


Research Article

Factorization Q-Learning Initialization for Parameter Optimization in Cellular Networks

Bosen Zeng ^{1,2}, Yong Zhong,^{1,2} Xianhua Niu,^{3,4} and Ji Tang⁵

¹Chengdu Institute of Computer Application, Chinese Academy of Sciences, Chengdu 610041, China

²University of Chinese Academy of Sciences, Beijing 100049, China

³School of Computer and Software Engineering, Xihua University, Chengdu 610039, China

⁴National Key Laboratory of Science and Technology on Communications, University of Electronic Science and Technology of China, Chengdu 611731, China

⁵Sichuan Branch, China United Network Communications Corporation Limited, Chengdu 610041, China

Correspondence should be addressed to Bosen Zeng; zengbosen19@mails.ucas.ac.cn

Received 14 September 2021; Revised 24 May 2022; Accepted 25 July 2022; Published 18 August 2022

Academic Editor: Mohd Dilshad Ansari

Copyright © 2022 Bosen Zeng et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Q-value initialization significantly influences the efficiency of Q-learning. However, there have been no precise rules to choose the initial Q-values as yet correctly, which are usually initialized to a default value. This paper proposes a novel Q-value initialization framework for cellular network applications and factorization Q-learning Initialization (FQI). The proposed method works as an add-on of Q-learning that automatically and efficiently initializes the nonupdated Q-values by utilizing the correlation model of the visited experiences built on factorization machines. In an open-source VoLTE network, FQI was introduced into Q-learning and four improved variants (Dyna Q-learning, $Q(\lambda)$ -learning, double Q-learning, and speedy Q-learning) for performance comparison. The experiment results demonstrate that the factorized algorithms based on FQI substantially outperform the original algorithms, often learning policies that attain 1.5-8 times higher final performance measured by the episode reward and the convergence episodes.

1. Introduction

The complexity and large scale of radio access technologies in future cellular networks impose significant operational challenges [1]. In a cellular network, there is a multitude of tunable parameters in every base station (BS) to maintain and optimize numerous performance indicators. The parameters have a significant impact on the performance of BSs and should be cognitively adapted to the dynamically changing network environments [2]. However, this is by no means an easy task. First, with more advanced features being deployed in the networks, the number of such parameters increases significantly, and the dependencies among these parameters are more intricate [3]. Moreover, the correlation between different parameters and performance indicators is beyond the capability of available analytical models, as cellular networks evolve to be extremely dynamic and complex due to the scale, density, and heterogeneity.

Recently, leveraging reinforcement learning (RL) to obtain the optimal control policy is emerging as a promising solution [4–6], which enables an autonomous agent to learn from their actions and consequences in the interactive environment. Q-learning [7] is a well-known model-free RL algorithm that finds an estimate of the optimal action-value function. For finite state-action problems, it has been shown that Q-learning converges to the optimal action-value function [8]. However, it suffers from slow convergence. The main reason is the combination of the sample-based stochastic approximation and the fact that the Bellman operator propagates information throughout the whole space. Many methods have been proposed to improve and speed up Q-learning, such as reducing the state space [9–11], modifying Q-value update [12–20], or specifying initial Q-values [21–23].

In this paper, we present a novel Q-value initialization framework and factorization Q-learning initialization (FQI)

to enhance the convergence of Q-learning for parameter optimization in cellular networks. We test the proposed framework on Q-learning and four improved variants (Dyna Q-learning [12], $Q(\lambda)$ -learning [13], double Q-learning [14], and speedy Q-learning [15]) in an open-source cellular environment: voice over-LTE (VoLTE) power control. The experimental results demonstrate that the factorized Q-learning and its variants based on FQI outperform the original algorithms by 1.5-8 times on the valid actions and convergence episodes.

The remainder of this article is organized as follows: Section 2 reviews previous works and relates them to the current research. In Section 3, we briefly describe the cellular network model and formulate the optimization problem. Section 4 introduces the preliminaries of Q-learning. Section 5 presents the FQI framework, its main design motivation, and the algorithm that we use to build it. Section 6 conducts extensive experiments to demonstrate the effectiveness of the proposed framework in an open-source simulated VoLTE network. Finally, the conclusion of the entire work, along with the further research possibilities in the area, have been documented in Section 6.

2. Related Work and Contributions

Q-learning has been widely applied in cellular networks as a tool for parameter optimization problems [24], such as backhaul optimization [25], handover optimization [26], resource optimization [27], and power control [28]. In order to improve and speed up Q-learning, many works have been studied, which involve three aspects: reducing the state space, improving Q-values update, and specifying initial Q-values.

The first type of method is mainly to reduce the searchable size of the state space. A hierarchical approach was proposed to decompose the RL problem into subproblems, where solving each of them will be more powerful than solving the entire problem [9]. The Kanerva coding approach was proposed to reduce the number of states of Q-learning for TCP congestion control [10]. The work in [11] relaxed the constraint of action space for 5G caching to reduce the learning complexity of Q-learning from exponential space size to linear space size.

The second type of method is mainly to modify the Q-value updating method. Dyna Q-learning [12] was proposed to assign each state-action pair a bonus inversely proportionate to the number of times the pair has been visited. $Q(\lambda)$ -learning [13] was proposed to extend Q-learning to eligibility traces, which combines Q-learning with the TD(λ) return estimation process. Double Q-learning [14] was proposed to overcome the overestimation problem by applying double estimators to Q-learning. Speedy Q-learning [15] was proposed to address the problem of slow convergence in the standard form of the Q-learning algorithm. A faster Q-table initialized method was proposed that does not only update the Q-value of a single state-action pair but adds estimates for the cost function for all other possible actions of the current state for the first visiting one state [16]. A matrix-gain approach was designed to accelerate the con-

vergence of Q-learning by optimizing its asymptotic variance [17]. A linear function approximation to update Q-values for 5G caching was used to offer faster convergence and reduce the complexity of Q-learning [18]. An acceleration scheme for Q-learning was proposed by incorporating the historical iterates of the Q-function [19]. A new Q-value updating mechanism was used, in which the Q value of the similar state-action pairs are updated synchronously [20].

The last type of method mainly appropriately specifies the initialization Q-value. It has been shown that the initial Q-values have a significant influence on the efficiency of RL for goal-directed tasks [21]. A method in which the Q-table is initialized to some maximum value and carefully lowered towards the empirical estimates was proposed [22]. A neural network-based Q-learning algorithm was proposed by appropriately specifying initial Q-values [23]. Obviously, a good Q-value initialization method can further strengthen the two types above methods to improve Q-learning performance further. Nevertheless, there have been no precise rules for choosing the initial Q-values as yet correctly, and Q-values are usually initialized to 0 or a random value.

To enhance the convergence of Q-learning by initializing Q-values automatically and efficiently for parameter optimization in cellular networks, this paper makes the following specific contributions:

- (1) Formulate Q-value initialization for the parameter optimization problem as a collaborative filtering problem that builds the correlation model between the visited experiences
- (2) Propose a novel Q-value initialization framework based on factorization machines, factorization Q-learning initialization (FQI), which continuously predicts Q-values that still default value based on the correlation model built on the visited experiences
- (3) Conduct a set of experiments in an open-source simulated VoLTE network. The results show that the proposed framework significantly improves the performance of Q-learning and four improved variants measured by the valid actions and convergence episodes

3. Preliminaries and Motivation

RL is learning what to do (how to map situations to actions) so as to maximize a numerical reward signal. The agent is not told which actions to take but instead must discover which actions yield the most reward by trying them. We describe the essential ingredients in the RL in this section briefly.

3.1. Markov Decision Processes. In order to formalize the RL problem, Markov decision processes (MDP) formally are used to describe an environment for most RL. An MDP is defined by $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma)$:

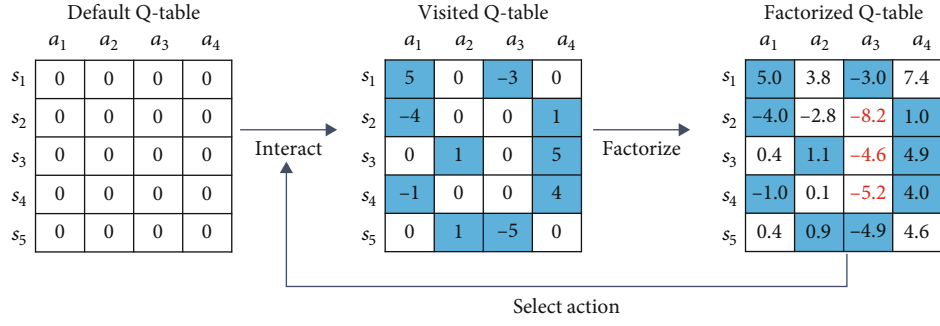


FIGURE 1: A motivating example for factorization Q-learning initialization. The blue entries indicate the visited state-action, that is, the visited experiences. The white entries are the Q-values that have never been updated.

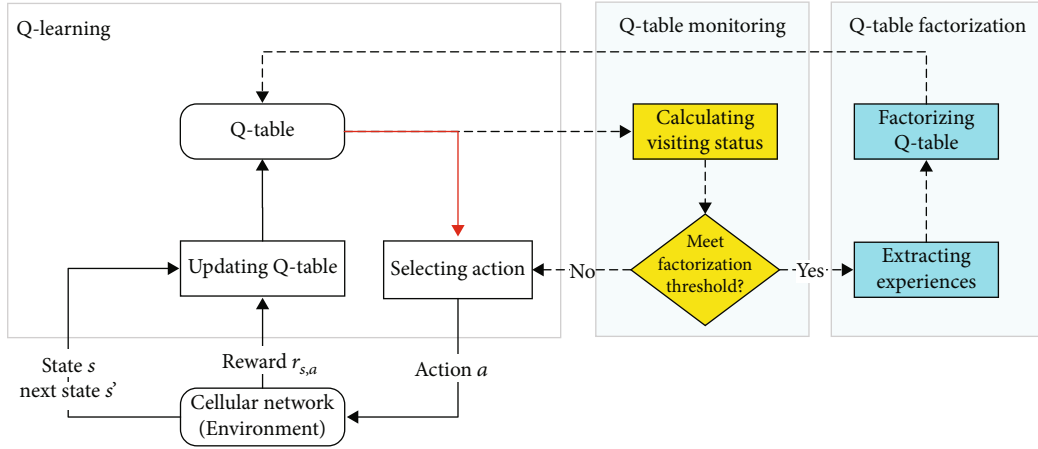


FIGURE 2: Framework overview. The solid red line is the action selection process of the original Q-learning, which is replaced by the dotted lines. If the factorization threshold is reached, Q-table will be factorized to be completed based on factorization machines.

- (i) \mathcal{S} is a finite set of possible states
- (ii) \mathcal{A} is a finite set of possible actions
- (iii) \mathcal{R} is a distribution of reward given (state, action) pair, $\mathcal{R}_s^a = \mathbb{E}[R_{t+1} | S_t = s, A_t = a]$
- (iv) \mathcal{P} is a state transition probability matrix, $\mathcal{P}_{ss'}^a = \mathbb{P}[S_{t+1} = s' | S_t = s, A_t = a]$
- (v) $\gamma \in [0, 1]$ is a discount factor

All states in MDP has Markov property, referring to the fact that the current state captures all relevant information from the history, $P(S_{t+1} | S_t) = P(S_{t+1} | S_1, \dots, S_t)$. A policy π is the behaviour function of the agent from \mathcal{S} to \mathcal{A} that specifies what action to take in each state. The objective of RL is to find the optimal policy π^* that maximizes the expected cumulative discounted reward:

$$\pi^* = \arg \max_{\pi} \mathbb{E} \left[\sum_{t \geq 0} \gamma^t r_t | \pi \right]. \quad (1)$$

3.2. *Q-Learning*. In Q-learning, the Q-value function is to measures how good a particular state-action pair is. The Q

	Feature vector								Target	
	S_1	S_2	S_3	S_4	S_5	a_1	a_2	a_3	a_4	$Q(s, a)$
X_1	1	0	0	0	0	1	0	0	0	5 y_1
X_2	1	0	0	0	0	0	0	1	0	-3 y_2
X_3	0	1	0	0	0	1	0	0	0	-4 y_3
X_4	0	1	0	0	0	0	0	0	1	1 y_4
X_5	0	0	1	0	0	0	1	0	0	1 y_5
X_6	0	0	1	0	0	0	0	0	1	5 y_6
X_7	0	0	0	1	0	1	0	0	0	-1 y_7
X_8	0	0	0	1	0	0	0	0	1	4 y_8
X_9	0	0	0	0	1	0	1	0	0	1 y_9
X_{10}	0	0	0	0	1	0	0	1	0	-5 y_{10}

FIGURE 3: Example for sparse real valued feature vectors x that is created from the visited Q-table in Figure 1. Every row represents a feature vector x_i with its corresponding target y_i . The first five columns (green) represent indicator variables for the visited state; the next four indicator variables (orange) for the visited action. The rightmost column is the target—here is the Q-value.

Input: Factorization threshold μ , latent factor k , max iterations e

- 1: Calculate the percentage of visited states p through Eq. (7) and the increment of the percentage of visited states δ by Eq. (8).
- 2: **If** $\delta \geq \mu$ **then**
- 3: Extract the visited entries $(s, a, Q(s, a))$ from Q table as experiences triples.
- 4: Concatenate the one hot encodings of state s and action a , respectively, in experience Tuples to form feature vector \mathbf{x} and take $Q(s, a)$ as the corresponding target.
- 5: Estimate the model parameters θ at most iterations e by Eq. (12).
- 6: Complete the Q table via Eq. (9).
- 7: Replace the original Q table with the factorized Q table.
- 8: **End if**

ALGORITHM 1: Factorization Q-learning initialization (FQI).

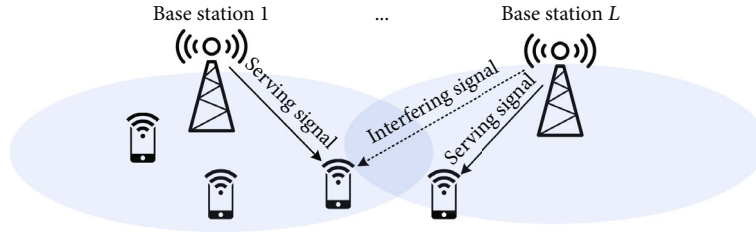


FIGURE 4: Network environment. Performing power control on the signal from the serving BS while coordinating interference from the other BS.

-value function at state s and action a is the expected cumulative reward from taking action a in state s and then following the policy π :

$$Q^\pi(s, a) = \mathbb{E} \left[\sum_{t \geq 0} \gamma^t r_t | s_0 = s, a_0 = a, \pi \right]. \quad (2)$$

The optimal Q-value function Q^* is the maximum expected cumulative reward achievable from a given (state, action) pair:

$$Q^*(s, a) = \max_{\pi} \mathbb{E} \left[\sum_{t \geq 0} \gamma^t r_t | s_0 = s, a_0 = a, \pi \right]. \quad (3)$$

Q^* satisfies the following Bellman equation:

$$Q^*(s, a) = \mathbb{E}_{s' \sim \mathcal{S}} \left[r + \gamma \max_{a'} Q^*(s', a') | s, a \right]. \quad (4)$$

The optimal policy π^* corresponds to taking the best action in any state as specified by Q^* . Q-learning uses the following value iteration algorithm to solve for the optimal policy:

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left[r + \gamma \max_a Q(s', a) - Q(s, a) \right]. \quad (5)$$

$Q(s, a)$ is the cumulated reward expected to be obtained after taking a at state s , which is updated according to the learning rate α , the discount factor γ , and the expected maximum value of the next state s' . Q-learning does converge to the optimal state-action value function if the state-action

pairs are to be explored infinitely often [8]. During the learning phase, the agent needs to decide which action to choose, either to find out more about the environment or to take one step closer to the goal. Techniques for selecting actions in RL are called exploration strategies. The most widely used exploration strategies are ϵ -greedy and Boltzmann.

- (i) ϵ -Greedy. The agent randomly explores with probability ϵ and takes the optimal action most of the time with probability $1 - \epsilon$
- (ii) Boltzmann. Boltzmann is an exponential weighting scheme broadly used for balancing exploration and exploitation. The probability of choosing an exponential function of the empirical mean of the reward of that action is denoted as follows:

$$P(a|s) = \frac{e^{Q(s,a)}}{\sum_a e^{Q(s,a)}}. \quad (6)$$

$P(a|s)$ denotes the probability the agent selects action a in state s .

3.3. Motivation. Q-values are usually set to default values in Q-learning and represented by Q-table, which has a row for each state and a column for each action. The update of the Q-table depends on the interaction with the environment, and the Q-values of the unvisited state-action pairs will remain the default value. In the early stages of learning, the visited state-action pairs are sparse, and most Q-values of the state-action pairs are default values. Therefore, Q-learning often performs extremely poorly in the early stages of learning due to less information about the environment being

forced to act more or less randomly. It is precisely these challenges above that constitute the prime motivations of this article. We are motivated to investigate the following issue: how to utilize the visited experiences to initialize the Q-values that have never been updated automatically and efficiently to bootstrap Q-learning exploration?

Assuming that there are m UEs and n BSs in a cellular network, we derive states, actions, and rewards from constructing the RL process to learn the best control strategy. In the following, we describe them one by one.

- (i) *States*. Let $S = \{s_i = \langle s_1^i, s_2^i, \dots, s_u^i \rangle, s_j^i \in O_j, 1 \leq j \leq u, 1 \leq i \leq |O_1| \times |O_2| \times \dots \times |O_u|\}$ be the set of states of cellular networks with the size $|S|$, where O_j is the discretization value set of the j -th network performance indicator or attribute
- (ii) *Actions*. Let $A = \{a_j = \langle a_1^j, a_2^j, \dots, a_l^j \rangle, a_i^j \in D_i, 1 \leq i \leq u, 1 \leq j \leq |D_1| \times |D_2| \times \dots \times |D_l|\}$ be the set of parameter combinations of cellular networks with the size $|A|$, where D_i is the valid value set of the i -th network parameter
- (iii) *Rewards*. The reward signal $r_{s,a}$ is obtained from the cellular networks after the agent takes a parameter setting a when it is in state s and moves to the next state s'
- (iv) *Q-Value Function*. The state-action value function is denoted $Q(s,a)$. It is the expected cumulated reward when starting in state s and selecting an action a

A motivating example is shown in Figure 1 to comprehend the idea of FQI better. The Q-table example has five states (s_1, s_2, s_3, s_4 , and s_5) and four actions (a_1, a_2, a_3 , and a_4). Initially, the default value of the Q-table is set to 0, which means that the agent has no information about the environment. Some state-action pairs are explored as the agent interacts with the environment, and the corresponding Q-values are updated. The explored experiences are illustrated by blue entries in the visited Q-table. Then, the problem we study is transferred to precisely predicting the Q-values with default values (white entries) based on the visited experiences. Once the unvisited entries are accurately predicted, the risk of random exploration to discover these unknown low Q-values (red numbers) can be reduced according to the factorized Q-table.

4. Factorization Q-Learning Initialization

This section shows the motivation behind this work and describes the proposed framework in detail. The key idea is to continuously capture the correlation between the states and actions from the visited experiences to predict the Q-values that have never been updated to mitigate the possibility of selecting poor parameter settings.

4.1. Proposed Framework. This section presents a novel Q-value initialization framework and factorization Q-learning

TABLE 1: Simulation parameters.

(a)	
Q-learning parameters	Value
Learning rate α	0.2
Discount factor γ	0.995
Number of states $ S $	46656
Number of actions $ A $	16
Initial exploration probability (ϵ -greedy)	1
Minimum exploration probability (ϵ -greedy)	0.05
Exploration decay rate (ϵ -greedy)	0.9995
λ (Q(λ)-learning)	0.5
Number of planning (Dyan Q-learning)	100
(b)	
Factorization parameters	Value
Latent factor k	8
Max iterations e	500
Optimization algorithm	Stochastic gradient descent
Linear regularization λ_1	$1e-10$
Regularization λ_2	$1e-08$
Factorization threshold μ	{0.01,0.02,0.03,0.04,0.05}
(c)	
Network parameters	Value
SINR target	0 dB
UE movement speed	2 km/h

ing initialization (FQI). The principal structure and the main building modules of FQI are shown in Figure 2, which bases on two main modules: (1) Q-table monitoring and (2) Q-table factorization. Q-table monitoring determines whether to factorize the Q-table, and Q-table factorization predicts the Q-values that have never been updated. The original Q-learning algorithm consists of the modules in the left box, and the direction of the solid black and red lines is the workflow. FQI (dotted lines) replaces the original action selection process (solid red line) with Q-table monitoring and Q-table factorization. Each is depicted in green and yellow blocks, respectively, and described in the subsequent subsections.

4.1.1. Q-Table Monitoring. To determine when to factorize Q-table, the percentage of visited states p will be calculated after each interaction with the environment:

$$p = \frac{\sum_{i=1}^{|S|} J \left(\left(\sum_{j=1}^{|A|} I_{i,j} \right) > 0 \right)}{|S|}, \quad (7)$$

where $I_{i,j}$ is the state-action indicator that is equal to 1 if action a_j at state s_i has been visited and 0 otherwise. $J(\cdot)$ is

TABLE 2: Comparison of performance.

Algorithms	Factorization	Convergence episodes		Average episode reward	
		ϵ -Greedy	Boltzmann	ϵ -Greedy	Boltzmann
Q-learning	Original	54.6	367.8	-12.14	-0.590
	Factorized	480.7	774.1	8.514	11.91
Dyna Q-learning [12]	Original	64.9	357.4	-10.26	-1.806
	Factorized	471.5	702.6	9.246	13.16
Q(λ)-learning [13]	Original	57.1	374	-11.90	0.018
	Factorized	464.1	718.6	7.253	10.42
Double Q-learning [14]	Original	56.8	340.6	-12.60	-0.510
	Factorized	479	824.8	9.393	14.25
Speedy Q-learning [15]	Original	68.7	365.6	-8.884	1.151
	Factorized	522	699.1	14.39	14.11

the visited state indicator function that is equal to 1 if an action has been visited at state s_i and is equal to 0 otherwise.

Then, the increment of the percentage of visited states δ is also obtained after every interaction as follows:

$$\delta = p - p^*, \quad (8)$$

where p^* is the percentage of visited states of the last Q-table factorization, and its initial value equals 0. If δ is greater than or equal to the introduced parameter factorization threshold μ , the Q-table will be replaced with the factorized Q-table based on Q-table factorization. Then, the agent selects actions based on the factorized Q-table. Otherwise, the agent selects actions according to the current Q-table.

4.1.2. Q-Table Factorization. The core idea of Q-table factorization is to capture the potential correlation between states and actions from the visited experiences to predict the Q-values that are still default values. Inspired by [29], we build the Q-table's correlation model between states and actions via factorization machine [30]. Factorization machines can model the interactions between different variables using factorized parameters even in problems with huge sparsity combining the advantages of support vector machines with a factorization model. The predicted Q-values of the factorization machine model equation are defined as

$$\hat{y}(\mathbf{x}) = w_0 + \sum_{i=1}^n w_i x_i + \sum_{i=1}^n \sum_{j=i+1}^n x_i x_j \sum_{f=1}^k v_{i,f} v_{j,f}, \quad (9)$$

where x is an n -dimensional feature vector, which is composed of the one-hot encodings of states s and actions a . An example of feature vector from the motivating example is shown in Figure 3. k is a hyperparameter that decides the dimensionality of the factorization, and the model parameters $\Theta = \{w_0, w_1, \dots, w_p, v_{1,1}, \dots, v_{p,k}\}$ are

$$w_0 \in \mathbb{R}, w \in \mathbb{R}^n, V \in \mathbb{R}^{n \times k}. \quad (10)$$

w_0 is the global bias. w_i models the strength of the i -th feature variable. $v_{i,f}$ and $v_{j,f}$ are the f -th value in the i -th feature variable and the j -th feature variable, respectively. $\sum_{f=1}^k v_{i,f} v_{j,f}$ models the interaction between the i -th and j -th variable by factorizing it.

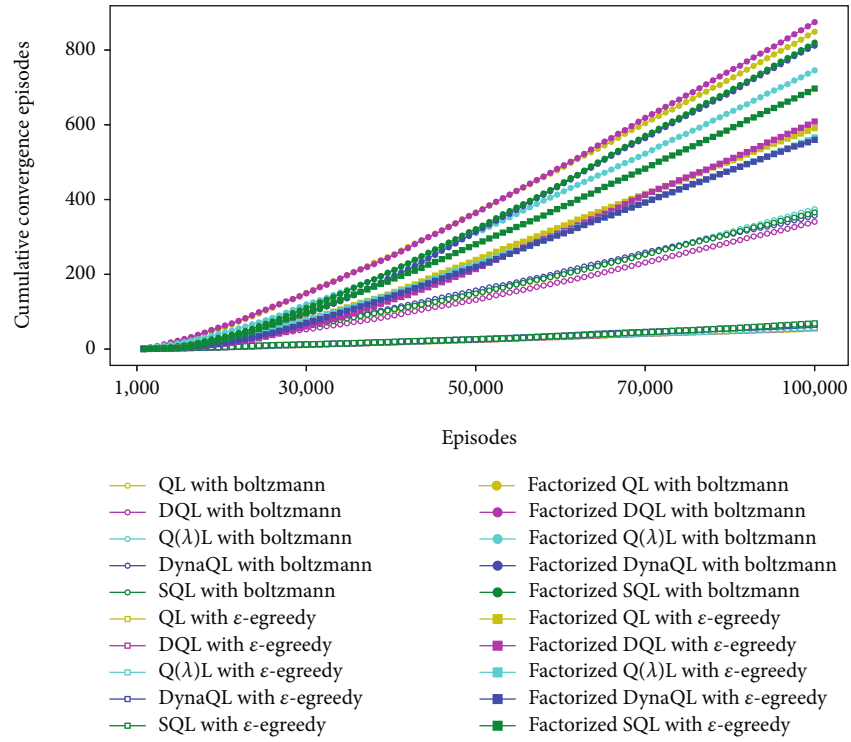
For the sake of simplicity, we denote $\hat{y} = \hat{y}(x)$. The parameters of the factorization machine model Eq. (9) are estimated by solving the following least square minimization problem:

$$\min_{w,v} \sum_{i=1}^p \sum_{j=1}^q I_{i,j} (\hat{y} - y)^2 + \lambda_1 \|w_i\|_2^2 + \lambda_2 \|v_{i,f}\|_2^2, \quad (11)$$

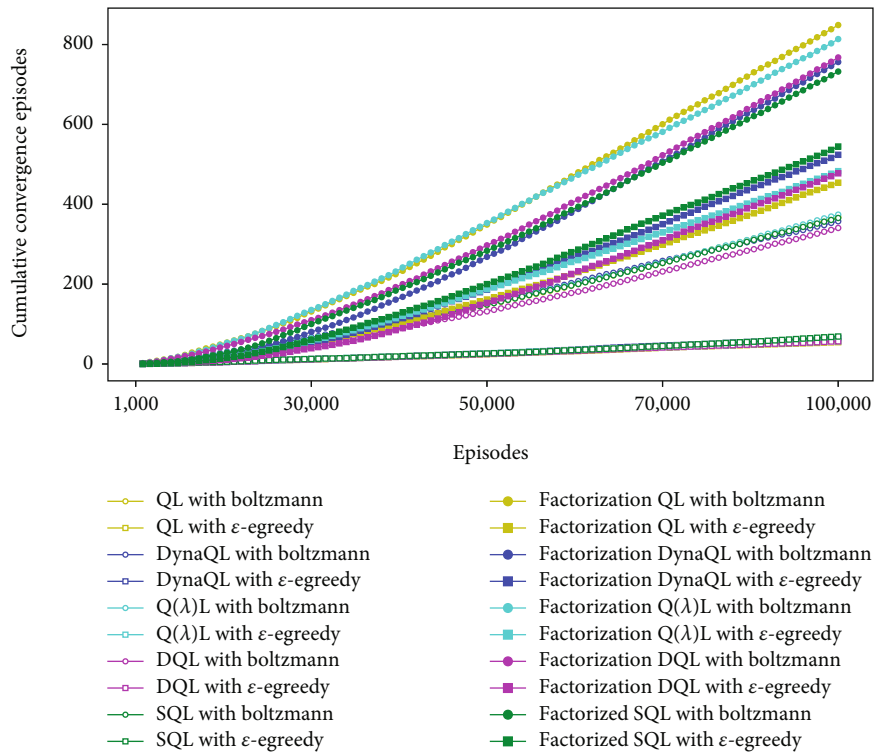
where λ_1 is the linear regularization parameter, and λ_2 is the L2 regularization parameter, which is used to prevent overfitting problems. The gradient of the factorization machine model is

$$\frac{\partial}{\partial \theta} \hat{y}(x) = \begin{cases} 1, & \text{if } \theta \text{ is } w_0, \\ x_i, & \text{if } \theta \text{ is } w_i, \\ x_i \sum_{j=1}^n v_{j,f} x_j - v_{i,f} x_i^2, & \text{if } \theta \text{ is } v_{i,f}. \end{cases} \quad (12)$$

Algorithm 1 sketches how FQI works. First, the value of the factorization threshold μ is set. After every interaction with the environment, the increment of the percentage of visited states δ is calculated (line 1). Q-table factorization is activated if δ is above μ (line 2). Then, the visited entries in Q-table will be extracted as the experiences triples (line 3). After the factorization machine model parameters have been estimated (line 4 and line 5), the Q-table will be completed via Eq. (9) (line 6). The factorized Q-table is used to bootstrap further exploration (line 7). In this way, the Q-learning agent is exposed continually to the factorized Q-table from the early stages of the learning process, thereby mitigating the possibility of selecting poor parameter settings in random exploration to discover these unvisited Q-values.



(a) $\mu = 0.01$



(b) $\mu = 0.02$

FIGURE 5: Continued.

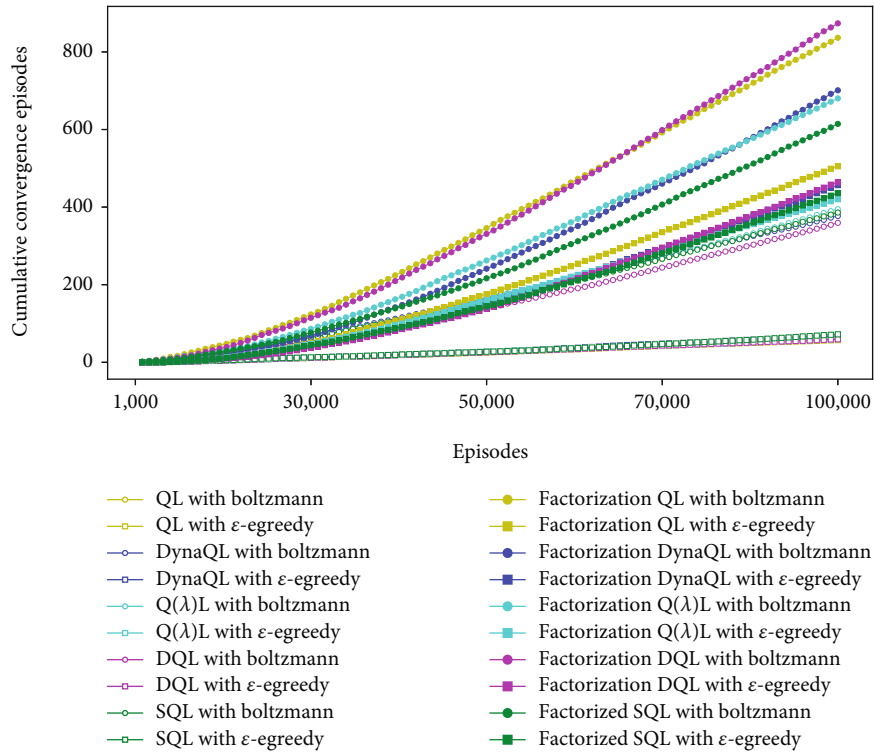
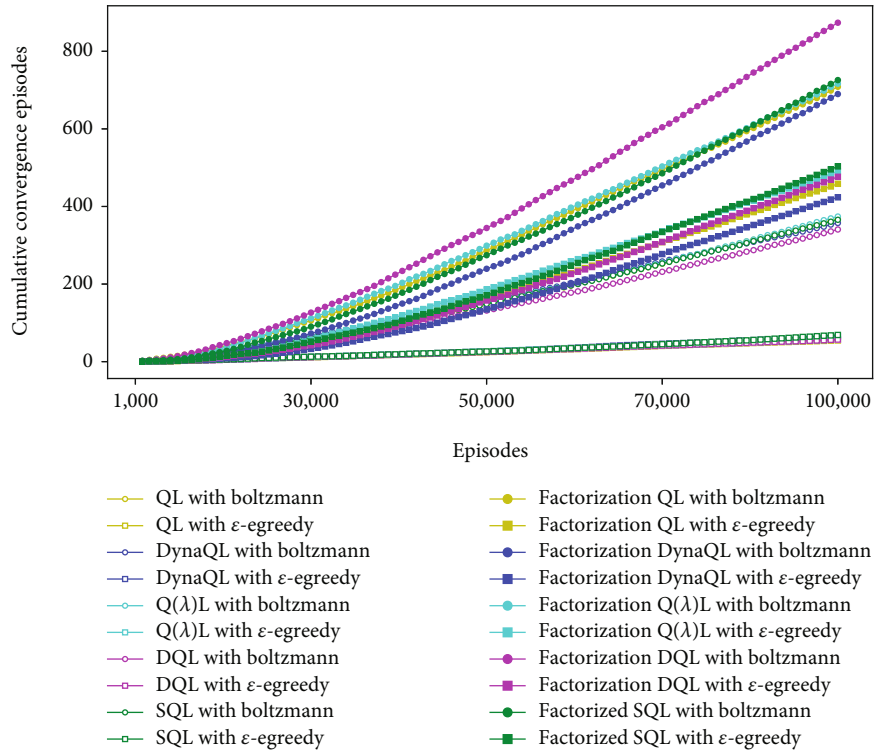


FIGURE 5: Continued.

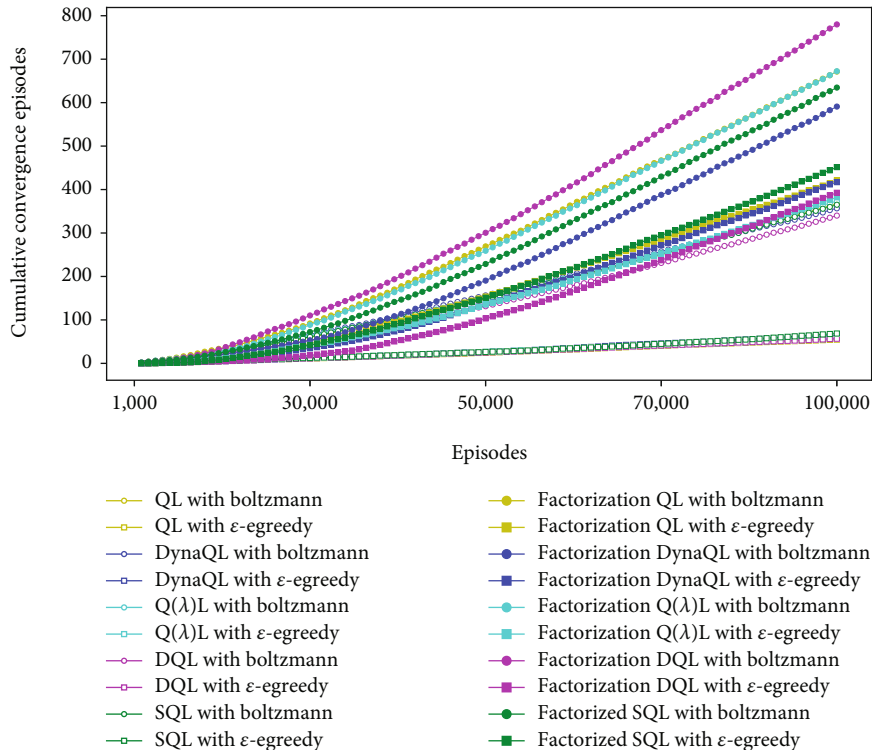


FIGURE 5: Cumulative convergence episodes for the original and factorized algorithms.

4.2. Complexity. The main computation of Algorithm 1 mainly lies in the execution times of factorizing the Q -table. The main computation for each Q -table factorization is to evaluate the loss function Eq. (11) and its gradients against the variables. It can be computed in linear time $O(kn)$ [30], where k is the number of latent factors and n is the dimensionality of the feature vector. Therefore, the main increased computation complexity compared with the standard Q -learning has only complexity $O(kn/\mu)$ where μ is the approximation threshold.

5. Experimental Results

In this section, we evaluate the performance of factorization Q -learning initialization (FQI) in an open-source simulated VoLTE network [31]. We deployed FQI on Q -learning (QL), double Q -learning (DQL), $Q(\lambda)$ -learning ($Q(\lambda)L$), Dyna Q -learning (DynaQ), and speedy Q -learning (SQL) to evaluate performance improvements on the measures of the valid actions and convergence episodes. First, we describe the adopted setup in Section 5.1 before delving into the experimental results in Section 5.2.

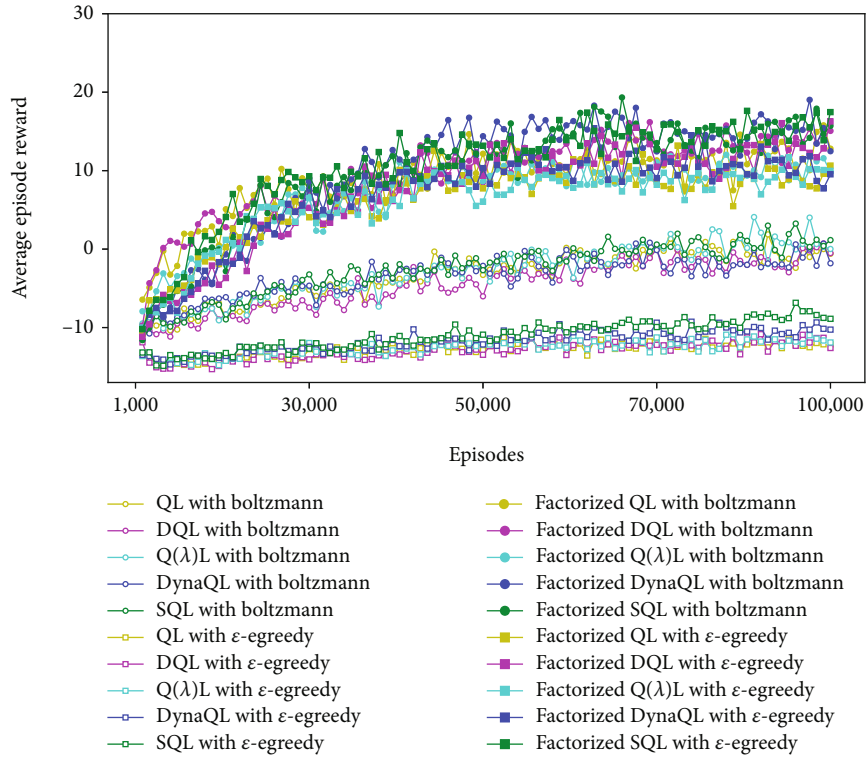
5.1. Simulation Setting. We consider an orthogonal frequency division multiplexing (OFDM) multiaccess downlink cellular network of L base stations (BS)s. There consists of one serving BS and at least one interfering BS. The user equipment (UE) are randomly scattered and mov-

ing in the BS service area engaged in VoLTE shown in Figure 4. Further system details are referred to [31].

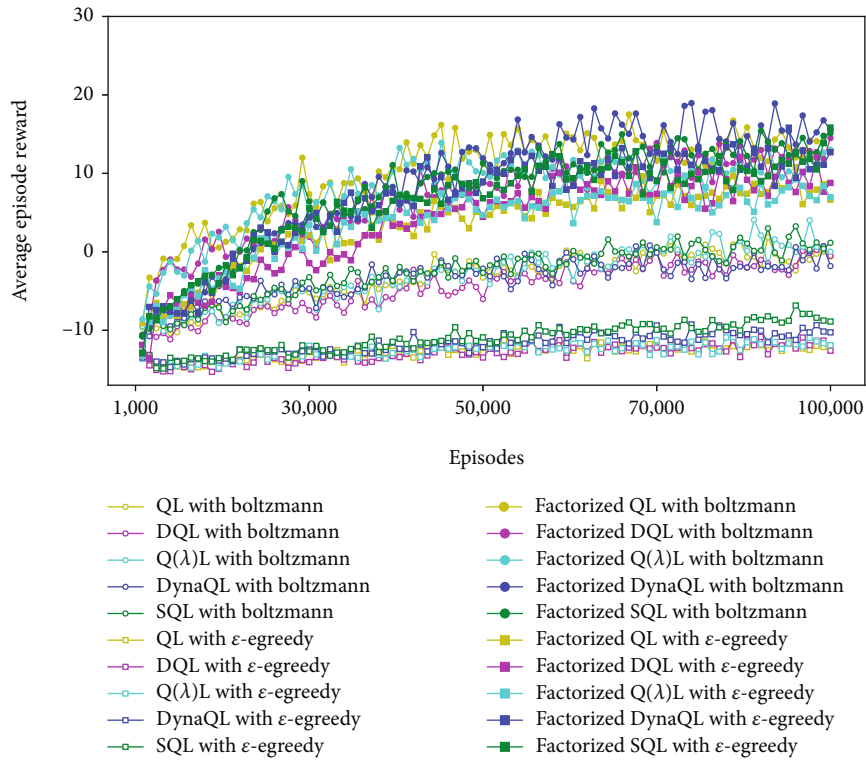
There are two BSs and two UEs, in which the UEs are moving with both log-normal shadow fading and small-scale fading. The target of QL applied in this environment is to jointly optimize the transmit power at the two BSs to make the UEs meet the target SINR. There are 16 parameter settings (actions) to choose from in this environment, which increase or decrease concurrently the transmit powers of the serving and interfering BSs. The environment has 46656 states which are discretized by the positions of the two UEs and the transmit powers of the serving and interfering BSs. ϵ -Greedy and Boltzmann are used to select actions, respectively.

An episode has a duration of 20 timesteps, where the agent selects an action to interact with the environment and receives a reward every timestep. If the SINR target of UEs is fulfilled, the agent receives a reward of 100, and the episode goes on. Otherwise, the episode is terminated prematurely, and the reward is -20. An episode is called converged if the target objective was fulfilled within ten timesteps. The main hyperparameters for QL and FQI in the experiment are shown in Table 1. All algorithms are implemented and available at [32].

5.2. Results and Observations. We ran the original and factorized algorithms based on FQI several times with random seeds. The aggregated results of the convergence episodes and average episode reward across five different factorization

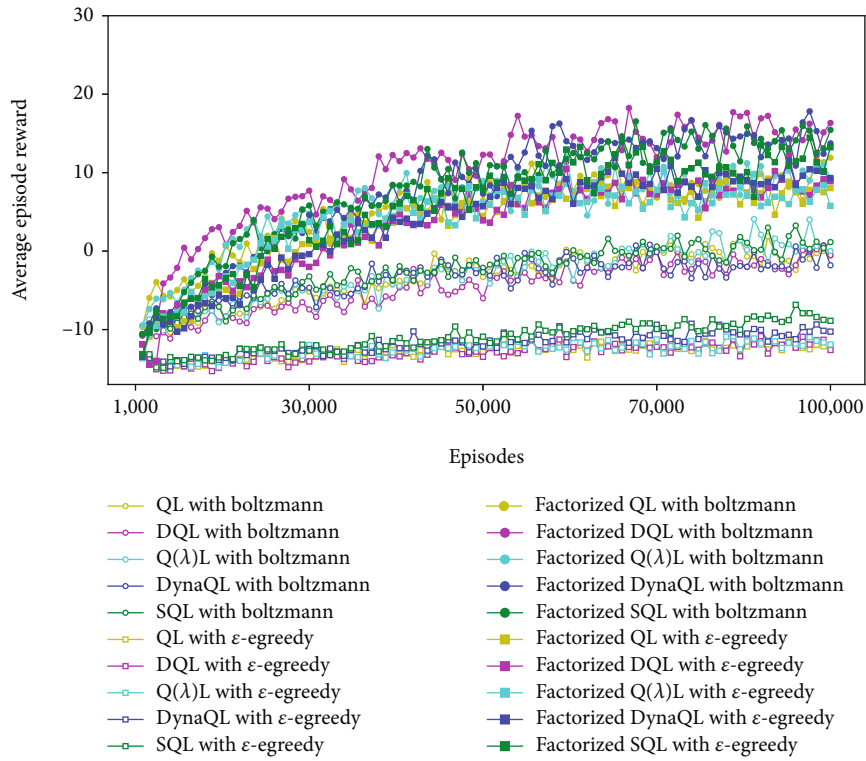


(a) $\mu = 0.01$

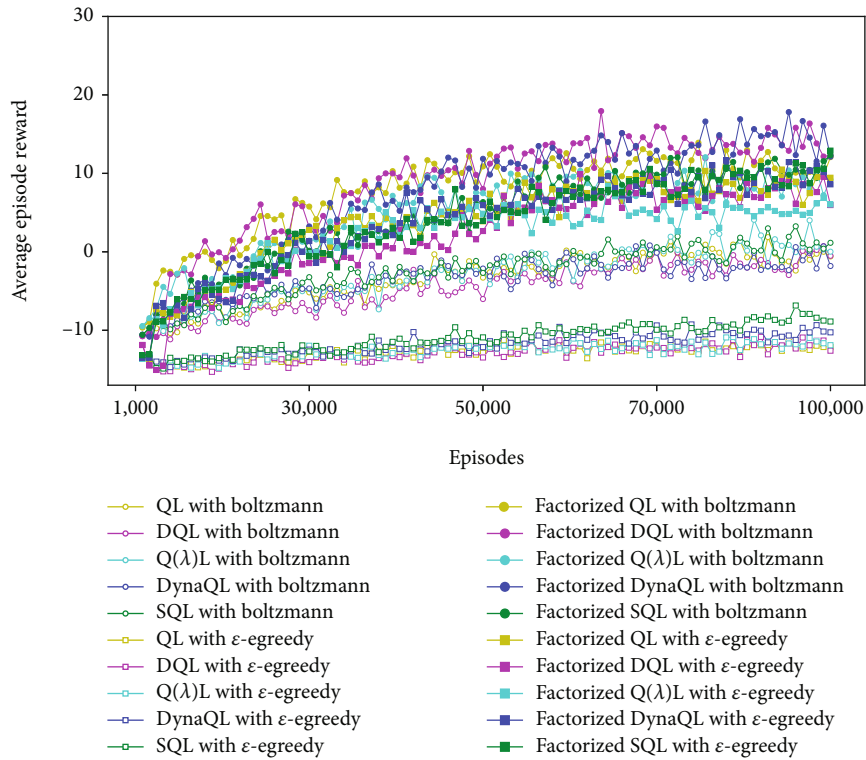


(b) $\mu = 0.02$

FIGURE 6: Continued.



(c) $\mu = 0.03$



(d) $\mu = 0.04$

FIGURE 6: Continued.

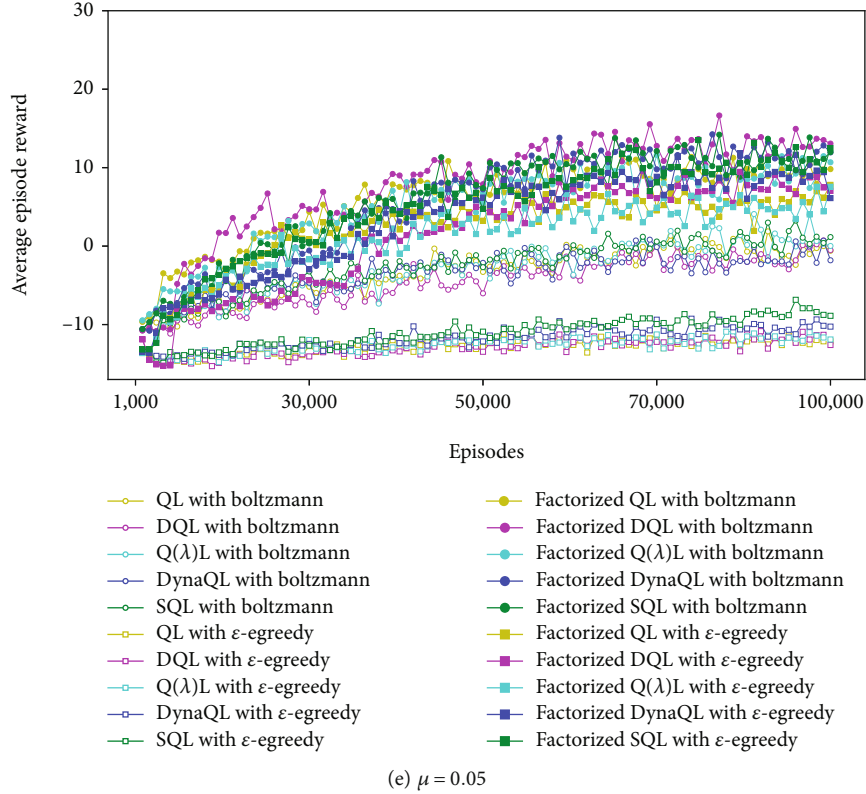


FIGURE 6: Average episode reward for the original and factorized algorithms.

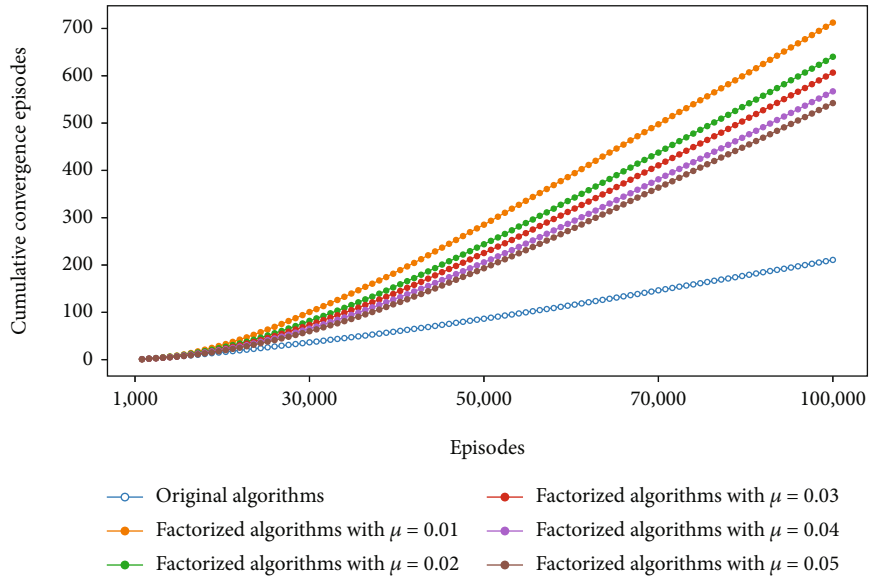
thresholds μ are reported in Table 2. The average episode reward directly reflects the SINR quality of UEs, while the convergence episodes refer to the continuous fulfillment of the SINR target. The larger the value of these two indicators, the power parameter tuning based on DRL algorithms can better allow UEs to meet the SINR target. We observed significant improvements for the factorized algorithms over the original algorithms, with average performance gains of 1.5-8 times higher final results for the original QL, DynaQ, $Q(\lambda)L$, DQL, and SQL. Since exposure to the factorized Q-table from the early stages of the learning process, the factorized algorithms can reach final higher results. Moreover, the factorized algorithms improve the performance under both ϵ -greedy and Boltzmann, and the algorithms with Boltzmann can obtain better performance than ϵ -greedy. This is because Boltzmann selects an action with probability based on Q-values through Eq. (6), rather than blindly accepting any random action such as ϵ -greedy, when it comes time for the agent to explore the environment.

More detailed results are provided in Figures 5 and 6 to distinguish the difference in performance at different factorization thresholds μ . Figure 5 shows the cumulative convergence episodes for the original and factorized algorithms for different factorization thresholds μ . In general, the performance of the factorized algorithms (solid markers) outperforms the original algorithms (blank markers). For the factorized algorithms, adopting Boltzmann (solid circle markers) is better than the performance of ϵ -greedy (solid square markers). However, when the parameter μ is 0.05,

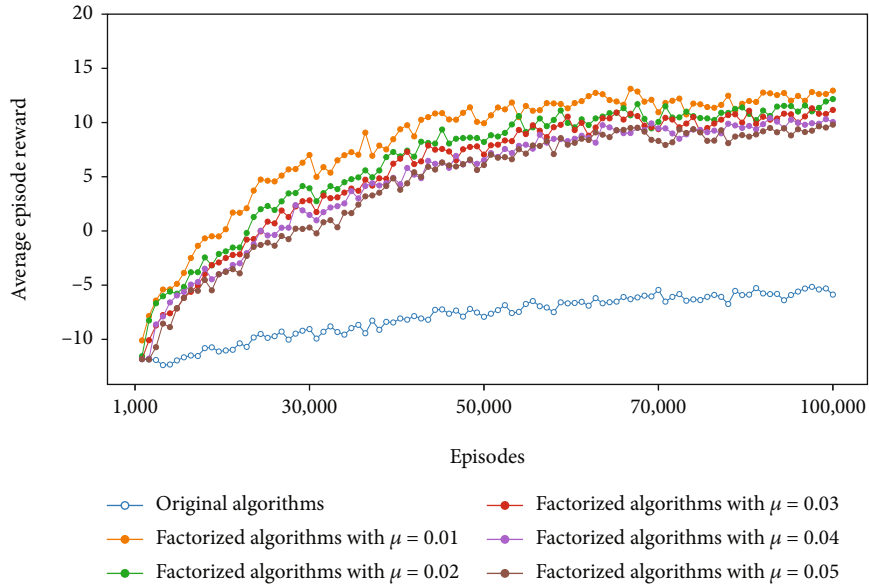
the performance curve of the factorized algorithms with ϵ -greedy partially coincides with the curve of the original algorithms with Boltzmann, indicating that the performance of the two algorithms is similar.

The results of the average episode reward of different μ are demonstrated in Figure 6. The main point to note is that the factorized algorithms (solid markers) are significantly better than the original algorithms, and the original with ϵ -greedy (blank square markers) is the worst. This means that the factorized algorithms can adjust the transmit power of BSs more efficiently than the original algorithms to allow UEs to meet the SINR target more. At the beginning of training, the original and factorized algorithms performed similarly. However, the performances of the factorized algorithms soon become superior to the original algorithms as training progresses. Furthermore, we can observe that the learning curve of the factorized algorithms with $\mu = 0.01$ is steeper than the original algorithms at the initial stage of learning and reaches the plateau faster.

To further study the impact of the factorization threshold μ , the average performance of original algorithms and factorized algorithms with different μ is shown in Figure 7. A key point to note is that the performance of the factorized algorithms improves as the parameter μ decreases. It should also be noted that lower μ leads to faster learning, but even a high threshold captures most of the benefits of Q-table factorization. This is because the smaller the parameter μ , the earlier and more frequently the factorized algorithms perform Q-table factorization to predict the



(a) Cumulative convergence episodes



(b) Average episode reward

FIGURE 7: Performance for the original and factorized algorithms at different μ .

nonupdated Q-values, which bootstraps agent exploration more.

6. Conclusion

In this article, we sought to enhance the convergence of Q-learning for parameter optimization in cellular networks. A Q-value initialization framework based on factorization machines and factorization Q-learning initialization (FQI) was proposed to keep on predicting the nonupdated Q-values based on the visited experiences to bootstrap exploration. We described the details of FQI and showed its effectiveness on Q-learning and its several improved variants, Dyna Q-learning, $Q(\lambda)$ -learning, double Q-learn-

ing, and speedy Q-learning with two widely used exploration strategies, ϵ -greedy, and Boltzmann. The experimental results in an open-source simulated VoLTE network show that the factorized algorithms based on our proposed framework are substantially better than the original algorithms, exceeding their final performance by 1.5-8 times. In addition, earlier and more Q-table factorization can improve the performance of the algorithms due to more guidance for agents to explore, which is more evident in the early stage of learning with too little information to explore efficiently.

A major issue for this work is that it cannot be directly used in the environment of continuous state and action. However, we note that the network parameters in cellular

networks are always discrete. Moreover, the continuous state space of each feature can be discretized, and these continuous values can be modeled by utilizing them as additional features.

7. Future Work

Interesting future work would include research to obtain more insight into the merits of the FQI framework. For instance, in a multiagent Q-learning scenario, FQI can be used to model the state-action interaction between multiple agents to maintain a union Q-table. Possibly, to learn sophisticated feature interactions behind agents' behaviors, replacing factorization machines in the proposed algorithm with DeepFM [33] can yield better results. More analysis on the performance of Q-learning and related algorithms such as zap Q-learning and delay Q-learning is desirable. Furthermore, it would be interesting to see how factorization zap Q-learning, factorization delayed Q-learning, and other extensions of Q-learning perform in practice when applied to FQI.

Data Availability

https://github.com/bszeng/Factorization_Q-learning_Initialization

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (62171387) and the China Postdoctoral Science Foundation (2019M663475).

References

- [1] R. Shafin, L. Liu, V. Chandrasekhar, H. Chen, J. Reed, and J. C. Zhang, "Artificial intelligence-enabled cellular networks: a critical path to beyond-5G and 6G," *IEEE Wireless Communications*, vol. 27, no. 2, pp. 212–217, 2020.
- [2] K. Samdanis and T. Taleb, "The road beyond 5G: a vision and insight of the key technologies," *IEEE Networks*, vol. 34, no. 2, pp. 135–141, 2020.
- [3] M. Agiwal, A. Roy, and N. Saxena, "Next generation 5G wireless networks: a comprehensive survey," *IEEE Communications Surveys and Tutorials*, vol. 18, no. 3, pp. 1617–1655, 2016.
- [4] Y. Chen, Y. Liu, M. Zeng et al., "Reinforcement learning meets wireless networks: a layering perspective," *IEEE Internet of Things Journal*, vol. 8, no. 1, pp. 85–111, 2021.
- [5] N. Kaur and A. Mittal, "CADxReport: chest x-ray report generation using co-attention mechanism and reinforcement learning," *Computers in Biology and Medicine*, vol. 145, article 105498, 2022.
- [6] W. Jin, S. Dong, C. Yu, and Q. Luo, "A data-driven hybrid ensemble AI model for COVID-19 infection forecast using multiple neural networks and reinforced learning," *Computers in Biology and Medicine*, vol. 146, article 105560, 2022.
- [7] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3-4, pp. 279–292, 1992.
- [8] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, MIT Press, Cambridge, MA, USA, 1998.
- [9] T. G. Dietterich, "Hierarchical reinforcement learning with the MAXQ value function decomposition," *Journal of Artificial Intelligence Research*, vol. 15, no. 1, pp. 227–303, 2000.
- [10] W. Li, F. Zhou, K. R. Chowdhury, and W. Meleis, "QTCP: adaptive congestion control with reinforcement learning," *IEEE Transactions on Network Science and Engineering*, vol. 6, no. 3, pp. 445–458, 2019.
- [11] P. Lin, Q. Song, J. Song, A. Jamalipour, and F. R. Yu, "Cooperative caching and transmission in CoMP-integrated cellular networks using reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 5, pp. 5508–5520, 2020.
- [12] R. S. Sutton, "Integrated architecture for learning, planning, and reacting based on approximating dynamic programming," in *Machine learning proceedings 1990*, pp. 216–224, Austin, Texas, 1990.
- [13] J. Peng and R. J. Williams, "Incremental multi-step Q-learning," *Machine Learning*, vol. 22, no. 1-3, pp. 283–290, 1996.
- [14] H. V. Hasselt, "Double Q-learning," *Advances in Neural Information Processing Systems*, vol. 23, 2010.
- [15] M. G. Azar, R. Muons, M. Ghavamzadeh, and H. J. Kappen, *Speedy Q-Learning*, Radboud University Nijmegen, 2011.
- [16] M. Simsek, A. Czylik, A. Galindo-Serrano, and L. Giupponi, "Improved decentralized Q-learning algorithm for interference reduction in LTE-femtocells," in *2011 Wireless Advanced*, pp. 138–143, London, UK, 2011.
- [17] A. M. Devraj and S. P. Meyn, "Zap Q-learning," *Advances in Neural Information Processing Systems*, vol. 30, pp. 2235–2244, 2017.
- [18] A. Sadeghi, F. Sheikholeslami, and G. B. Giannakis, "Optimal and scalable caching for 5G using reinforcement learning of space-time popularities," *IEEE Journal on Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 180–190, 2018.
- [19] B. W. Weng, L. Zhao, H. Q. Xiong, and W. Zhang, "Momentum-based accelerated Q-learning," <https://arxiv.org/abs/1910.11673>.
- [20] W. Liao, X. Wei, and J. Lai, "Smooth Q-learning: accelerate convergence of Q-learning using similarity," <https://arxiv.org/abs/2106.01134>.
- [21] E. Wiewiora, "Potential-based shaping and Q-value initialization are equivalent," *Journal of Artificial Intelligence Research*, vol. 19, no. 1, pp. 205–208, 2003.
- [22] A. L. Strehl, L. Li, E. Wiewiora, J. Langford, and M. L. Littman, "PAC model-free reinforcement learning," in *Proceedings of the 23rd international conference on Machine learning*, pp. 881–888, Pittsburgh, Pennsylvania, USA, 2006.
- [23] Y. Song, Y. B. Li, C. H. Li, and G. F. Zhang, "An efficient initialization approach of Q-learning for mobile robots," *International Journal of Control, Automation and Systems*, vol. 10, no. 1, pp. 166–172, 2012.
- [24] P. V. Klaine, M. A. Imran, O. Onireti, and R. D. Souza, "A survey of machine learning techniques applied to self-organizing cellular networks," *IEEE Communications Surveys and Tutorials*, vol. 19, no. 4, pp. 2392–2431, 2017.
- [25] M. Jaber, M. A. Imran, R. Tafazolli, and A. Tukmanov, "A multiple attribute user-centric backhaul provisioning scheme using distributed SON," in *2016 IEEE global communications*

- conference (GLOBECOM), pp. 1–6, Washington, DC, USA, 2016.
- [26] S. S. Mwanje, L. C. Schmelz, and A. Mitschele-Thiel, “Cognitive cellular networks: a Q-learning framework for self-organizing networks,” *IEEE Transactions on Network Science and Engineering*, vol. 13, no. 1, pp. 85–98, 2016.
- [27] M. Miozzo, L. Giupponi, M. Rossi, and P. Dini, “Switch-on/off policies for energy harvesting small cells through distributed Q-learning,” in *2017 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, pp. 1–6, San Francisco, CA, USA, 2017.
- [28] F. B. Mismar, J. Choi, and B. L. Evans, “A framework for automated cellular network tuning with reinforcement learning,” *IEEE Transactions on Communications*, vol. 67, no. 10, pp. 7152–7167, 2019.
- [29] B. S. Zeng, Y. Zhong, and X. H. Niu, “A data-driven performance prediction approach for cellular network parameter setting via factorization machine,” in *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*, pp. 1322–1327, Singapore, Singapore, 2020.
- [30] S. Rendle, “Factorization machines,” in *2010 IEEE International conference on data mining*, pp. 995–1000, Sydney, NSW, Australia, 2010.
- [31] F. B. Mismar, B. L. Evans, and A. Alkhateeb, “Deep reinforcement learning for 5G networks: joint beamforming, power control, and interference coordination,” *IEEE Transactions on Communications*, vol. 68, no. 3, pp. 1581–1592, 2020.
- [32] B. S. Zeng, “Source code,” 2022, https://github.com/bszeng/Factorization_Q-learning_Initialization.
- [33] H. F. Guo, R. M. Tang, Y. M. Ye, Z. G. Li, and X. Q. He, “Deepfm: a factorization-machine based neural network for ctr prediction,” <https://arxiv.org/abs/1703.04247>.