Learning Automata Based Q-learning for Content Placement in Cooperative Caching

Zhong Yang, *Student Member, IEEE,* Yuanwei Liu, *Senior Member, IEEE,*
Yue Chen, *Senior Member, IEEE,* Lei Jiao, *Senior Member, IEEE*

***Abstract*—An optimization problem of content placement in *cooperative caching* is formulated, with the aim of maximizing the sum mean opinion score (MOS) of mobile users. Firstly, as user mobility and content popularity have significant impacts on the user experience, a recurrent neural network (RNN) is invoked for *user mobility prediction* and *content popularity prediction*. More particularly, practical data collected from GPS-tracker app on smartphones is tackled to test the accuracy of user mobility prediction. Then, based on the predicted mobile users' positions and content popularity, a learning automata based Q-learning (LAQL) algorithm for cooperative caching is proposed, in which learning automata (LA) is invoked for Q-learning to obtain an optimal action selection in a random and stationary environment. It is proven that the LA based action selection scheme is capable of enabling every state to select the optimal action with arbitrary high probability if Q-learning is able to converge to the optimal Q value eventually. In the LAQL algorithm, a central processor acts as the intelligent agent, which allocate contents to BSs according to the reward or penalty from the feedback of the BSs and users, iteratively. To characterize the performance of the proposed LAQL algorithms, sum MOS of users is applied to define the reward function. Extensive simulation results reveal that: 1) the prediction error of RNNs based algorithm lessen with the increase of iterations and nodes; 2) the proposed LAQL achieves significant performance improvement against traditional Q-learning algorithm; and 3) the cooperative caching scheme is capable of outperforming non-cooperative caching and random caching of 3% and 4%, respectively.**

***Index Terms*—Learning automata based Q-learning, quality of experience (QoE), wireless cooperative caching, user mobility prediction, content popularity prediction.**

I. Introduction

It is expected that the first fifth generation (5G)-network based solutions will be commercially launched by 2020 [2]. According to a recent report from Cisco [3], mobile data traffic has grown 18-fold over the past 5 years and will grow at a compound annual growth rate (CAGR) of 47 percent from 2016 to 2021. More particularly, mobile video traffic accounted for 60 percent of total mobile data traffic in 2016, and over three-fourths (78 percent) of the worlds mobile data traffic will be video by 2021. In the vast area of 5G wireless systems, an extraordinary variety of technological innovations was included to support a large number of devices over limited spectrum resources [4–7]. Investigation presented in [8] shows that backhaul consumes up to 50 percent of the power in a wireless access network. Proactive caching at the base stations (BSs) is a promising approach to dealing with this problem, by introducing caching capabilities at BSs and then pre-fetching contents during off-peak hours before being requested locally by users (see [9], and references therein).

Designing proactive caching policy for each BS independently (e.g., each BS caching the most popular contents) may result in insufficient utilization of caching resources (see [10], and references therein). Cooperative caching increases content diversity in networks by exchanging availability information, thus further exploits the limited storage capacity and achieves more efficient wireless resource utilization (see [11–14], and references therein). The key idea of cooperative caching is to comprehensively utilize the caching capacity of BSs to store specific contents, thus improving QoE of all users in networks. However, the challenges of cooperative caching is how to place contents in different BSs, according to the caching capacity of the BSs, the content popularity and the user mobility. In [11], multicast-aware cooperative caching is developed using maximum distance separable (MDS) codes for minimizing the long-term average backhaul load. In [12], delay-optimal cooperative edge caching is explored, where a greedy content placement algorithm is proposed for reducing the average file transmission delay.

Based on aforementioned advantages of wireless caching, research efforts have been dedicated to caching solutions in wireless communication systems. Aiming to minimize the average downloading latency, the caching problem was formulated as an integer-linear programming problem in [15], which are solved by subgradient method. In [16], multi-hop cooperative caching is proposed for improving the success probability of content sharing [17]. The emerging machine learning paradigm, owing to its broader impact, has profoundly transformed our society, and it has been successfully applied in many applications [18], including playing Go games [19], playing atari [20], human-level control [21]. Wireless data traffic contains strong correlations and statistical features in various dimensions, thereby the application of machine learning in wireless cooperative caching networks optimization presents a novel perspective. In this paper, we formulate a long-term optimization problem of content placement in cooperative caching. To predict user mobility and content popularity, we adopt a recurrent neural network (RNN) to predict user mobility and content popularity in the near future time slot. After obtaining user mobility and content popularity, we propose a learning automata based Q-learning (LAQL)

Part of this paper was presented in IEEE Global Communication Conference (GLOBECOM), Abu Dhabi, UAE, Dec. 2018 [1].

Z. Yang, Y. Liu and Y. Chen are with the School of Electronic Engineering and Computer Science, Queen Mary University of London, London E1 4NS, UK. (email:{zhong.yang, yuanwei.liu, yue.chen}@qmul.ac.uk),

L. Jiao is with the Department of Information and Communication Technology, University of Agder, 4879 Grimstad, Norway. (email: lei.jiao@uia.no).

algorithm for content placement.

A. Motivation and Related Works

Sparked by the aforementioned potential benefits, we therefore explore the potential performance enhancement brought by reinforcement learning (RL) for wireless cooperative caching networks. The authors in [22] proposed a context-aware proactive caching algorithm, which learns content popularity online by regularly observing context information of connected users. In [23], content caching is considered in a software-defined hyper-cellular networks (SD-HCN), aiming at minimizing the average content provisioning cost.

RL algorithm is essentially also one of stochastic optimization algorithms. The advantage of RL is that it can learning the reward or penalty from the feedback of the environment without know the transition probabilities of states. There are some research contributions on RL for content caching [24–29]. The authors in [24–26] utilize the deep neural network to approximate the Q-table of RL, i.e., the deep Q-network. However, the neural networks in DQN are unstable and hard to reproduce. In [30], an approach to perform proactive content caching was proposed based on the powerful frameworks of echo state networks (ESNs) and sublinear algorithms, which predicts the content popularity and users' periodic mobility pattern. In [31], ESNs are used to effectively predict each user's content request distribution and its mobility pattern when limited human-centric information such as users' visited locations, requested contents, gender, job, and device type, etc. Q-learning enables learning in an unknown environment as well as overcoming the prohibitive computational requirements, which has been widely used in wireless networks. In [27], the authors model the proactive caching problem as a Markov decision process with the goal of minimizing the long-term average energy cost. The proposed threshold-based proactive caching scheme is proven optimal. However, the system setup ignores the interference between BSs in the cooperative area, which we consider in our scenario. The authors in [28] propose a Q-learning based caching policy without knowing the transition probabilities. But only one base station caching is considered in [28], which may result in insufficient utilization of caching resources according to [10]. In [29], a hyper deep Q-networks (DQNs) is developed for the caching system for a parent caching node and multiple leaf nodes. The above research contributions have laid a solid foundation for caching, but two critical problems are still not solved. The first problem is the user mobility and content popularity prediction. [27] only assumes that the content popularity remains relevant to the user for random lifetimes. [28] agrees that content popularity is important for caching, but it does not provide a sufficient approach to predict content popularity. Another problem is that for RL, the action selection scheme has significant influence on the performance of RL.

B. Contributions and Organization

The aforementioned research contributions considered a static content placement of cooperative caching in the scenario that user mobility and content popularity are static and formulate a static optimization problem [15, 16]. In practical scenario, we can predict user mobility and content popularity according to the collected prior data. In this article, we study the wireless cooperative caching system, where the caching capacity and the backhaul capacity are restricted. It is worth pointing out that the characteristics of cooperative caching make it challenging to apply the RL, because the number of caching states increases exponentially with the number of contents and BSs. With the development of RL and high computing speed of new computer, we design an autonomous agent, that perceives the environment, and the task of the agent is to learn from this non-direct and delayed payoff so as to maximize the cumulative effect of subsequent actions. The agent improves its performance and chooses behavior through learning.

Driven by solving all the aforementioned issues, we present a systematic approach for content cooperative caching. More specifically, we propose a learning automata based Q-learning (LAQL) algorithm for content placement in wireless cooperative caching networks (i.e., we consider "what" and "where" to cache). Instead of complex calculation of the optimal content placement, our designed algorithm enables a natural learning paradigm by state-action interaction with the environment it operates in, which can be used to mimic experienced operators. In our contributions of the Globecom article, we adopted Q-learning for the content placement in cooperative caching. In this article, we further propose a learning automata based Q-learning algorithm to improve the performance of cooperative caching. Further more, we utilize recurrent neural networks (RNNs) to predict the content popularity, while in the Globecom work, the content popularity is assumed to follow a Zipf distribution. Our main contributions are summarized as follows.

1) We propose a content cooperative caching framework to study the QoE of mobile users. We establish the correlation between user mobility, content popularity, and content placement in cooperative caching, by modeling the content placement with user mobility and content popularity. Then, we formulate the sum mean opinion score (MOS) maximization problem subject to the BSs' caching capacity, which is a combinatorial optimization problem. We mathematically prove that the formulated problem is nondeterministic polynomial-time (NP) hard.
2) We invoke recurrent neural networks (RNN) for user mobility prediction and content popularity prediction. We utilize practical trajectory data collected from GPS-tracker app on smartphones to validate the accuracy of user mobility prediction[1].
3) We conceive an LAQL algorithm based solution for content placement in cooperative caching. In contrast to a conventional Q-learning algorithm where ϵ-greedy is utilized, we design an action selection scheme invoking learning automata (LA). Additionally, we prove that the LA based action selection scheme is capable of enabling

[1]The dataset has been shared by authors in Github. It is shown on the websit: https://github.com/swfo/User-Mobility-Dataset.git. Our approach can accommodate other datasets without loss of generality.

every state to select the optimal action with arbitrary high probability if Q-learning is capable of converging to the optimal Q value eventually.

4) We demonstrate that the performance of the proposed LAQL based content cooperative caching with content popularity and user mobility prediction outperforms conventional Q-learning. Meanwhile content cooperative caching is capable of outperforming non-cooperative caching and random caching algorithm in terms of QoE of users.

The rest of the paper is organized as follows. In Section II, the system model for content caching in different BSs is presented. In Section III, the RNN based user mobility and content popularity prediction are investigated. LAQL for content cooperative caching is formulated in Section IV. Simulation results are presented in Section V, before we conclude this work in Section VI. Table I provides a summary of the notations used in this paper.

II. System Model

A. System Description

We consider a circle cooperative area composed of M base stations (BSs) with one transmit antennas and N_u single antenna users (as shown in Fig. 1). The central processor collects relevant information from the system, and acts as a master to control the BS (slaves), and implements a centrally controlled adjustment of caching policies [28]. Denote $\mathcal{M}=\{1,\cdots,M\}$ and $\mathcal{N}_u=\{1,\cdots,N_u\}$ be the BSs set and the user set, respectively. We assume that the content library in the central network consists of F contents denoted by $\mathcal{F}=\{1,\cdots,F\}$, and the size of f-th content is $l(f)$. For simplicity, we assume that each content has the same size 1, i.e., $l(f)=1$. This assumption is easily found by dividing contents into blocks with the same size [32]. The position of user i is denoted as $z_i^u(t)$. The position of BS m is denoted by $z_m^b(t)$. Assume that the BSs have same caching capacity s. To guarantee the limited storage capacity of BSs, the content caching vector of BS m is defined as $\mathbf{X_m}(t)=[x_{1,m}(t),x_{2,m}(t)\cdots,x_{s,m}(t)]$, where $x_{j,m}(t)\in[1,F],\forall j\in[1,s]$. Therefore, the cooperative caching matrix is denoted as:

$$\mathbf{X}(t)=\begin{bmatrix}X_1(t)\\X_2(t)\\\vdots\\X_M(t)\end{bmatrix}=\begin{bmatrix}x_{1,1}(t),x_{2,1}(t)\cdots,x_{s,1}(t)\\x_{1,2}(t),x_{2,2}(t)\cdots,x_{s,2}(t)\\\vdots\\x_{1,M}(t),x_{2,M}(t)\cdots,x_{s,M}(t)\end{bmatrix}. \tag{1}$$

where t is the time index. $\mathbf{X}(t)$ denotes the content placement matrix in time slot t.

We set the position of users randomly, and the position of user i is denoted as $z_i^u(t)=(z_i^{ux}(t),z_i^{uy}(t))$, and the position of BS m is represented by $z_m^b(t)=\left(z_m^{bx}(t),z_m^{by}(t)\right)$. As such, the distance between user i and BS m is given by $\|r_{mi}(t)\|=\left\|z_i^u(t)-z_m^b(t)\right\|$. We consider a wireless communication system with FDMA.

Consider content f in the cooperative region, which is cached in multiple BSs, denoted by $\mathcal{M}_f$. These BSs jointly transmit content f to user i who requests for it. The received signal-interference-noise ratio (SINR) of coherent received signal of user i requesting for content f is given by

$$\mathrm{SINR}_{if}(t)=\frac{\left(\sum\limits_{m\in\{\mathcal{M}_f\}}\sqrt{\rho(t)}\,|h_{mi}(t)|\,r_{mi}(t)^{-\alpha/2}\right)^2}{\mathrm{I}_{if}(t)+BN_0}, \tag{2}$$

where ρ denotes the equally transmit power of all BSs, and $|r_{mi}|^{-\alpha}$ represents a standard distance-dependent power law pathloss attenuation between BS m and user i, where $\alpha\geq 2$ is the pathloss exponent. $h_{mi}(t)$ denotes the small-scale Rayleigh fading between BS m and user i. B is the bandwidth. N_0 is the noise spectral density. $\mathrm{I}_{if}(t)$ is the sum of interfering signal power from BSs that do not cache content f at time slot t, which is given by

$$\mathrm{I}_{if}(t)=\sum_{s\in[1,F],v\neq f}\left(\sum_{n\in\{\mathcal{M}_v\}}\sqrt{\rho(t)}\,|h_{ni}(t)|\,r_{ni}(t)^{-\alpha/2}\right)^2, \tag{3}$$

where $\mathcal{M}_v$ represents all the base stations (BSs) that cache content v. The fist sum function implies that the BSs that cache different contents are non-coherent interference. While the second sum function inside implies that the BSs that cache same content are coherent interference.

The bandwidth of each downlink user is denoted as B, Therefore, based on Shannon's capacity formula, the achievable rate of user i requesting for content f at time t can be expressed as

$$R_{if}(t)=B\log_2\left(1+\mathrm{SINR}_{if}(t)\right). \tag{4}$$

In the cooperative caching scenario, users are moving in a time duration T. The time duration is sliced into time slots. Each time slot is a time interval, which has sufficiently large blocklength. In the beginning of each time slot, each user requests for one content. We assume that users make requests

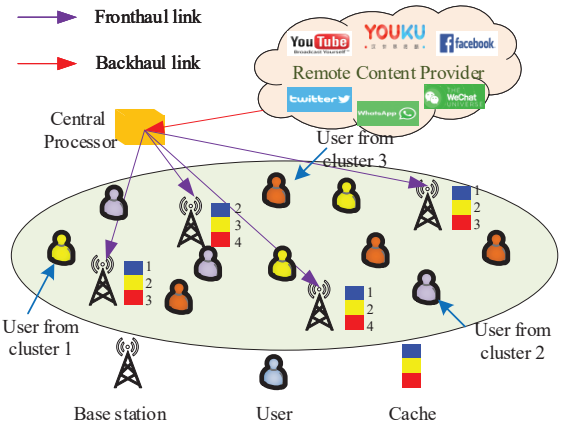


Fig. 1. Schematic of the wireless cooperative caching scenario. The users are served by BSs cooperatively, and contents cached in different BSs (slaves) are placed by cental processor (master).

TABLE I
LIST OF NOTATIONS

Notation	Description	Notation	Description
M	The number of BSs	$R_{m\min}$	The minimum fronthaul capacity of BS m
N_u	The number of users	MOS_{if}	The mean opinion score of user i requesting for content f
F	The number of contents	RTT	The round trip time
$l()$	The size of content	FS	The web page size
z_i^u	The position of user i	MSS	The maximum segment size
z_m^b	The position of BS m	L	The number of slow start cycles with idle periods
s	The caching capacity of BSs	x_{l-1}	The input of layer $l-1$
$\mathbf{X_m}$	The caching vector of BS m	W_{ax}^l	The input weight of layer l
W_{aa}^l	The transmit weight of layer l	W_{ya}^l	The output weight of layer l
r_{mi}	The distance between user i and BS m	b_l^o	The output bias of layer l
ρ	The transmit power of BSs	y^{output}	The output of the network
h_{mi}	The small scale fading of BS m and user i	y^{real}	The real data
σ^2	The power of the additive noise	$Q^{\pi}(s,a)$	The Q table
I_{if}	The sum of interfering signal power	$Q^*(s,a)$	The optimal state-action value function
B	The bandwidth of each downlink user	$u_i(t)$	The times action i has been rewarded when selected
p_{if}	The popularity of content f for user i	$v_i(t)$	The number of times that action i has been selected
$R_{m\max}$	The maximum fronthaul capacity of BS m	$P_a(t)$	The probability distribution of LA

simultaneously. Since users are moving, distances between users and BSs are dynamic. The content popularity is also dynamic in different time slots. Therefore, we firstly predict the user mobility and content popularity in the next time slot. Then, based on the predicted information, we allocate contents to BSs before users actually request for contents.

If the content is not cached in the BS, then the content needs to be fetched from the centre network using fronthaul link. Due to the fronthaul link capacity constraints, the total transmitting rate of each BS should not exceed its fronthaul capacity $R_{m\max}$. Assuming that there are H_m users associated with BS m, whose requested content is not cached, thus

$$\sum_{i=1}^{H_m} B\log_2\left(1+\mathrm{SINR}_{if}(t)\right) \le R_{m\max}. \tag{5}$$

Another constraint is that the transmit rate for each user should not be less than the minimum transmit rate $R_{i\min}$, thus

$$B\log_2\left(1+\mathrm{SINR}_{if}(t)\right) \ge R_{i\min}. \tag{6}$$

B. Quality-of-Experience Model

As in [33], the definition of quality-of-experience (QoE) is "The overall acceptability of an application or service, as perceived subjectively by the end user." The QoE is a users' satisfaction description during the process of interactions between users and services [34]. Mean opinion score (MOS) model is widely adopted for measuring users' QoE of services such as: file downloading, web browsing, and video streaming. The MOS methodology is capable of linking the technical objective function with the subjective user perceived quality [35]. In this article, we concentrate on the web browsing applications. Therefore, based on [35], we define our MOS of user i in time t as

$$\mathrm{MOS}_{if}(t) = -C_1\ln\left(d\left(R_{if}(t)\right)\right) + C_2, \tag{7}$$

where C_1 and C_2 are constants determined by analyzing the experimental results of the web browsing applicants, which are set to be 1.120 and 4.6746, respectively. $\mathrm{MOS}_i(t)$ represents the user perceived quality expressed in real numbers ranging from 1 to 5 (i.e., the score 1 represents extremely low quality, whereas score 5 represents excellent quality. $R_{t,i}$ is achievable sum rate of user i at time t, and $d\left(R_{t,i}\right)$ is the delay time of a user receiving a content. According to [36],

$$\begin{aligned} d\left(R_{if}(t)\right) = {} & 3\mathrm{RTT} + \tfrac{\mathrm{FS}}{R_{if}(t)} + \\ & L\left(\tfrac{\mathrm{MSS}}{R_{if}(t)} + \mathrm{RTT}\right) - \tfrac{2\mathrm{MSS}\left(2^{\mathrm{L}}-1\right)}{R_{if}(t)}, \end{aligned} \tag{8}$$

where $R_{if}(t)$ represents the data rate, RTT is the round trip time, FS is the web page size, and MSS is the maximum segment size. L is the number of slow start cycles with idle periods, $L = \min\left[L_1, L_2\right]$, which is defined as [36]: $L_1 = \log_2\left(\frac{R_{if}(t)\mathrm{RTT}}{\mathrm{MSS}} + 1\right) - 1$, $L_2 = \log_2\left(\frac{\mathrm{FS}}{2\mathrm{MSS}} + 1\right) - 1$, where L_1 denotes the number of cycles that the congestion window takes to reach the bandwidth-delay product and L_2 is the number of slow start cycles before the webpage size is completely transferred. Substituting (8) into (7), we obtain MOS of user i in equation (9):

C. Problem Formulation

We obtain users mobility and content popularity from Section III. Hereafter, the objective is to maximize the overall $\mathrm{MOS}_{tot} = \sum_{t=1}^{T}\sum_{i=1}^{N}\sum_{f=1}^{F} p_{if}(t)\,\mathrm{MOS}_{if}(t)$. The content placement problem in cooperative caching is to determine the cached contents for each BS aiming to maximize the sum MOS of users, subject to the caching capacity constraint. The optimization problem is formally expressed as

$$(\mathbf{P1})\ \max_{\mathbf{X}} \mathrm{MOS}_{tot} = \sum_{t=1}^{T}\sum_{i=1}^{N}\sum_{f=1}^{F} p_{if}(t)\,\mathrm{MOS}_{if}(t), \tag{10a}$$

$$\text{s.t.}\quad C_1:\ x_{j,m}(t) \in [1,F], \forall j, \forall m, \forall t, \tag{10b}$$

$$C_2:\ p_{if}(t) \in (0,1), \forall i, \forall f, \forall t, \tag{10c}$$

$$C_3:\ \sum_{i=1}^{H_m} B\log_2\left(1+\mathrm{SINR}_{if}(t)\right) \le R_{m\max}, \forall f, \forall t, \tag{10d}$$

$$C_4:\ B\log_2\left(1+\mathrm{SINR}_{if}(t)\right) \ge R_{i\min}, \forall i, \forall f, \forall t, \tag{10e}$$

$$\mathrm{MOS}_{if}(t) = -C_1 \ln\left[3\mathrm{RTT} + \frac{\mathrm{FS}}{R_{if}(t)} + L\left(\frac{\mathrm{MSS}}{R_{if}(t)} + \mathrm{RTT}\right) - \frac{2\mathrm{MSS}\left(2^L - 1\right)}{R_{if}(t)}\right] + C_2. \tag{9}$$

where $x_{j,m}(t)$ denotes the caching decision of BS m at time t, $p_{if}(t)$ denotes the popularity of user i requesting for content f at time t. (10d) and (10e) are BS fronthaul link capacity constraint and user minimum transmit rate constraint, respectively.

Lemma 1. *The optimization problem in (10) is a NP-hard problem.*

Proof: See Appendix A . ■

As indicated by **Lemma 1**, the problem is NP-hard and therefore the problem has no polynomial-time solution discovered so far. We solve the formulated problem in two steps. The first step is to predict user mobility and content popularity. This is a time series prediction problem, therefore we adopt RNN algorithm in this step. The first step is illustrated in Section III. After obtaining user mobility and content popularity, the second step is to solve the caching matrix in (10). As can be seen from (10), the objective function is a long-term maximization problem. Meanwhile, the goal of RL is to obtain the long-term sum reward. Therefore, we adopt RL for the formulated problem. Furthermore, we consider our problem in a discrete time slot, where the caching decision for each time slot is only dependent on the information of last time slot. This meets the character of Markov decision process (MDP). Moreover, the formulated long-term maximization problem is non-trivial for conventional optimization approaches to obtain an optimal or a satisfying suboptimal solutions. Instead of utilizing conventional optimization approaches, we utilize Q-learning algorithm to solve the problem. In order to improve the performance of Q-learning, we adopt learning automata (LA) for the action selection of every state in Q-learning.

III. RNN Based User Mobility and Content Popularity Prediction

In reality, users are mobile, and may have periodic mobility patterns. We assume that the time interval of the change of users' mobility is T. Universal approximation theorems in [37] show that neural networks have a striking fact that they compute any function at all. Recurrent neural networks are well suited to time series prediction problems [38–40]. Hencefore in this section, we propose an RNN based user position and content popularity prediction algorithm. The inputs the RNN for user mobility prediction are users' positions in one hour containing 12 positions.

Definition 1. *As in Fig. 2, the inputs of the RNN cell are current input data x_l and hidden state data from previous layer a_l, the output is hidden state a_{l+1} which is given to the next RNN cell and also utilized to predict y_l^{output}. The definition of the parameters adopted in RNNs are given as below.*

- *Input layer (IL): Passive layer, receiving single value of input, and duplicating the value to multiple outputs. $z_i^u(t)$ represents the position of the i-th user at time t, and the network input is the history combination of H positions, i.e., $[z_i^u(t-H), z_i^u(t-H+1), \cdots, z_i^u(t-1)]$. Here we set $H = 6$ which means that we use history one hour positions to predict positions in the next hour.*
- *Output Layer (OL): Active layer, modifying the data of input, and exporting given outputs for the network. $y^{output} = [y_{t+1,i}, y_{t+2,i}, \cdots, y_{t+N,i}]$ represents the positions of the i-th user for the next step, where N represents length of the track. The output of the RNN is given as below*

$$y_l^{output} = \mathrm{softmax}\left(W_{ya}^l a_l + b_l^o\right), \tag{11}$$

where W_{ya}^l and b_l^o denote the output weight matrix and the output bias of layer l respectively. The softmax function is a normalized exponential function. a_l is the state of the RNN cell, which is given as

$$a_l = \tanh\left(W_{ax}^l x_{l-1}^{input} + W_{aa}^l a_{l-1} + b_l\right), \tag{12}$$

where W_{ax}^l denotes the input weight matrix of layer l, W_{aa}^l denotes the transmit weight matrix of layer l and b_l denotes the bias of layer l. $\tanh(\bullet)$ is the activation function (i.e., Hyperbolic tangent function.), which is applied to node inputs to produce node output.
- *Mean square error (MSE): The performance of the RNN based user mobility prediction developed in our work is evaluated by MSE, which adjusts the weight of the network after each iteration, thus it affects the rate of convergence. The forward propagation algorithm will pass the information through the network to make a prediction in the output layer. After forward propagation, the network uses a MSE function to measure the loss between trained output and output:*

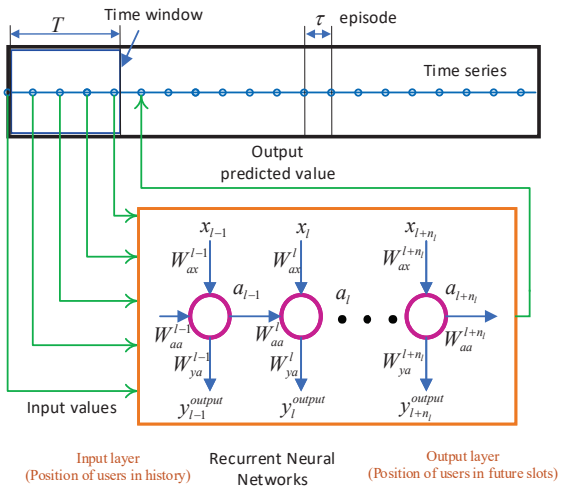


Fig. 2. Flow chart of the RNN for time series prediction.

$$\mathrm{MSE}\left(y^{output}, y^{real}\right) = \frac{1}{N_u}\sum_{i=1}^{N_u}\sqrt{\frac{1}{N_o}\sum_{j=1}^{N_o}\left[y_i^{output}(j) - y_i^{real}(j)\right]^2}, \tag{13}$$

*where y^{output} and y^{real} denotes the output of the network and real data, respectively. The details of RNN based user mobility prediction are shown in **Algorithm 1**.*

Algorithm 1 The RNN algorithm for user mobility prediction.

Stage One: Training
Input: Time interval H, training data: history positions of all users $[z_i^u(t-H), z_i^u(t-H+1), \cdots, z_i^u(t-1)]$. Parameters of the RNN: number of hidden layers 2, learning rate 0.1, goal of MSE, the number of epochs.
Initialize number of RNN cells, the weight matrix and the bias.
for each episode **do**
  Choose input from training set;
  Forward propagation algorithm: calculating the output of the layers;
  Calculate MSE function;
  Backward propagation algorithm: adjust the weights and bias in each RNN cell according to the MSE function;
end for
Output: Weights and bias of each RNN cell of the network.
Stage Two: Testing
Input: Interval H, Testing data: Positions of users, Parameters of RNN: weights, bias of the network, the number of epochs.
for each episode **do**
  Choose input from testing set;
  Forward propagation: calculating the output of the layers;
  Calculate loss function;
end for
Output: The accuracy between predicted user's positions and real data.

The stability of RNNs with respect to small perturbations of the inputs is meaningful, because small perturbations like users' unfrequent visit positions and abnormal contents surfing are not main concern in our scenario. We focus on scenarios that people periodical visit some places, and the content popularity are stable.

Remark 1. *Results suggest that the RNN for user mobility prediction that are learned by backpropagation have nonintuitive characteristics and intrinsic blind spots, and its structure is connected to the data distribution in a non-obvious way [41]. Thus in our scenario, data preprocessing (i.e., perturbations elimination) is of great importance to increase the stability of RNNs based user mobility prediction.*

Remark 2. *The advantage of utilizing RNNs for content popularity prediction is that it stores the non-linear characters in the multi-parameters of RNNs, which reflects the nature of the relationship between history content popularity and future content popularity.*

Content popularity plays an important role in overall achievable sum rate, therefore content popularity is another element that we predict for content cooperative caching. Note that the dimension of content popularity is larger than user mobility. For this reason, we set more hidden layers than user mobility prediction network. The structure of the neural network based content popularity prediction contains three hidden layers and a fully connected output layer. The input data is partitioned into blocks of size $k_{in} = 5 \times 1$, where each block represents for content popularity in 5 time slots. The output is a block of size $k_{out} = 1 \times 1$, where each block represents the content popularity in the next time slot.

IV. Learning Automata based Q-Learning for Content Cooperative Caching

A. Q-learning Based Solution

Q-learning is an off-policy and model-free RL algorithm compared to policy iteration and value iteration. Moreover, the agent of Q-learning does not need to traverse all states and actions on contrary to value iteration. Q-learning stores state and action information into a Q table $Q^{\pi}(s,a)$, where π is the policy, s represents the state, and a represents the action. The objective of the Q-learning is to find optimal policy π^*, and estimate the optimal state-action value function $Q^*(s,a) := Q^{\pi}(s,a), \forall s, a$, which can be shown that [42]

$$\pi^*(s) = \arg\max_a Q^*(s,a), \forall s \in S, \tag{14}$$

where $\pi^*(s)$ denotes the optimal policy of state s, and S denotes the set of all states.

The expression for optimal strategy $V_{\pi^*}(s)$ is rewritten as

$$V_{\pi^*}(s) = \max_{a \in A}\left[r(s,a) + \gamma \sum_{s^* \in S} P_{s,s^*} V_{\pi^*}(s^*)\right]. \tag{15}$$

where Bellmans optimality criterion states that there is one optimal strategy in a single environment setting [43].

With the above circumstances, we define the states and the actions of Q-learning in our scenario as below

1) **States space:** states are matrices with size of $S \times M$, and the total number of states is $F^{S \times M}$.
2) **Actions space:** actions we take here is to change the state, and thus the actions are increase one or decrease one of the element state matrix. As a result, the total number of actions are $2 \times S \times M + 1$, with the last action $[0]_{S \times M}$, which means that the state does not need to change.
3) **Reward function:** the reward that we give here is based on objective function MOS:

$$r(a_t, s_t, s_{t+1}) = \begin{cases} 0 & \mathrm{MOS}_{s_{t+1}} \geq \mathrm{MOS}_{s_t} \\ 1 & \mathrm{MOS}_{s_{t+1}} < \mathrm{MOS}_{s_t}. \end{cases} \tag{16}$$

4) For Q value (or state-action value) update, Bellman Equation is applied,

$$Q(s,a) \leftarrow (1-\alpha) Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') \right]. \tag{17}$$

where $\alpha \in (0,1)$ denotes the learning rate, which determines the speed of learning, (i.e., a larger α value leads to a faster learning process, yet may result in non-convergence of learning process, and a smaller α value leads to a slower learning process). $\gamma \in (0,1)$ denotes the discount factor, which determines the balance between historical Q value and future Q value, (i.e., a larger γ value means that the future Q value is more important, and a smaller γ value will balance more on the current Q value).

In Q-learning algorithm, the state matrix $(s_t, a_t, s_{t+1}, Q(s,a))$ is stored in a matrix to train the Q matrix. Using greedy Q-learning algorithm to search best state-action policy may cause an issue that some states may never be visited. For this reason, the agent might become stuck at certain states without even knowing about better possibilities. Therefore we propose a LAQL based cooperative caching algorithm. LAQL based cooperative caching algorithm is a tradeoff between exploration and exploitation, where exploration means to explore the effect of unknown actions, while exploitation means to remain at the current optimal action since learning.

Q-learning achieves optimal solution if $\varepsilon = 1$. However, in this circumstance, the probability that Q-learning achieves local-optimal solution is much higher than that Q-learning achieves optimal solution. For this reason, we normally set $\varepsilon < 1$ to explore all states, and to obtain quasi-optimal solution.

B. Learning Automata Based Q-learning For Cooperative Caching

An LA is an adaptive decision-making unit which learns the optimal action from a set of actions offered by the environment it operates in. Learning automata and Q-learning are used in different scenarios. Learning automata is usually applied to an agent in a random and stationary environment that has one state, such as multi-armed bandit problem, that has only one state, while Q-learning is used by an agent in an environment that can be described by multiple states. In our scenario, the optimization problem has complex structures and long horizons, which poses a challenge for the RL agent due to the difficulty in specifying the problem in terms of reward functions as well as large variances in the learning agent. We address these problems by combining LA with Q-learning for the cooperative caching. An illustration of the learning automata based Q-learning (LAQL) algorithm for content cooperative caching can be found in Fig. 3.

The goal of Q-learning is to find an optimal policy that maximize the expected sum of discounted return:

$$\pi^* = \arg\max_{\pi} E^{\pi} \left[\sum_{t=0}^{T-1} \gamma^{t+1} \widetilde{r}\left(\widetilde{s}_t, \widetilde{s}_{t+1}\right) \right]. \tag{19}$$

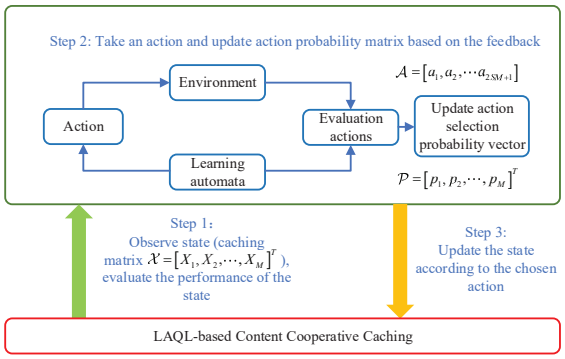


Fig. 3. An illustration of LAQL-based content cooperative caching.

Algorithm 2 The LAQL Algorithm for Content Cooperative Caching (Training Stage)

Input: Caching matrix of state s, caching action a, $Q(s,a)$, users' positions $z_i^u(t)$, and content popularity $p_{if}(t)$.

1: Initialise Q-table $Q(s,a)$, state s, and action selection probability vector in LA for each state s.
2: **repeat**
3: **for** each episode **do**
4: **for** each step **do**
5: Choose cooperative caching action a according to **Learning Automata Based Action Selection Scheme:**
5.1: The agent chooses one of the actions according to the distribution of $P_a(t)$ in the LA for the current state;
5.2: Update the reward probability estimation based on the feedback from the environment for the chosen action.
5.3: **If** action is active, i.e., a reward is achieved. **Then**

$$\begin{cases} p_i(t+1) = max\left\{p_i(t) - \Delta, 0\right\}, i \neq h \\ p_h(t+1) = 1 - \sum\limits_{i, i \neq h} p_i(t+1) \end{cases} \tag{18}$$

Else $P_a(t+1) = P_a(t)$.
EndIf
6: Take the chosen action a, thereafter calculate reward r of the action a and state s;
7: Update Q-table: calculate Q value using the defined reward function in (17);
8: State update: $s \leftarrow s'$;
9: **end for**
10: **end for**
11: **until** state s does not change.

Output: Q-table $Q(s,a)$, and content cooperative caching matrix of BSs X.

In our scenario, the state space and action space are discrete. Different from traditional Q-learning, the agent of LAQL algorithm chooses the actions according to the effectiveness of actions (i.e., whether the action is a success or failure). The LAQL algorithm consists of three steps. In each iteration,

firstly, an action probability vector is maintained: $P_a(t) = [p_1(t), p_2(t), \cdots, p_{n_a}(t)]$, where n_a represents the number of actions of the target state and $\sum_{i=1\cdots n_a} p_i(t) = 1$. We select an action according to the probability distribution $P_a(t)$. Secondly, based on the current feedback, the maximum likelihood reward probability is estimated for the chosen action in the current iteration. Suppose the ith action of the target state is chosen in the current iteration, then the reward probability $\widetilde{d}_i(t)$ is updated as [44, 45]:

$$\begin{aligned} u_i(t) &= u_i(t-1) + (1 - r(t)) \\ v_i(t) &= v_i(t-1) + 1 \\ \widetilde{d}_i(t) &= {}^{u_i(t)}/_{v_i(t)}, \end{aligned} \tag{20}$$

where $u_i(t)$ denotes the number of times that action i has been rewarded when selected. $v_i(t)$ represents the number of times that action i has been selected.

Algorithm 3 The LAQL Algorithm for Content Cooperative Caching (Testing Stage)

Input: state s, action a.

Generate a content cooperative caching matrix of network randomly, and set it as the initiating state s;
for each iteration **do**
Choose an action of the current state according to **Learning Automata based Action Selection Scheme**;
Calculate reward, and update state;
end for

Output: Content cooperative caching matrix of BSs X.

Thirdly, if $r(t) = 0$, meaning a reward is achieved, the LAQL agent increases the probability of selecting the current best action based on the current reward estimates of all actions. In more details, if $\widetilde{d}_h(t)$ is the largest element among all reward estimates of all actions, we update $p_i(t), i \in \{1, 2, \cdots, n_a\}$ as:

$$\begin{cases} p_i(t+1) = max\{p_i(t) - \Delta, 0\}, i \neq h \\ p_h(t+1) = 1 - \sum_{i, i \neq h} p_i(t+1) \end{cases} \tag{21}$$

where $\Delta = {}^{1}/_{n_a \kappa}$, and κ being a positive integer.

If $r(t) = 1$, then $P_a(t+1) = P_a(t)$.

The details of LAQL for content cooperative caching are illustrated in **Algorithm 2** and **Algorithm 3**. Clearly, although the converged $Q(s, a)$ value indicates the long term average reward of the actions for a given state s and action a, due to the random nature of the studied scenario, i.e., the uncertainty of the destination state upon an action, the optimal action may not surely return a higher reward than the suboptimal actions at a certain time instant when we select it. Therefore, although it is important to apply LA to balance exploration and exploitation, it is necessary to show that LA can converge to the optimal action with high probability in this random environment.

Lemma 2. *Given that $Q(s, a)$ is capable of converging to the optimal Q value, the LA corresponding to every state s can select the optimal action with arbitrary high probability.*

Proof: See Appendix B . ■

Remark 3. *The proposed LA based action selection scheme enables the states in Q-table to select the optimal action.*

C. Computational Issues

The computational complexities of traditional Q-learning and LAQL are quite different. In the process of content caching in different BSs, the complexity of Q-learning based cooperative caching depends primarily on the number of BSs M, caching capacity of BS S the number of contents F, as can be seen in TABLE II. We consider an example where $F = 10$, $S = 4$, $M = 10$, the computational complexity is shown in the second column in TABLE II. As can be seen from TABLE II, LAQL based cooperative caching takes 32000 operations, whereas optimal caching takes $10^{4\times10}$ and ε-greedy Q-learning takes 4000. It can also be seen from TABLE II that with the increased number of contents in the central network, the number of operations of Q-learning algorithm and LAQL algorithm increase exponentially.

In terms of storage requirements, however, the optimal caching method possesses lowest number of memory unites since it does not need to memorize much knowledge. The table of LAQL and ε-greedy Q-learning requires a higher number of memory units, the maximum of which is $F^{SM} \times (2 \times S \times M + 1)$.

V. Numerical Results

In this section, simulations are conducted to evaluate the performance of the proposed algorithms. We first consider the situation that the users move in a random walk model, after that we utilize user mobility data collected from smartphones to test the performance of neural network based user mobility prediction. The position data of on single user is collected from daily life using GPS-tracker app on an Android system smartphone. We open the APP when we start the journey, and record the positions during the journey. Meanwhile, we also consider random walk model for content popularity. We use random walk model to test the prediction accuracy of our proposed neural network for content popularity prediction. For the simulation setup, we consider a cooperative circle region with length of a side 4 km. There are $M = 2$ BSs in the region, and $N = 100$ users are randomly and independently distributed in the region. We assume that the total number of contents is $F = 10$. All the simulations are performed on a desktop with an Intel Core i7-7700 3.6 GHz CPU and 16 GB memory.

To investigate the performance of the proposed algorithms, three different algorithms are simulated for cooperative caching system, respectively. The benchmark scheme we compared is non-cooperative caching, i,e,. BSs do not cooperative with each other. In other words, BSs cache the most popularity contents independently. Another benchmark is random caching, i.e., the BSs do not know the prior information such as content popularity and user mobility, hence BSs cache contents randomly.

TABLE II
NUMBER OF OPERATIONS REQUIRED FOR DIFFERENT SCHEMES

Method	Number of Operations	Numerical Example
Optimal caching	F^{SM}	$10^{4\times10}$
ε-greedy Q-learning Algorithm	FSM	$10\times4\times10$
Learning automata based Q-learning Algorithm	$FSM\times2SM$	$10\times4\times10\times2\times4\times10$

TABLE III
PARAMETER CONFIGURATIONS

Parameter	Description	Value
F	Content library size	10
M	Number of BSs	2
N_u	Number of Users	100
L	Size of content	1
B	Bandwidth	20MHz [30]
P_t	Transmit power of each BS	20 dBm [30]
α	Path loss exponent	3
s	Caching capacity	4
C_1	constant	1.120 [46]
C_2	constant	4.6746 [46]

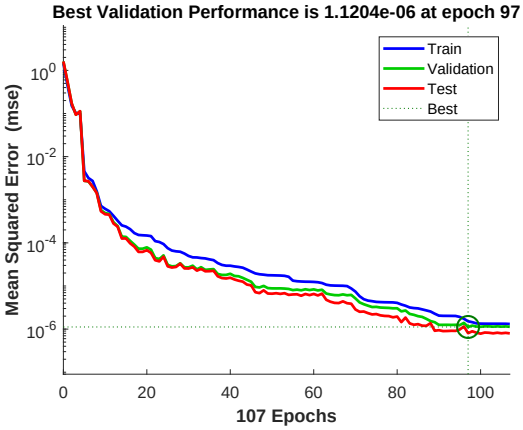


Fig. 4. MSE of RNNs for content popularity prediction.

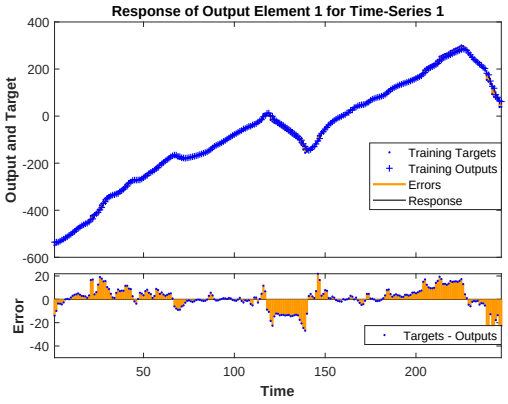


Fig. 6. Comparison between outputs of the RNN and the real data.

A. Simulation Results of Content Popularity Prediction

In this subsection, we utilize history data of content popularity to predict content popularity in the future. More precisely, content popularity in 5 time slots are provided as RNN input, while the output is content popularity in the next slot. As is shown in Fig. 4, MSE of the RNN decreases with epochs. It reached bottom after approximately 90 epochs.

B. Simulation Results of User Mobility Prediction

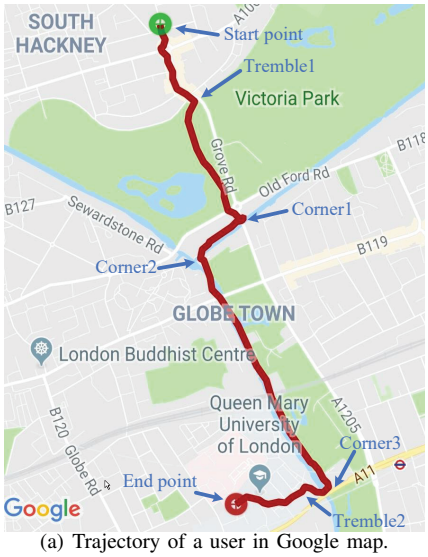
Sociological researches have provided us various methods to collect people's moving trajectory. However, it consumes a lot of resources (i.e., time, manpower, and money) to collect a large number of data. In this paper, we collect movements of one user, to validate the accuracy of our proposed RNNs based prediction algorithm. In Fig. 5(a), we use measured data from smartphone to test the prediction accuracy of the RNN. The data is collected from daily life using GPS-tracker app on an Android smartphone. Fig. 5(a) shows the trajectory of user in Google Maps, and there are trembles and corners in the trajectory, which will effect prediction accuracy. The results indicated in Fig. 5(b) confirm the accuracy of RNNs. This phenomenon is also confirmed by the insights in **Remark 2**.

Fig. 6 shows performance of RNNs based user mobility prediction, where the outputs and the targets denote the output of RNN and real trajectory, respectively. According to Fig. 6, the performance of RNNs is reduced when the movement of the user is disturbed, e.g., at about 70 X axis where there is a tremble, in about 130 and 250 where the user changes the direction. Therefore, to improve the performance of RNNs based user mobility prediction, we ought to rule out the uncommon positions.

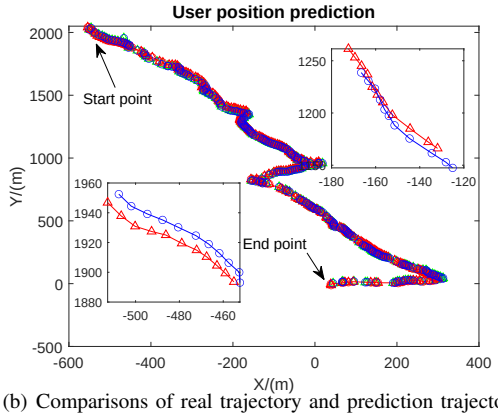
C. Simulation Results of LAQL Based Cooperative Caching

In this subsection, we compare the proposed LAQL based cooperative caching with the above mentioned traditional caching schemes. Fig. 7 shows QoE distribution of users in this rectangle area, with Q-learning based caching scheme. It can be observed that users near BSs have better QoE than those who are far from BSs. This is easy to understand because the distance between a user and a BS has important influence on MOS of the user, and also this can be further explored to analysis the deployment of BSs (i.e., deployment schemes and density of BSs).

Figure 8 shows the sum MOS of users as a function of transmit power. We first adjust the learning rate α and discount factor γ of conventional Q-learning algorithm, because α and γ have significant influence of Q-learning. After several adjustment, we set $\alpha = 0.75$ and $\gamma = 0.6$. After that, we set the learning rate and discount factor of LAQL the same as conventional Q-learning, aiming at reducing the interference of learning rate and discount factor between conventional Q-learning and the proposed LAQL algorithm. As can be seen from this figure, MOS increases with greater transmit power. We see that LAQL based cooperative caching outperforms conventional Q-learning. This is because the action selection scheme of LAQL enables the agent to choose optimal action



(a) Trajectory of a user in Google map.



(b) Comparisons of real trajectory and prediction trajectory.

Fig. 5. User mobility prediction using RNNs (data from smartphone).

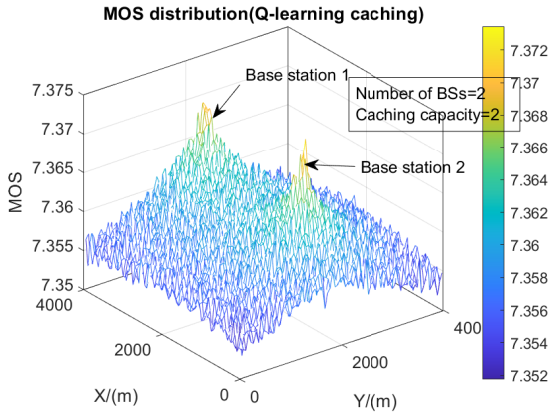


Fig. 7. The MOS of distribution of users in different positions of the area.

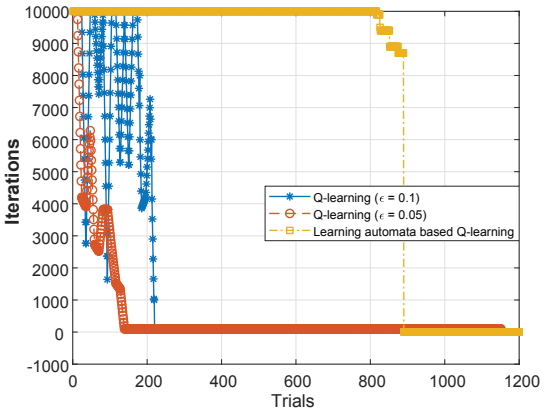


Fig. 9. The convergence of learning automata based Q-learning and conventional Q-learning over trials.

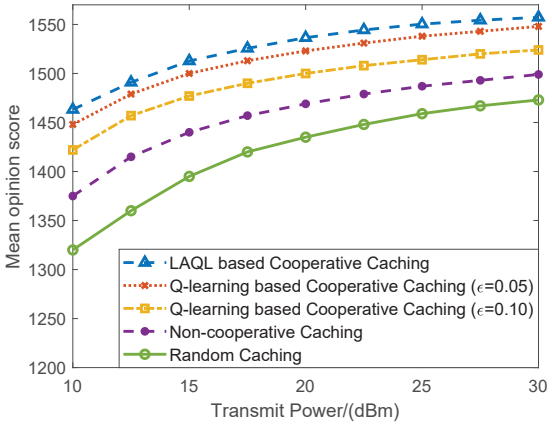


Fig. 8. The MOS of different caching schemes vs. transmit power.

in each state, while action selection scheme in conventional Q-learning is based on a stochastic mechanism (i.e., ϵ-greedy). Fig. 8 also shows that cooperative caching outperforms random caching and non-cooperative caching. The reason is that the cooperative caching is capable of better balancing the caching resource in different BSs than non-cooperative caching.

Fig. 9 shows the convergence of LAQL and ε-greedy Q-learning. For each trial, we try 10000 iterations. As can be seen from the figure, the curves of Q-learning fluctuate a lot, and with the increase of ϵ, Q-learning takes more trials to converge. One can also see from Fig. 8 that, although the proposed LAQL algorithm takes more trails to converge, the performance of LAQL outperforms ε-greedy Q-learning when LAQL converges. The reason is that for the proposed LAQL algorithm, the optimal action is chosen for every state.

VI. Conclusion

In this article, content placement scheme in cooperative caching was studied along with user mobility and content popularity prediction for wireless communication system. In order to fulfill backhaul constraints, an optimization problem of users' MOS maximization was formulated. 1) For user position and content popularity prediction, an RNN algorithm was

proposed that utilized the powerful nonlinear fitting ability of the RNN. The efficiency of the proposed scheme was validated by practical testing data set. 2) For content cooperative caching problem, an LAQL based cooperative caching algorithm was proposed, utilizing LA for optimal action selection in every state. The effectiveness of the proposed solution was illustrated by numerical experiments. One promising extension of this work is to consider a scenario in which the sizes of contents are dynamic and with high variety. Another extension is to apply the learning automata based action selection scheme in deep reinforcement learning (DRL) algorithms to improve the learning speed.

Appendix A: Proof of Lemma 1

The express of the objective function in (10) is (9). To prove that the optimization problem in (10) is NP-hard. We first consider the NP-hardness of the following problem.

Problem 1 (N-Disjoint Set Cover Problem)**.** *Given a set $\mathcal{A}$ and a collection $\mathcal{B}$, the N-disjoint set cover problem is to determine whether $\mathcal{B}$ can be divided into N disjoint set covers or not. More generally, consider a bipartite graph $\mathcal{G} = (\mathcal{A}, \mathcal{B}, \mathcal{E})$ with edges $\mathcal{E}$ between two disjoint vertex sets $\mathcal{A}$ and $\mathcal{B}$. Thus the N-disjoint set cover problem is whether there exist N disjoint sets $\mathcal{B}_1, \mathcal{B}_2, \cdots \mathcal{B}_N \subset \mathcal{B}$ in which $|\mathcal{B}_1| + |\mathcal{B}_2| + \cdots + |\mathcal{B}_N| = |\mathcal{B}|$ and $\mathcal{A} = \underset{\boldsymbol{B}\in\mathcal{B}_1}{U} N(\boldsymbol{B}) = \underset{\boldsymbol{B}\in\mathcal{B}_2}{U} N(\boldsymbol{B}) = \cdots = \underset{\boldsymbol{B}\in\mathcal{B}_N}{U} N(\boldsymbol{B})$. The N-disjoint set cover problem can be denoted as* $\mathbf{NDSC}(\mathcal{A}, \mathcal{B}, \mathcal{E})$.

According to [47], the $\mathbf{NDSC}(\mathcal{A}, \mathcal{B}, \mathcal{E})$ is proved to be NP-complete when $(N \geq 2)$. Consider $\mathcal{A}$ to be the set of mobile users, $\mathcal{B}$ to be BSs with caching capacity, and $\mathcal{E}$ to be the edges representing connections between mobile users and BSs. The content popularity is $P = [p_1, p_2, \cdots, p_F]$ where $p_f < 1$ for $\forall f \in \mathcal{F}$. Without loss of generality, we set the caching capacity of each BS is equals to 1, which means that each BS can cache only one content. According to (9), the maximization of the total MOS of users can only be gained by the BS caches all contents that the user requested. In other words, each user has at least one neighboring BS caching content 1 and other neighboring BS caching contents 2 to F. Letting $\mathcal{B}_1$, $\mathcal{B}_2$, and $\mathcal{B}_F$ be the disjoint sets of BSs containing file 1 to file F, separately. We conclude that determining whether the objective function (10) is maximized is equivalent to determining the existence of and forming a N-disjoint set cover problem. From the analysis above, one can conclude that problem (10) is NP-hard.

Appendix B: Proof of Lemma 2

We shall prove **Lemma 2** in two steps. Firstly, we show that we can convert the average reward value for an action to a reward probability, given that a higher average reward corresponds a higher reward probability. In other words, we need to map the value of Q value[2] from $(-\infty, +\infty)$ to $(0, 1)$.

[2]For Q-learning, it is necessary to have bounded rewards for the actions in order to make the system converge. Here we use the range $(-\infty, +\infty)$ to indicate the bounded rewards can be arbitrarily large.

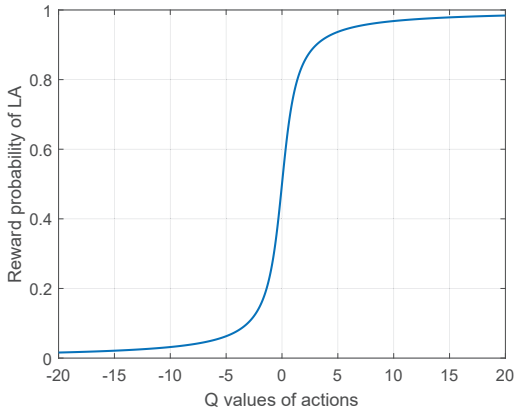


Fig. 10. Mapping reward value to probability.

This conversion can be done by a modified arctangent-shape function, as shown in Fig. 10. Given that $Q(s, a)$ can converge to the optimal Q value, we can be certain that there exists a long term stationary reward behind each action of each state, and thus a stationary reward probability after conversion. Secondly, we need to show that the LA is ϵ-optimal in the random and stationary environment, represented by the reward probability behind each action, but hidden before learning. The proof of ϵ-optimal is not trivial and the detailed proof can be found in [44]. Here we just outline the main steps of the proof presently.

Now let's suppose there are r actions for state s. The real reward in probability for action i, after conversion from the long term average reward value of action i, is d_i. Correspondingly, the estimated reward probability by the LA for action i at time t is $\hat{d}(t)_i$, where $i \in \{1, \ldots, r\}$. Define $\mathbf{D} = [d_1, \ldots, d_r]$ and $\hat{\mathbf{D}}(t) = [\hat{d}_1(t), \ldots, \hat{d}_r(t)]$. Let action probability vector be $\mathbf{P}(t) = [p_1(t), p_2(t), ..., p_r(t)]$, which is used to determine the issue of which action is to be selected at t, where $\sum_{j=1\ldots r} p_j(t) = 1$. Let m be the index of the optimal action. To show the ϵ-optimality of LA, we need to prove that given any $\epsilon > 0$ and $\delta > 0$, there exist an $N_0 > 0$ and a $t_0 < \infty$ such that for all time $t \geq t_0$ and for any positive learning parameter $N > N_0$,

$$Pr\{p_m(t) > 1 - \epsilon\} > 1 - \delta. \tag{24}$$

This result has been proven in [44], and we illustrate here only the outline of the proof, just for a quick reference to the theorem that has been used to support the result. One is recommended to refer to the original paper for a thorough elaboration of the reasoning.

The ϵ-optimality of the DPA was based on the submartingale property of the Action Probability sequence $\{p_m(t)_{t>t_0}\}$. Thus the submartingale convergence theory and the theory of Regular functions were invoked to prove that the D-PA will converge to the optimal action in probability, i.e., $Pr\{p_m(\infty) = 1\} \to 1$.

$\{p_m(t)_{t>t_0}\}$ being a submartingale was proven by the definition of submartingale. Firstly, $p_m(t)$ is a probability, hence $E[p_m(t)] \leq 1 < \infty$ holds. Secondly, according to the

$$
\begin{aligned}
&E[p_m(t+1)|\mathbf{P}(t)] \\
&= \sum_{j=1\ldots r} p_j(t)\left(d_j\left(q(t)(p_m(t)+c_t\Delta)+(1-q(t))(p_m(t)-\Delta)\right)+(1-d_j)p_m(t)\right) \\
&= p_m(t) + \sum_{j=1\ldots r} p_j(t)d_j\left(q(t)(c_t\Delta+\Delta)-\Delta\right).
\end{aligned} \tag{22}
$$

$$
\begin{aligned}
U^{\infty}\Phi(\mathbf{P}) &= E[\Phi(\mathbf{P}(\infty))|\mathbf{P}(0)=\mathbf{P}] \\
&= \sum_{j=1}^{r}\Phi(\mathbf{e}_j)Pr\{\mathbf{P}(\infty)=\mathbf{e}_j|\mathbf{P}(0)=\mathbf{P}\}, \quad \textit{the definition of Expectation} \\
&= \Phi(\mathbf{e}_m)Pr\{\mathbf{P}(\infty)=\mathbf{e}_m|\mathbf{P}(0)=\mathbf{P}\}, \quad \textit{the boundry condition } \Phi(\mathbf{e}_j)=0,\ \forall\ j\neq m \\
&= Pr\{\mathbf{P}(\infty)=\mathbf{e}_m|\mathbf{P}(0)=\mathbf{P}\}, \quad \textit{the boundry condition } \Phi(\mathbf{e}_m)=1 \\
&= \Gamma_m(\mathbf{P}).
\end{aligned} \tag{23}
$$

learning mechanism of DPA, the expectation of $p_m(t)$ can be calculated explicitly in (22), where $q(t) = Pr(\hat{d}_m(t) > \hat{d}_i(t))$, $\forall i \neq m$, is the accuracy probability of the reward estimates. By the definition of submartingale, one can see that if $\forall t > t_0, \quad q(t) > \frac{\Delta}{c_t\Delta+\Delta} = \frac{1}{c_t+1} \geq \frac{1}{2}$ holds, then $E[p_m(t+1)|\mathbf{P}(t)] > p_m(t)$ becomes true, and $\{p_m(t)_{t>t_0}\}$ is thus a submartingale.

It has been proven in [44] that the accuracy probability of the reward estimates can be arbitrarily high. The proof itself involves quite a bit algebraic manipulations, and is thus complicated, but the rationale behind it is straight forward. It proved that after t_0, each action, with a very high probability, will be selected for a large number of times. Then, given that each action has been tried for a large number of times, the estimate of its reward will certainly become accurate enough to surpass the number $\frac{1}{2}$, which grantees $\{p_m(t)_{t>t_0}\}$ for being a submartingale.

Once the submartingale property is proven, by the submartingale convergence theory, $p_m(\infty) = 0$ or 1 holds. To prove the DPA converges to the optional action, we need to shown that $p_m(\infty) = 1$ will happen with arbitrarily large probability, rather than $p_m(\infty) = 0$. This is then equivalent to proving the convergence probability below

$$\Gamma_m(\mathbf{P}) = Pr\{\mathbf{P}(\infty)=\mathbf{e}_m|\mathbf{P}(0)=\mathbf{P}\} \rightarrow 1, \tag{25}$$

where $\mathbf{e}_j$ is the unit vector with the j^{th} element being 1. To prove Eq. (25), we now need to clarify the following definitions [48]:

$\Phi(\mathbf{P})$: a function of $\mathbf{P}$.

U: an operator such that $U\Phi(\mathbf{P}) = E[\Phi(\mathbf{P}(n+1))|\mathbf{P}(n) = \mathbf{P}]$,

applying U for n times: $U^n\Phi(\mathbf{P}) = E[\Phi(\mathbf{P}(n))|\mathbf{P}(0) = \mathbf{P}]$.

The function $\Phi(\mathbf{P})$ is:

- Superregular: If $U\Phi(\mathbf{P}) \leq \Phi(\mathbf{P})$. Then applying U repeatedly yields:

$$\Phi(\mathbf{P}) \geq U\Phi(\mathbf{P}) \geq U^2\Phi(\mathbf{P}) \geq ... \geq U^{\infty}\Phi(\mathbf{P}). \tag{26}$$

- Subregular: If $U\Phi(\mathbf{P}) \geq \Phi(\mathbf{P})$. In this case, if we apply U repeatedly, we have

$$\Phi(\mathbf{P}) \leq U\Phi(\mathbf{P}) \leq U^2\Phi(\mathbf{P}) \leq ... \leq U^{\infty}\Phi(\mathbf{P}). \tag{27}$$

- Regular: If $U\Phi(\mathbf{P}) = \Phi(\mathbf{P})$. In such a case, it follows that:

$$\Phi(\mathbf{P}) = U\Phi(\mathbf{P}) = U^2\Phi(\mathbf{P}) = ... = U^{\infty}\Phi(\mathbf{P}). \tag{28}$$

If we suppose $\Phi(\mathbf{P})$ satisfies the boundary conditions $\Phi(\mathbf{e}_m) = 1$ and $\Phi(\mathbf{e}_j) = 0, (\forall j \neq m)$, then, one can calculate $U^{\infty}\Phi(\mathbf{P})$ and obtain very interesting results as equation (23).

Equation (23) tells that the convergence probability can be studied by applying U an infinite number of times to the function $\Phi(\mathbf{P})$. What's more, if $\Phi(\mathbf{P})$ is a Regular function, the sequence of operations will lead to a function that equals the function $\Phi(\mathbf{P})$ itself, whereas if $\Phi(\mathbf{P})$ is a subregular/superregular function, the sequence of operations can be bounded from below/above by $\Phi(\mathbf{P})$.

What the authors have done in [44], is to find a subregular function of $\mathbf{P}$ that meets the boundary conditions, to bound the convergence probability of $\Gamma_m(\mathbf{P})$ from below. Then they proved the subregular function itself converges to unity, thus implies that the convergence probability, which is greater than or equal to the subregular function of $\mathbf{P}$, will converge to unity as well.

REFERENCES

[1] Z. Yang, Y. Liu, and Y. Chen, "Q-learning for content placement in wireless cooperative caching," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Abu Dhabi, UAE, Dec. 2018.

[2] J. Montalban, P. Scopelliti, and *et al.*, "Multimedia multicast services in 5G networks: Subgrouping and non-orthogonal multiple access techniques," *IEEE Commun. Mag.*, vol. 56, no. 3, pp. 91–95, Mar. 2018.

[3] Cisco, *Cisco visual networking index: Global mobile data traffic forecast update, 2016-2021*. White paper, Feb. 2017.

[4] Z. Qin, J. Fan, Y. Liu, Y. Gao, and G. Y. Li, "Sparse representation for wireless communications: A compressive sensing approach," *IEEE Signal Process. Mag.*, vol. 35, no. 3, pp. 40–58, May 2018.

[5] Y. Liu, Z. Ding, M. Elkashlan, and H. V. Poor, "Cooperative non-orthogonal multiple access with simultaneous wireless information and power transfer," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 4, pp. 938–953, Apr. 2016.

[6] Y. Liu, Z. Ding, M. Elkashlan, and J. Yuan, "Nonorthogonal multiple access in large-scale underlay cognitive radio networks," *IEEE Trans. Veh. Technol.*, vol. 65, no. 12, pp. 10 152–10 157, Dec. 2016.

[7] Y. Liu, M. Elkashlan, Z. Ding, and G. K. Karagiannidis, "Fairness of user clustering in mimo non-orthogonal multiple access systems," *IEEE Commun. Lett.*, vol. 20, no. 7, pp. 1465–1468, Jul. 2016.

[8] S. Tombaz, P. Monti, and *et al.*, "Is backhaul becoming a bottleneck for green wireless access networks?" in *IEEE Proc. of International Commun. Conf. (ICC)*, Jun. 2014.

[9] A. Ioannou and S. Weber, "A survey of caching policies and forwarding mechanisms in information-centric networking," *IEEE Commun. Surv. Tut.*, vol. 18, no. 4, pp. 2847–2886, Fourthquarter 2016.

[10] D. Liu, B. Chen, C. Yang, and A. F. Molisch, "Caching at the wireless edge: design aspects, challenges, and future directions," *IEEE Commun. Mag.*, vol. 54, no. 9, pp. 22–28, Sep. 2016.

[11] J. Liao, K. K. Wong, Y. Zhang, Z. Zheng, and K. Yang, "Coding, multicast, and cooperation for cache-enabled heterogeneous small cell networks," *IEEE Trans. Wireless Commun.*, vol. 16, no. 10, pp. 6838–6853, Oct. 2017.

[12] S. Zhang, P. He, K. Suto, P. Yang, L. Zhao, and X. S. Shen, "Cooperative edge caching in user-centric clustered mobile networks," *IEEE Trans. Mob. Comput.*, vol. 8, no. 99, pp. 1791–1805, Aug. 2018.

[13] Y. Wang, X. Tao, X. Zhang, and Y. Gu, "Cooperative caching placement in cache-enabled D2D underlaid cellular network," *IEEE Commun. Lett.*, vol. 21, no. 5, pp. 1151–1154, May 2017.

[14] Y. Liu, Z. Qin, M. Elkashlan, A. Nallanathan, and J. A. McCann, "Non-orthogonal multiple access in large-scale heterogeneous networks," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 12, pp. 2667–2680, Dec. 2017.

[15] W. Jiang, G. Feng, and S. Qin, "Optimal cooperative content caching and delivery policy for heterogeneous cellular networks," *IEEE Trans. Mob. Comput.*, vol. 16, no. 5, pp. 1382–1393, May 2017.

[16] F. Wang, J. Xu, X. Wang, and S. Cui, "Joint offloading and computing optimization in wireless powered mobile-edge computing systems," *IEEE Trans. Commun. Technol.*, vol. 17, no. 3, pp. 1784–1797, Mar. 2018.

[17] Y. Liu, Z. Qin, M. Elkashlan, Y. Gao, and L. Hanzo, "Enhancing the physical layer security of non-orthogonal multiple access in large-scale networks," *IEEE Trans. Wireless Commun.*, vol. 16, no. 3, pp. 1656–1672, Mar. 2017.

[18] Z. Qin, H. Ye, and G. L. and *et al.*, "Deep learning in physical layer communications," *ArXiv*, 2018. [Online]. Available: https://arxiv.org/abs/1807.11713

[19] D. Silver, A. Huang, and *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016.

[20] V. Mnih, K. Kavukcuoglu, D. Silver, and *et al.*, "Playing atari with deep reinforcement learning," *ArXiv*, 2013. [Online]. Available: https://arxiv.org/abs/1312.5602

[21] V. Mnih, K. Kavukcuoglu, and *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[22] O. A. S. Muller and *et al.*, "Context-aware proactive content caching with service differentiation in wireless networks," *IEEE Trans. Wireless Commun.*, vol. 16, no. 2, pp. 1024–1036, Feb. 2017.

[23] Q. Li, W. Shi, X. Ge, and Z. Niu, "Cooperative edge caching in software-defined hyper-cellular networks," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2596–2605, Nov. 2017.

[24] A. Sadeghi, G. Wang, and G. B. Giannakis, "Adaptive caching via deep reinforcement learning," *ArXiv*, 2019. [Online]. Available: https://arxiv.org/abs/1902.10301

[25] C. Zhong, M. C. Gursoy, and S. Velipasalar, "A deep reinforcement learning-based framework for content caching," *ArXiv*, 2017. [Online]. Available: https://arxiv.org/abs/1712.08132

[26] Y. He, Z. Zhang, F. R. Yu, N. Zhao, H. Yin, V. C. M. Leung, and Y. Zhang, "Deep-reinforcement-learning-based optimization for cache-enabled opportunistic interference alignment wireless networks," *IEEE Trans. Veh. Technol.*, vol. 66, no. 11, pp. 10 433–10 445, Nov. 2017.

[27] S. O. Somuyiwa, A. Gyrgy, and D. Gndz, "A reinforcement-learning approach to proactive caching in wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 6, pp. 1331–1344, Jun. 2018.

[28] A. Sadeghi, F. Sheikholeslami, and G. B. Giannakis, "Optimal and scalable caching for 5G using reinforcement learning of space-time popularities," *IEEE J. Sel. Areas Commun.*, vol. 12, no. 1, pp. 180–190, Feb. 2018.

[29] A. Sadeghi, G. Wang, and G. B. Giannakis, "Deep reinforcement learning for adaptive caching in hierarchical content delivery networks," *IEEE Cog. Commun. Netw.*, vol. 5, no. 4, pp. 1024–1033, Dec. 2019.

[30] M. Chen, W. Saad, C. Yin, and M. Debbah, "Echo state networks for proactive caching in cloud-based radio access networks with mobile users," *IEEE Trans. Wireless Commun.*, vol. 16, no. 6, pp. 3520–3535, Jun. 2017.

[31] M. Chen, W. Saad, and C. Yin, "Virtual Reality over Wireless Networks: Quality-of-Service Model and Learning-Based Resource Management," *ArXiv*, 2017. [Online]. Available: https://arxiv.org/abs/1703.04209v2

[32] K. Shanmugam, N. Golrezaei, A. G. Dimakis, A. F. Molisch, and G. Caire, "Femtocaching: Wireless content delivery through distributed caching helpers," *IEEE Trans. Inf. Theory*, vol. 59, no. 12, pp. 8402–8413, Dec 2013.

[33] P. Document, "Definition of quality of experience (QoE)," *Document COM 12-LS 62-E, ITU-T*, January 2007.

[34] Y. Wang, W. Zhou, and P. Zhang, *QoE Management in Wireless Networks*, 3rd ed. Springer International Publishing, 2017.

[35] M. Rugelj, U. Sedlar, M. Volk, and *et al.*, "Novel cross-layer QoE-aware radio resource allocation algorithms in multiuser OFDMA systems," *IEEE Trans. Commun.*, vol. 62, no. 9, pp. 3196–3208, Sep. 2014.

[36] P. Ameigeiras, J. J. Ramos-Munoz, J. Navarro-Ortiz, P. Mogensen, and J. M. Lopez-Soler, "QoE oriented cross-layer design of a resource allocation algorithm in beyond 3G systems," *Computer Communications*, vol. 33, no. 5, pp. 571–582, 2010.

[37] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, vol. 4, no. 2, pp. 251 – 257, 1991.

[38] R. Nallapati, F. Zhai, and B. Zhou, "SummaRuNNer: A recurrent neural network based sequence model for extractive summarization of documents," in *AAAI Conference on Artificial Intelligence (AAAI-17)*, California, USA, Feb. 2017.

[39] N. Laptev, J. Yosinski, L. E. Li, and S. Smyl, "Time-series extreme event forecasting with neural networks at uber," in *International Conference on Machine Learning (ICML-17)*, California, USA, Dec. 2017.

[40] Z. Che, S. Purushotham, K. Cho, D. Sontag, and Y. Liu., "Recurrent neural networks for multivariate time series with missing values," *Scientific reports*, no. 8(1), pp. 2045–2322, 2018.

[41] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *ArXiv*, 2013. [Online]. Available: http://arxiv.org/abs/1312.6199

[42] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2016.

[43] C. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3, pp. 279–292, May 1992.

[44] X. Zhang, B. J. Oommen, O.-C. Granmo, and L. Jiao, "A formal proof of the optimality of discretized pursuit algorithms," *Applied Intelligence*, vol. 44, no. 2, pp. 282–294, Mar 2016.

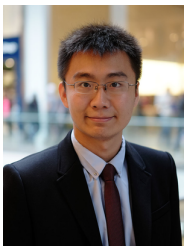
[45] X. Zhang, L. Jiao, J. B. Oommen, and O.-C. Granmo, "A conclusive analysis of the finite-time behavior of the discretized pursuit learning automaton," *IEEE Trans. neural net. learning sys.*, vol. 31, no. 1, pp. 284–294, Jan. 2020.

[46] J. Karedal, A. J. Johansson, F. Tufvesson, and A. F. Molisch, "A measurement-based fading model for wireless personal area networks," *IEEE Trans. Wireless Commun.*, vol. 7, no. 11, pp. 4575–4585, Nov. 2008.

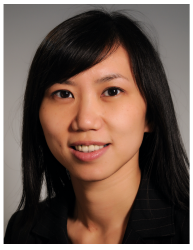
[47] C. Mihaela and D. Ding-Zhu, "Improving wireless sensor network lifetime through power aware organization," *Wireless Networks*, vol. 11, no. 3, pp. 333–340, May 2005.

[48] K. S. Narendra and M. A. Thathachar, *Learning automata: an introduction*. Courier Corporation, 2012.

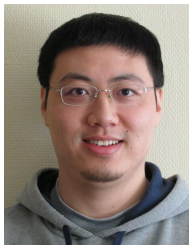
Zhong Yang (S'18) received the B.S. degree from the Nanjing University of Science and Technology (NUST), Nanjing, China, in 2013. He is currently pursuing the Ph.D. degree in School of Electronic Engineering and Computer Science at Queen Mary University of London. His research interests include machine learning, non-orthogonal multiple access, reconfigurable intelligent surface, mobile edge computing and caching.



Yuanwei Liu (S'13-M'16-SM'19) received the B.S. and M.S. degrees from the Beijing University of Posts and Telecommunications in 2011 and 2014, respectively, and the Ph.D. degree in electrical engineering from the Queen Mary University of London, U.K., in 2016. He was with the Department of Informatics, King's College London, from 2016 to 2017, where he was a Post-Doctoral Research Fellow. He has been a Lecturer (Assistant Professor) with the School of Electronic Engineering and Computer Science, Queen Mary University of London, since 2017. His research interests include 5G wireless networks, Internet of Things, machine learning, stochastic geometry, and matching theory. He received the Exemplary Reviewer Certificate of the IEEE WIRELESS COMMUNICATION LETTERS in 2015 and the IEEE TRANSACTIONS ON COMMUNICATIONS and the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS in 2016 and 2017. He has served as a TPC Member for many IEEE conferences, such as GLOBECOM and ICC. He currently serves as an Editor of the IEEE TRANSACTIONS ON COMMUNICATIONS, the IEEE COMMUNICATIONS LETTERS and the IEEE ACCESS. He is also a guest editor for IEEE JSTSP special issue on "Signal Processing Advances for Non-Orthogonal Multiple Access in Next Generation Wireless Networks".



Yue Chen (S'02-M'03-SM'15) received the bachelor's and master's degrees from the Beijing University of Posts and Telecommunications, Beijing, China, in 1997 and 2000, respectively, and the Ph.D. degree from the Queen Mary University of London (QMUL), London, U.K., in 2003. She is currently a Professor of telecommunications engineering with the School of Electronic Engineering and Computer Science, QMUL. Her current research interests include intelligent radio resource management for wireless networks, cognitive and cooperative wireless networking, mobile edge computing, HetNets, smart energy systems, and Internet of Things. She is in the editorial board of serving as an Editor of the IEEE COMMUNICATION LETTERS. She has served as a TPC Member for many IEEE conferences, such as GLOBECOM and ICC.



Lei Jiao received the B.E. degree in telecommunications engineering from Hunan University, Changsha, China, in 2005, the M.E. degree in communication and information system from Shandong University, Jinan, China, in 2008, and the Ph.D. degree in information and communication technology from University of Agder (UiA), Norway, in 2012. He is now working at the Department of Information and Communication Technology, UiA, as an Associate Professor. His research interests include reinforcement learning, resource allocation and performance evaluation for communication and energy systems.