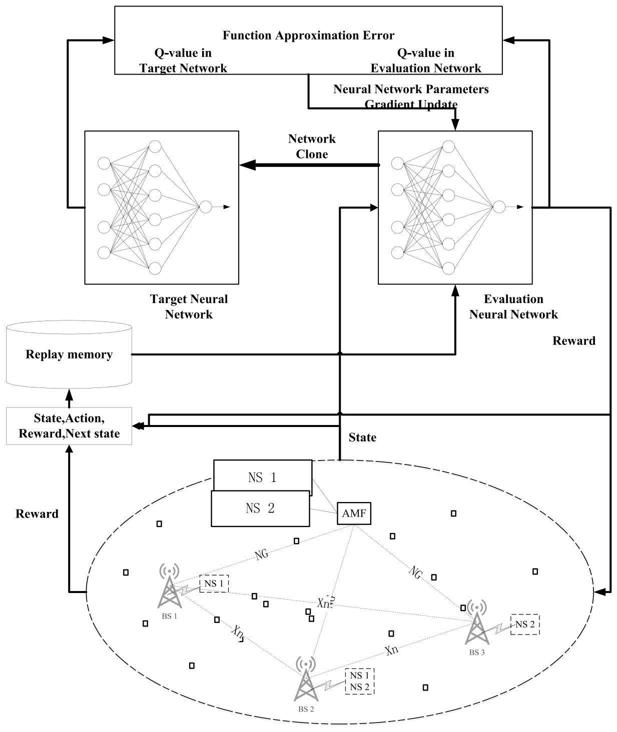
Function Approximation Error
Q-value in Target Network
Q-value in Evaluation Network
Neural Network Parameters Gradient Update
Network Clone
Target Neural Network
Evaluation Neural Network
Replay memory
Reward
State,Action, Reward,Next state
State
Reward
NS 1
NS 2
AMF
NG
NG
Xn
Xn
Xn
NS 1
BS 1
NS 2
BS 3
NS 1
NS 2
BS 2

Fig. 2 Deep reinforcement learning. We design the architecture diagram of the deep reinforcement learning algorithm as shown in this figure. During the execution of the algorithm, the physical network needs to provide information such as network status, user actions and reward functions. Using this information, the agent performs deep Q-learning algorithm as shown in algorithm 1 for training. we design the neural network used for the approximation of the value function as the three-layer neural network

layer and the output layer. The input layer consists of a set of nodes representing the state-action space, and the number of nodes depends on the size of the state-action space. According to the theoretical model in Section IV, we take the user's current access status, network available resource status and access actions as inputs. The output layer consists of a node whose output represents the Q-value of the neural network fit under the current parameters. The nodes between the input layer and the hidden layer, hidden layer and output layer are fully connected model. The value of the connection weight between each pair of nodes determines the ability of the neural network to correctly approximate the Q value.

In the actual network scenario, the RAN obtains related information of all network slices through the CN and notifies the UE of the slice information and the corresponding resource information. The UE stores the acquired information and learns using the DQN algorithm. In the actual handoff, the UE first determines its current state and obtains the Q-value corresponding to each action through the currently trained neural network. Then agents choose handoff action according to the $\varepsilon - greedy$ strategy. It is worth noting that although the algorithm is an online learning algorithm; the agent can collect training information and train its strategy in the background in the gap between every two decision time points. In addition, the resulting strategy of training can also be applied to similar scenarios. Therefore, the DQN-based handoff algorithm can efficiently perform and make handoff decisions in accordance with the policies it has learned.

Algorithm 1 Deep Q-Learning Algorithm

```
Input: Network State $S$, Action Space $A$, Reward $R$, Learning rate $\alpha$,
       Discount Factor $\gamma$
Output: Handoff Strategy $(\pi_*^1,\ldots,\pi_*^n)$

1: Initialize the experience replay pool $D$. The number of data is $H$.
2: Initialize the action value function $Q$ with random weight $\theta$
3: $\theta^- = \theta$ . Initialize the target network value function $\hat{Q}$ with parameter
$\theta^-$
4: For episode=1,2,⋯⋯, $G$ do
5:   Judging the current state $s_I$
6:   For $t$=1,2⋯⋯$T$ do
7:     With probability $\varepsilon$ select a random action $a_t$
8:     otherwise select $a_t = \arg\max_a Q(s_t, a; \theta)$
9:     Execute action $a_t$ and observe reward $r_t$ and next state $s_{t+1}$
11:    Store data $(s_t, a_t, r_t, s_{t+1})$ in $D$
12:    Sample random data $(s_j, a_j, r_j, s_{j+1})$ from $D$
13:    Set
$y_j = \begin{cases} r_j & \text{if episode terminates at step } j+1 \\ r_j + \gamma \max_{a'} \hat{Q}(s_{j+1}, a'; \theta^-) & \text{otherwise} \end{cases}$
14:    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ and update
        parameter $\theta$
15:     Every C steps reset $\theta^- = \theta$
16:   End For
17: End For
```

6.4 DQN-based handoff process

In detail, as shown in Fig. 3, the UE periodically measures and reports the obtained QoS to the source BS and NS, and the source SDN controller checks if the handover trigger condition is satisfied (such the received SINR is lower than a threshold). Then the UE uses DQN to select the target BS and NS and sends the handover request to the corresponding SDN controllers. After the confirmation of handover command, this handover is executed by the target and the source SDN controllers. Before the handover is completed, the target SDN controller calculates the reward value of this handover decision and then broadcasts the reward to the UEs. The UEs served by the same type of this target NS use DQN to update Q-table. Finally, the resource of source BS and NS is released by the SDN controller.

7 Results and discussion

In this section, we compare the DQN-based handoff algorithm with greedy, RSS-based and other algorithms to verify the performance of our proposed algorithm. In greedy algorithm, the UE always choose the action with the highest reward. And the UE which performs RSS-based algorithm selects the slice that provides the largest received signal strength (RSS). In addition, with reference to [13], we consider the following two handoff strategies: (1) At each decision point, the UE selects the slice that provides the largest available bandwidth, which we denote as BW-based. (2) At each decision point, the UE does not perform a handoff action, and we denote this policy as fixed.

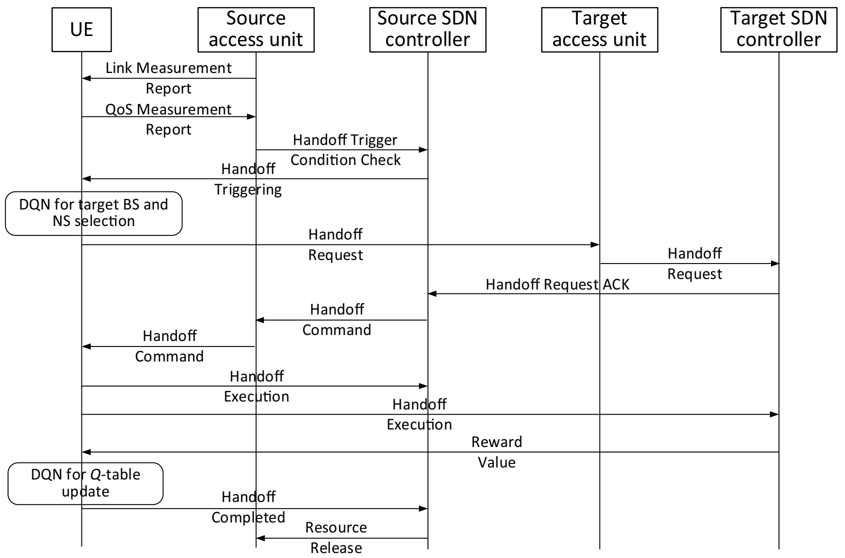

Fig. 3 DQN-based handoff process. As shown in the figure, the UE periodically measures and reports the obtained QoS to the source BS or NS, and the source SDN controller checks if the handover trigger condition is satisfied. Then the UE uses DQN to select the target BS and NS and sends the handover request to the corresponding SDN controllers. After the confirmation of handover decision, this handover is executed by the target and the source SDN controllers. Before the handover is completed, the target SDN controller calculates the reward value of this handover decision and then broadcasts the reward to the UEs. The UEs served by the same type of the target NS use DQN to update Q-table. Finally, the resource of source BS and NS is released by the SDN controller

Table 1 Network parameters

Parameter	Value
Number of BSs M	3
Number of slices N	2
Number of UEs	Less than 8
$b_{j,k}^{max}$	U[0,10] units
handoff signaling overhead $K_{x,y}$	1060
arrival parameter λ	[2, 1, 1, 2]
leaving parameter μ	[1, 2, 1, 2]

Table 2 Algorithm parameters

Parameter	Value
Number of input neurons	8
Number of hidden neurons	25
Number of output neurons	1
Learning rate	0.001
Exploration parameter ε	0.1
C	5
discount factor γ	0.9

7.1 Simulation settings

We consider the network scenario as shown in Fig. 1. The user's position is fixed or low-speed motion near its initial position, and therefore, its channel condition can be considered to be unchanged. In the range covered by the signals of the three base stations, two network slices are deployed for the UE to select, and the deployment relationship is the same as that of Fig. 1. The system parameters are listed in Table 1. The maximum bandwidth that the network can provide when the BS k accesses the slice j is randomly selected from [0, 10] units. In order to simplify the model, we assume that the signaling overhead $K_{x,y}$ satisfies $K_{x,y} = K_{y,x}$. In order to simulate a real dynamic network scenario, refer to [11, 12], the number of users in each slice dynamically changes with time. The user will arrive at the network according to the Poisson process with the parameters $\lambda = [\lambda_1, \lambda_2,\ldots,\lambda_n]$ and the leave with the parameter $\mu = [\mu_1, \mu_2, \ldots, \mu_n]$, where λ_n denotes the user arrival parameter of the *nth* slice and μ_n denotes the user leaving parameter of the *n-th* slice.

On this basis, the three-layer fully connected neural network is used for the approximation of value functions in reinforcement learning. The relevant parameters are shown in Table 2. The input layer contains 8 neurons, indicating the state and action. The hidden layer consists of 25 neurons whose activation function is set to the sigmoid function. The output layer has one neuron, and the activation function is a linear function. During the execution of the DQN algorithm, we set the exploration parameter to 0.1, the interval number C to 5 and the discount factor to 0.9.

7.2 Numerical results

According to Tables 1 and 2, in the network scenario shown in Fig. 1, this section compares the performance of the DQN algorithm with other algorithms by simulation. The result is the mean of 100 random simulation results.

Figure 4 shows the overall cumulative reward of the system as the handoff decision progresses. As can be seen from the figure, as the number of decision steps increases, the cumulative return of each strategy gradually increases. Among them, the DQN-based handoff algorithm performs better than other algorithms. And as the number of decision steps increases, the advantages of the algorithm become more apparent. As mentioned earlier, the fixed algorithm takes a non-handoff strategy and cannot actively switch when there are better slices. BW-based only considers the currently available bandwidth, and RSS only considers channel conditions. Both schemes do not consider handoff overhead. Although greedy takes into account bandwidth, channel conditions and handoff overhead, it does not consider the behavior of other users and the impact of external environment changes. Therefore, its performance is lower than the spectrally efficient RSS algorithm. The proposed DQN algorithm can comprehensively consider the bandwidth, channel conditions, handoff overhead and the change law of the external environment in a learning way to obtain more optimized performance. Further, Fig. 5 shows the variation in system throughput as the number of decision steps increases. As can be seen from the figure, the DQN algorithm always achieves the highest throughput performance.

In Figs. 6 and 7, we compare the handoff probability and the cumulative handoff times of different algorithms with the progress of the handover decision. As can be seen from

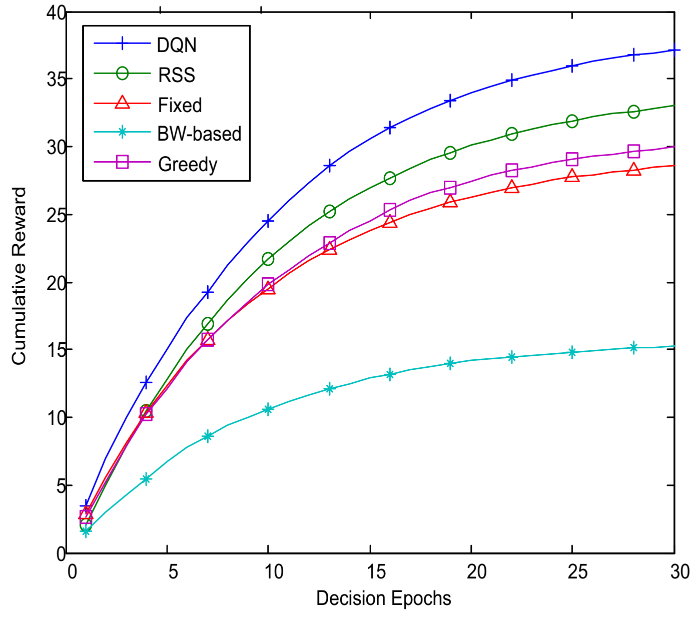

Fig. 4 Cumulative reward. This figure shows the overall cumulative reward of the system as the handoff decision progresses. In the figure, as the number of decision steps increases, the cumulative return of each strategy gradually increases. Among them, the DQN-based handoff algorithm performs better than other algorithms. And as the number of decision steps increases, the advantages of the algorithm become more apparent

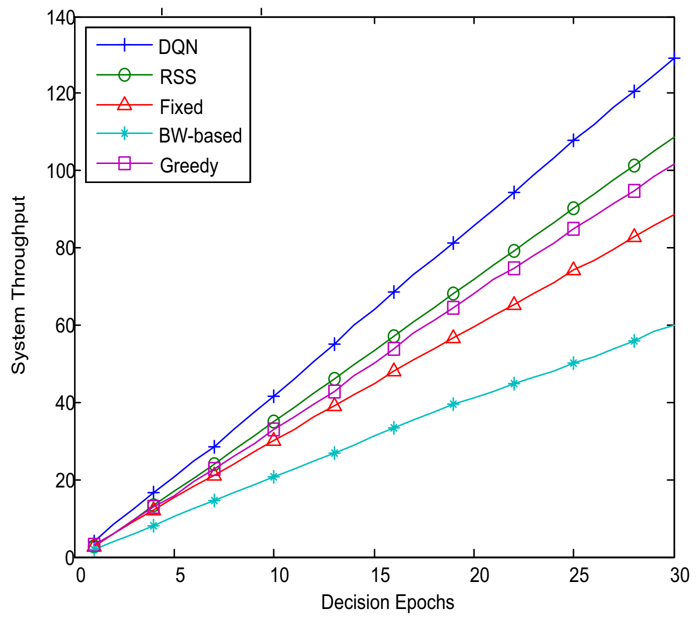

Fig. 5 Throughput. This figure shows the variation in system throughput as the number of decision steps increases. As shown in the figure, the DQN algorithm always achieves the highest throughput performance

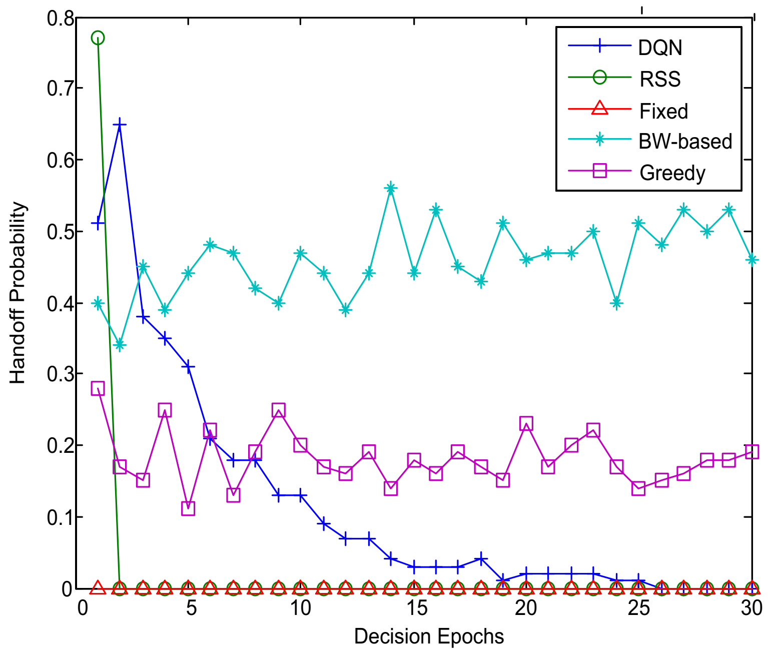

Fig. 6 Handoff probability. We compare the handoff probability of different algorithms with the progress of the handover decision in this figure. In the initial stage of decision-making, the handoff probability number of DQN algorithm is higher than greedy, RSS and fixed algorithm. However, as the decision progresses, the handoff probability of DQN algorithm decreases quickly

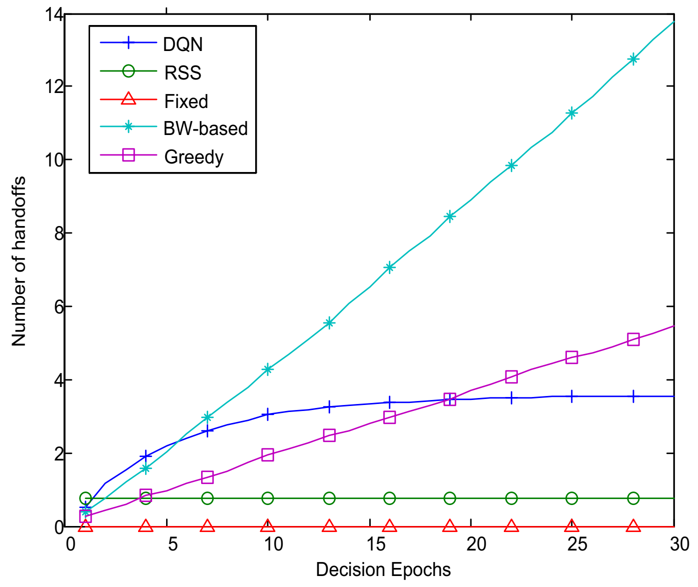

Fig. 7 Number of handoff. In this figure, we compare the handoff probability and the cumulative handoff times of different algorithms with the progress of the handover decision. As shown in the figure, at the beginning of decision-making, the handoff number of DQN algorithm is slightly higher than greedy, RSS and fixed algorithm. However, as the decision progresses, the number of handoffs required by the DQN algorithm is gradually reduced. It is worth noting that after 30 decision times, the DQN algorithm only performs two more handoffs than the RSS algorithm, but the throughput performance is improved by nearly 30% compared to the RSS algorithm

the figure, in the initial stage of decision-making, the handoff number of DQN algorithm is slightly higher than greedy, RSS and fixed algorithm. However, as the decision progresses, the number of handoffs required by the DQN algorithm is gradually reduced. As mentioned earlier, due to the non-handoff strategy, the handoff number of the fixed algorithm is always zero. Since we assume that the location of each user is fixed in the simulation, the channel conditions of the user relative to each base station do not change. Therefore, the RSS algorithm does not perform handoff after finding the base station with the best signal strength and has fewer handoff times. BW-based algorithm only considers the currently available bandwidth and does not consider the signaling overhead caused by handoff, so the number of handoffs is high. Although greedy comprehensively considers the bandwidth, channel conditions and handoff overhead, it does not consider the behavior of other users and the influence of external environment changes and also makes the number of handoffs relatively high. It is worth noting that after 30 decision times, the DQN algorithm only performs two more handoffs than the RSS algorithm, but the throughput performance is improved by nearly 30% compared to the RSS algorithm.

Figures 8 and 9 show the convergence of neural network parameters in the DQN algorithm. Figure 8 shows the relationship between the weight of the connection between the input layer neurons and the hidden layer neurons and the number of trainings. As shown in the figure, after 100,000 trainings, the weight parameters tend to be stable. Further, as shown in Fig. 9 the convergence of the action value function where Q(s,a_1), Q(s,a_2) and Q(s,a_3), respectively, correspond to the action value functions corresponding

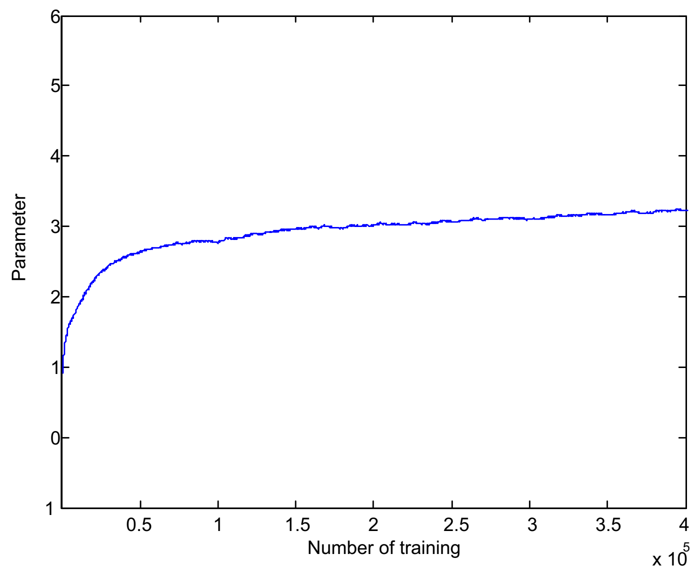

Fig. 8 Parameter convergence. This figure shows the relationship between the weight of the connection between the input layer neurons and the hidden layer neurons and the number of trainings. As shown in the figure, after 100,000 trainings, the weight parameters tend to be stable

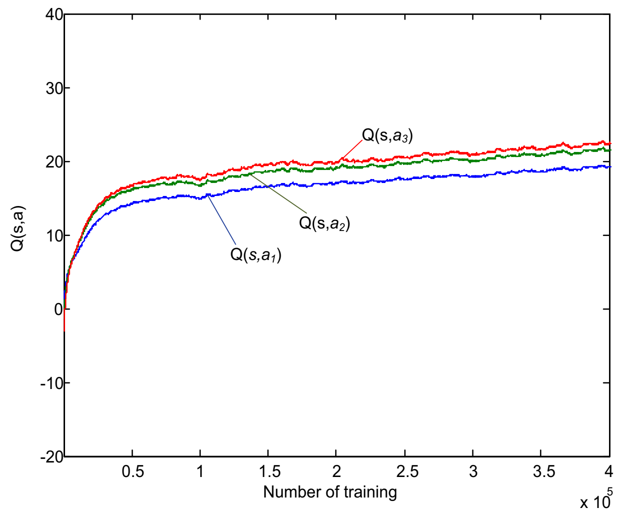

Fig. 9 Value function convergence. This figure gives the convergence of the action value function where Q(s,a_1), Q(s,a_2) and Q(s,a_3), respectively, correspond to the action value functions corresponding to the three different actions by the user in the same state s. As shown in the figure, the convergence of the action value function corresponding to the state s is similar to that in Fig. 8

to the three different actions by the user in the same state s. As shown in the figure, the convergence of the action value function corresponding to the state s is similar to that in Fig. 8. As the number of training increases, it tends to stabilize after about 100,000 training sessions.

As can be seen in conjunction with Figs. 3, 4, 5, 6, 7, 8 and 9, the DQN algorithm can achieve a better cumulative return. In the actual network, by increasing the number of handoff times, the system can achieve higher throughput. It is worth noting that although the algorithm is an online learning algorithm, the agent can collect training information and train its strategy in the background in the gap between every two decision time points, thus accelerating the convergence of the algorithm. In addition, the resulting strategy of training can also be applied to similar scenarios. Therefore, the DQN-based handoff algorithm can efficiently perform and make handoff decisions in accordance with the policies it has learned.

8 Conclusion

In this paper, we propose an intelligent handoff strategy based on reinforcement learning in slice-based mobile networks. Considering the dynamic network state, we model the handoff problem as a Markov decision process (MDP). We thus resort to reinforcement learning theory and propose an intelligent handoff algorithm based on deep reinforcement learning. The effectiveness of our proposed handoff algorithm is validated via simulation experiments. The numerical results show that compared with the typical handoff algorithm in traditional networks, the proposed algorithm can enable users to obtain better transmission performance.

Acknowledgements
The authors acknowledged the anonymous reviewers and editors for their efforts in constructive and generous feedback.

Authors' contributions
Yuansheng Wu contributed to the conception and design of the study. Guanqun Zhao is the main author of the current paper. Dadong Ni and Junyi Du commented on the work. All authors read and approved the final manuscript.

Funding
Not applicable.

Availability of data and materials
Not applicable.

Declarations

Competing interests
The authors declare that they have no competing interests.

Author details
[1] Southwest China Institute of Electronic Technology, Chengdu 610036, China. [2] Huiqing Technology Ltd, Beijing 101500, China.

Received: 22 December 2020 Accepted: 4 March 2021
Published online: 19 March 2021

References

1. W. Guan et al., A service-oriented deployment policy of end-to-end network slicing based on complex network theory. IEEE Access **6**, 19691–19701 (2018)
2. J. Ordonez Lucena, P. Ameigeiras, D. Lopez, J.J. Ramos-Munoz, J. Lorca, J. Folgueira, Network slicing for 5G with SDN/NFV: concepts, architectures, and challenges. IEEE Commun. Mag. **55**(5), 80–87 (2017)
3. H. Wei, Z. Zhang, B. Fan, Network slice access selection scheme in 5G, in *2017 IEEE 2nd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)* (IEEE, 2017)
4. M.R. Sama et al., Service-based slice selection function for 5G, in *2016 IEEE Global communications conference (GLOBECOM)* (IEEE, 2016)
5. 3GPP, TS 38.300, *NR and NG-RAN Overall Description. Stage 2 (Release 15)* (2018)
6. R. Sutton, A. Barto. *Reinforcement Learning: An Introduction* (MIT Press, Cambridge, 1998)
7. M. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming* (Wiley and Sons, Hoboken, 1994)
8. A.H. Arani, M.J. Omidi, A. Mehbodniya, et al., A handoff algorithm based on estimated load for dense green 5G networks, in *2015 IEEE Global Communications Conference (GLOBECOM)* (2015), pp. 1–7

9. G. Araniti, J. Cosmas, A. Iera, et al., Energy efficient handover algorithm for green radio networks, in *2014 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting* (2014), pp. 1–6
10. Y. Sun et al., The SMART handoff policy for millimeter wave heterogeneous cellular networks. IEEE Trans. Mob. Comput. **17**(6), 1456–1468 (2018)
11. H. Tabrizi, G. Farhadi, J. Cioffi, Dynamic handoff decision in heterogeneous wireless systems: Q-learning approach, in *2012 IEEE International Conference on Communications (ICC)* (IEEE, 2012)
12. L. Chen, H. Li, An MDP-based vertical handoff decision algorithm for heterogeneous wireless networks, in *2016 IEEE Wireless Communications and Networking Conference (WCNC)* (IEEE, 2016)
13. E. Stevens-Navarro, Y. Lin, V.W.S. Wong, An MDP-based vertical handoff decision algorithm for heterogeneous wireless networks. IEEE Trans. Veh. Technol. **57**(2), 1243–1254 (2008)
14. Y. Sun, W. Jiang, G. Feng, P. Valente Klaine, L. Zhang, M.A. Imran, Y.-C. Liang, Efficient handover mechanism for radio access network slicing by exploiting distributed learning. IEEE Trans. Netw. Serv. Manag. **17**(4), 2620–2633 (2020). https://doi.org/10.1109/TNSM.2020.3031079
15. Y. Liu, G. Feng, Y. Sun, S. Qin, Y.-C. Liang, Device association for RAN slicing based on hybrid federated deep reinforcement learning. IEEE Trans. Veh. Technol. **69**(12), 15731–15745 (2020). https://doi.org/10.1109/TVT.2020.3033035
16. Y. Sun, S. Qin, G. Feng, L. Zhang, M.A. Imran, Service provisioning framework for RAN slicing: user admissibility, slice association and bandwidth allocation. IEEE Trans. Mob. Comput. (2020). https://doi.org/10.1109/TMC.2020.3000657
17. G. Zhao, S. Qin, G. Feng, Y. Sun, Network slice selection in softwarization-based mobile networks. Trans. Emerg. Telecommun. Technol. **31**(1), e3617 (2020). https://doi.org/10.1002/ett.3617
18. 3GPP TS 36.331, *E-UTRA Radio Resource Control (RRC); Protocol specification (Release 9)* (2018)

Publisher's Note