IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received April 9, 2019, accepted May 5, 2019, date of publication May 17, 2019, date of current version June 4, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2917564

Q-Learning Based Content Placement Method for Dynamic Cloud Content Delivery Networks

YUJIE LIU, DIANJIE LU, GUIJUAN ZHANG, JIE TIAN, AND WEIZHI XU
College of Information Science and Engineering, Shandong Normal University, Jinan 250014, China
Shandong Provincial Key Laboratory for Distributed Computer Software Novel Technology, Jinan 250014, China

Corresponding author: Dianjie Lu (ludianjie@sdnu.edu.cn)

This work was supported by National Natural Science Foundation of China (No. 61572299, 61801278, and 61602285), Shandong Provincial Natural Science Foundation of China (No. ZR2019MF017, and ZR2017QF008), Funding for Study Abroad Program by the Government of Shandong ProvinceïijŇShandong Provincial Scientific Research Programs in Colleges and Universities (J18KA310).

ABSTRACT How to reduce the content placement cost of cloud content delivery networks (CCDNs) is a hot topic in recent years. Traditional content placement methods mainly reduce the cost of content placement by constructing delivery trees, but they cannot adapt to the dynamic deployment of cloud proxy servers in the CCDNs. In addition, the traditional content placement method only provides delivery paths according to local decision-making without considering global dynamics of the congestion in the CCDNs, which is also one of the main factors causing high cost of content placement. To solve these problems, we propose a content placement model based on Q-learning for the dynamic CCDNs, called Q-content placement model (Q-CPM). This Q-learning approach can lead to better routing decisions due to up-to-date and more reliable congestion values. Then, based on the Q-CPM model, an algorithm is proposed to construct the Q-adaptive delivery tree (Q-ADT). In this algorithm, local and nonlocal congestion information is propagated over network learning packets. Through this algorithm, the paths with low congestion cost will be selected and can adapt to the dynamic cloud delivery environment. The experimental results show that the method can adapt to the dynamic changes of the CCDNs flexibly and reduce the overall congestion cost of content placement effectively.

INDEX TERMS Content placement, dynamic CCDNs, congestion information, Q-learning.

I. INTRODUCTION
By deploying a large number of data centers and edge servers, the traditional content delivery networks (CDNs) release the replicas to the ‘‘edge’’ of the network where users can obtain the content as close as possible and avoid the congestion of the central server [1], [2]. However, the traditional CDN systems are difficult to expand since it has high deployment cost. With the development of cloud computing, resource leasing (such as storage and bandwidth) is allowed to build CDNs in the cloud, called cloud content delivery networks (CCDNs) [3]. Content providers (CPs) can rent cloud resources or leverage the services provided by public CCDN infrastructure providers (e.g., Amazon CloudFront and Google Cloud CDN) to build their own CCDN. CCDNs leverage the flexibility of the cloud to distribute content on the Internet quickly and easily. Amazon provides CloudFront [4] as a content distribution service, and CloudFont also provides a pay-as-you-go model where customers can register their own servers to build their own CDN. MediaWise Cloud [5] provides a new cloud coordination framework that any user can become a CDN provider and offer a pay-as-you-go model. Unlike Amazon CloudFront's use of its own cloud services, MediaWise Cloud leverages several public cloud providers to deliver services. The CCDNs is able to enhance the scalability, the flexibility and the elasticity while lowering the cost of content storage and delivery [6].

The associate editor coordinating the review of this manuscript and approving it for publication was Muhammad Maaz Rehan.

Content placement is one of the key technologies in CCDNs. To save the cost of content placement, the CPs usually reduce the number of replicas by building the multicast delivery trees [7]. However, most delivery tree building methods aims at static networks and cannot adapt to the dynamic characteristics of frequent changes of cloud proxy server in CCDNs. In addition, network congestion is also one of the main factors that cause the high cost of traditional content placement methods of CCDNs. Traditional content placement method only provides delivery paths according to local decision-making without considering the global dynamics of the congestion in the CCDNs. Therefore, in view of these dynamic features of CCDNs, how to reduce the cost of content placement is still a challenging problem.

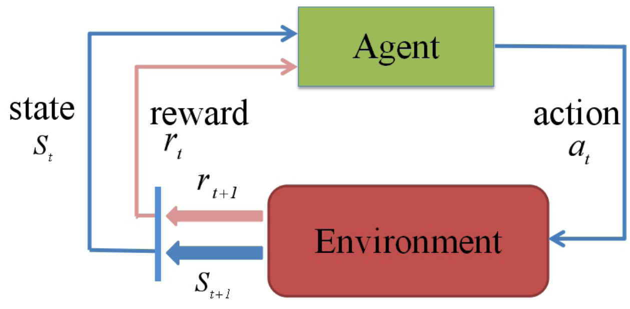

FIGURE 1. Reinforcement learning framework.

The reinforcement learning method is a solution to the global optimization by learning its own experience. The reinforcement learning task includes two main subjects, Agent and Environment, where Agent is the learner and also the decision maker. The learner achieves his goal by interacting with the environment. The interaction process is shown in Figure 1. From the figure, it can be seen that this is a serialization process. At time t, the learner sends out action a_t based on the current state s_t, the environment responds, and generates new states s_{t+1} and corresponding returns r_{t+1}, where states s and returns r appear in pairs. Reinforced learning methods can acquire knowledge in the environment of action evaluation, and constantly improving the action plan to adapt to the environment while facing the dynamic selections [8]. Q-learning is a typical method in reinforcement learning which has been widely used [9]. Based on Q-learning, the Q-routing algorithm [10] is proposed for packet routing of the dynamic network. Compared with traditional routing methods, this method can balance the load of dynamic networks better and provide a more reasonable solution for content delivery.

In this paper, a content placement model based on Q-learning for dynamic CCDNs, called Q-content placement model (Q-CPM), is proposed. Based on the Q-CPM, an algorithm is proposed to construct the Q-adaptive delivery tree (Q-ADT). In this algorithm, the local and nonlocal congestion information can be derived by the learning packets. The local congestion information indicates the time taken for the packet to be transmitted from a cloud proxy server to its neighbor. The nonlocal congestion information gives a global view of the congestion conditions from the current cloud proxy server to the terminating proxy server. Each cloud proxy server stores the information at Q-table and makes the update using learning packets. Therefore, the algorithm can adapt to the dynamically changing network and select a path with low congestion cost to deliver data.

The remainder of the paper is organized as follows. We review related studies in Section II. In Section III, we introduce the Q-CPM model which is a novel content placement model for the dynamic CCDNs. Furthermore, we present an algorithm to construct Q-ADT which is a kind of adaptive delivery tree in Section IV. Then, we show the simulation results in Section V to evaluate our proposed algorithms. Finally, we give concluding remarks as well as future directions in Section VI.

II. RELATED WORK

Information delivery is a critical research topic in the future network [11], [12]. As one of the most important technology of information delivery, content placement has been widely studied by many researchers. Kangasharju *et al.* [13] formulated the content placement as a combinatorial optimization problem and provided an optimal replica placement solution. Jia *et al.* [14] proposed two heuristic algorithms to solve the replica placement problem by constructing the derivation tree. Wendell *et al.* [15] proposed a distributed server selection mechanism that can effectively balance the load between edge servers. Aram *et al.* [16] studied the problem of optimal replica server deployment and content placement in the urban content delivery network and proposed an optimization design to minimize the cost of server deployment. Xu *et al.* [17] proposed a joint optimization method to reduce the request load of content distribution network. However, the above methods need to deploy a large number of edge servers with high cost.

As cloud computing develops, CPs are able to build their own CCDN by leasing cloud resources. Recently, extensive research efforts related to CCDNs have focused on content placement optimization. Haghighi *et al.* [18] and Zeng *et al.* [19] proposed a QoS based greedy heuristic algorithm to optimize the placement of replicas in CCDNs. Broberg and Tari [20] proposed a low-cost delivery mechanism by placing content in different cloud storage networks and then redirecting requests of users to the corresponding replicas. Hu *et al.* [21] formulated the content placement as an optimization problem to reduce the overall flow and the delay. To reduce the cost of content placement in CCDNs, many scholars have done a lot of research. Salahuddin *et al.* [22] and Lin *et al.* [23] designed and implemented a cloud storage-based content delivery framework to assist with replica placement to achieve optimal content delivery on the cloud under diverse needs, effectively reducing the content placement cost. Lu *et al.* [24] proposed a session-based cloud video delivery network framework for the dynamic characteristics of mobile users in cloud video delivery networks which effectively reduced the cost and improved the performance of the algorithm. Documents [25] and [26] propose efficient cache placement strategy in wireless content delivery networks. Zheng and Zheng [27] propose an improved heuristic genetic algorithm for static content delivery in cloud storage and obtain optimal content delivery program. However, all these methods have low scalability since they are introduced without considering the changes of users' demands sufficiently.

To solve this problem, most CPs use a multicast tree structure to enhance the scalability while reducing the network overhead in the content delivery processes. Pierre *et al.* [7] proposed a tree building algorithm and designed a content replacement strategy based on the hot part of the replica. This method has good scalability, but it does not consider the influence of network congestion. Gong *et al.* [28] proposed a multicast tree building method based on the Steiner tree in order to achieve lower cost and lower power consumption. Aibin *et al.* [29] focus on a problem related to joint optimization of unicast, anycast, and multicast traffic. They proposed several heuristic algorithms and the results show that the multicasting provides the best performance in content delivery than unicasting and anycasting. Due to the inherent uncertainty in network links, Fu *et al.* [30] model the network as an uncertain graph, determines the link between the source destination nodes, and reduces the content delivery path detection cost as a whole. Nowadays, CPs can lease cloud resources to dynamically establish cloud based content delivery networks. However, traditional content delivery methods are mostly directed to static networks with unchanged network status and cannot be directly applied to the dynamic CCDNs.

Reinforcement learning method can adapt to dynamic network scenarios. For example, Narayanan and Jagannathan [31] proposed an optimal adaptive scheduling method based on hybrid Q-learning for large-scale uncertain interconnected systems. Alsheikh *et al.* [32] mapped the stochastic networks into the Markov decision process (MDP) models and proposed various reinforcement learning methods to solve them. In order to reduce the traffic load in future cellular networks and achieve better caching efficiency, Gu *et al.* [33] propose a distributed cache replacement strategy based on Q-learning. Psounis *et al.* [34] formulated the problem of obtaining an optimal replacement algorithm with respect to hit rate as a Markov decision process, and obtained an online replacement algorithm that is optimal conditioning on the state of the Markov process. However, few of existing work investigates the content placement optimization using the reinforcement learning method in dynamic CCDNs. Because of the dynamic characteristics of cloud proxy server in CCDNs and the dynamic changes of network congestion, the network connection is uncertain, and we cannot know the real situation of network connection. By applying Q-learning method to CCDNs, learning knowledge can be acquired through trial and error learning, and the next routing decision can be determined based on the information previously learnt. So it can adapt to the changing environment.

In this paper, a novel content placement model based on Q-learning is proposed for dynamic CCDNs. This model uses the Q-learning approach in order to make better routing decisions since it can derive the up-to-date information of network topology and congestion. Furthermore, we present an adaptive delivery tree construction algorithm based on the proposed content placement model. The algorithm propagated local and nonlocal congestion information over the network learning packets. Through this algorithm, a path with low congestion cost is selected and can adapt to the dynamic cloud delivery environment.

III. CONTENT PLACEMENT MODEL FOR DYNAMIC CCDNs

Traditional CCDNs network architecture consists of user, source server and cloud proxy server. As shown in Figure 2, the source server sends a copy of the data to the cloud proxy server. When the user accesses the data content, the cloud proxy server near the user preferentially provides the data content to the user.

In this part, we analyze the dynamic characteristics for content placement of CCDNs and introduce our proposed content placement model. First of all, we define a list of notations that will be used throughout the paper, as shown in Table 1.

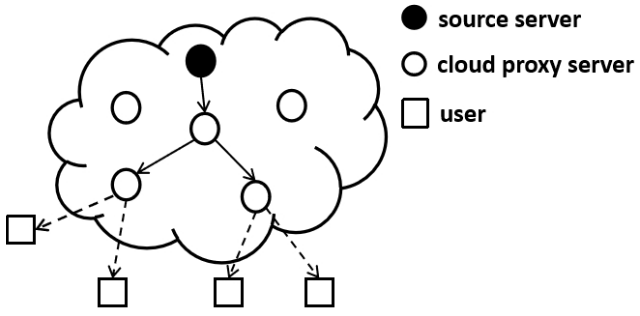

FIGURE 2. Traditional CCDNs architecture.

TABLE 1. Notations used in this paper.

Notation	Descriptions
S	the state space
A	the action set
A_s	the set of all optional actions in state s
$R: S \times A$	the reward function
$Q_i(k,d)$	the estimated congestion cost from node i to termination node d through node k
f	the best congestion cost estimation
$N(i)$	the neighbor of node i
t_i	the queuing time of the request packet in the queue of node i
δ	the time taken for the packet to be transmitted between two nodes
γ	the discount factor
T	a certain time interval
dis	the distance from the sender to the destination node
O	the source server node
D_j	the destination node j
$q(D_j)$	the node sequence from node D_j to its terminating node

A. DYNAMIC CHARACTERISTICS OF CCDNs

First, we take the multicast delivery tree as an example to look into how the content placement methods work. In CCDNs, we classify the cloud servers into three types, including the source servers, the destination cloud proxy servers and the

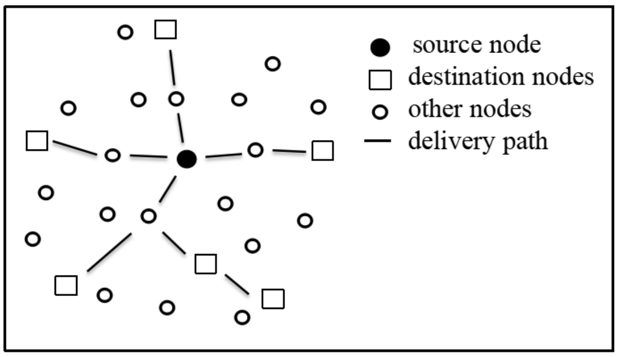

FIGURE 3. Traditional delivery tree structure.

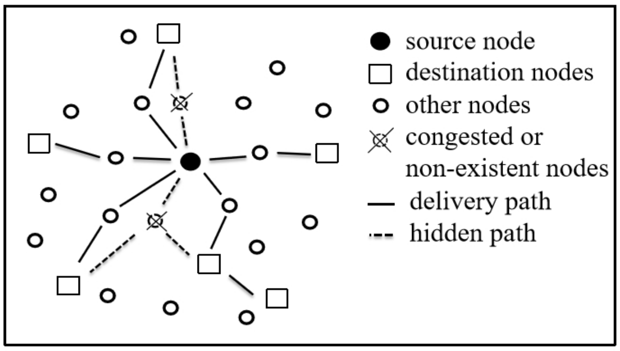

FIGURE 4. Adaptive delivery tree structure.

intermediate cloud proxy servers. Initially, the source servers store the original data and place the content to the destination cloud proxy servers. At the same time, the intermediate cloud proxy servers can participate as relay servers in the process of content placement, but the intermediate cloud proxy servers do not store content. For convenience, we consider only one source server and call the source server as the source node. Similarly, we call all cloud proxy servers as nodes and the intermediate cloud proxy servers as relay nodes. As shown in Figure 3, a delivery tree originated from the source node is built based on a static topology. When the network scenario remains unchanged, the scheduled tree can achieve an effective delivery with low cost. However, the network scene is dynamic in CCDNs.

The dynamic characteristics of CCDNs can be summarized as follows.

- Dynamics of the topology: CPs can lease cloud proxy servers when needed and terminate leasing when not needed. Thus, the allocation and de-allocation of the cloud proxy servers make the topology of CCDNs change dynamically.
- Dynamics of the network congestion: In CCDNs, the network congestion changes with the number of packets processed by the cloud proxy servers. It is difficult to obtain the global information of congestion for the traditional content placement methods. They can only provide the relatively fixed delivery paths based on local decision-making.

Unfortunately, distributing data along the scheduled delivery tree will result in a higher congestion cost since it cannot adapt to the dynamics of the topology and congestion in CCDNs well. Thus, the content placement algorithm that comprehensively consider these dynamic factors should be tailored for CCDNs.

To this end, a novel content placement model based on Q-learning (Q-CPM) is proposed. Our Q-CPM uses an adaptive delivery tree structure which adjusts the strategy constantly according to the dynamic network situation. Figure 4 shows the structure of the adaptive delivery tree. The black circle in the figure represents the source node. The rectangle represents the destination node and the small circle represents the other nodes in the network. The virtual circle indicates the node which has been removed or has a high congestion. The delivery tree connected by the dotted line is a potential solution, that is, the path may be selected if these nodes exist or remain low congestion. Our adaptive delivery tree structure can adjust the path (e.g. the delivery tree is formed by the solid lines) according to the existence of the node and the congestion status of the node, thus maintaining a low congestion cost.

B. Q-LEARNING BASED CONTENT PLACEMENT MODEL

In this part, we propose a Q-learning based content placement model, called Q-CPM. First, we define a CCDN as a directed graph $G = (V, E)$, where V is the vertex set which represents the nodes in the network, and E is the edge set which represents all possible links in the network. Next, the relevant terminologies are given as follows.

Definition 1 (State Space): The state space can be denoted by S where a state $s \in S$ is transferred to the next state s' by taking an action a. The state of satisfying the termination condition is called a termination state. In our problem, each step of the decision-making process requires the current node to select the next neighbor node to send the data. So, the current node (e.g. $v \in V$) can be looked as the current state (e.g. $s = v$) and the next node (e.g.$v' \in V$) as the next state (e.g. $s' = v'$). Specially, the state arriving at the termination node (in Definition 2) is the termination state.

The way we construct the distribution tree is that each destination node requests to establish a connection with other destination nodes, and finds the destination node whose distance is closer to the source node than itself. If there is no such destination node, it will directly establish a connection with the source node. When all destination nodes establish a connection, it will construct a distribution tree. We define the node that satisfies the connection condition with the destination node as the termination node (in Definition 2).

Definition 2 (Termination Nodes): If the node is the source node or other destination node that has obtained a copy of the data and is closer to the source node than the sender, we refer to this node as the terminating node of the sender.

Definition 3 (Action Set): In Q-learning algorithm, the action set is the set of all actions that the current state reaches the next state. In our problem, the path selection from the current node to the next node is the action selection. All optional paths of the node constitute the action set A. And we have $A = \cup_{s \in S} A_s = E$ where A_s represents the set of all optional edges in state s.

Definition 4 (Rewards): In Q-learning, $R : S \times A$ is used to represent the reward function. If (s, a) is transferred to the next state s', then the reward function can be written as $r\left(s'|s, a\right)$. In our problem, if the current node j reaches the next node i through path selection, the reward function of the process is represented by $r\,(j, i)$.

Based on Q-learning, we learn the state of the network by Q value and then use the Q value to make routing decisions. Each node j in the network represents its own view of the network state through its Q-table. The action at node j is to find the neighbor node with the lowest expected congestion cost to deliver the data packet, so that the congestion cost of the request to its terminating node is the lowest. In order to ensure that each step of the path selection process has congestion awareness, we define the congestion cost estimation as follows.

Definition 5 (Congestion Cost Estimation): We denote $Q_i\,(k, d)$ as the congestion cost estimation of node i transmitting the request packet to the terminating node d passing through node k. When the request packet is sent from node j to node i, node i sends its best congestion cost estimation f for the terminating node d back to node j in the form of learning packets. Therefore, the best congestion cost estimation can be defined as

$$f = min_{k \in N(i)} Q_i\,(k, d)\,, \tag{1}$$

where $N\,(i)$ represents the neighbor of node i.

Definition 6 (Reward Function): For the reward function r, we consider the network congestion situation by introducing the queuing time of packets. Then, the reward from state to state can be written as

$$r\,(j,i) = \delta + t_i, \tag{2}$$

where t_i denotes the queuing time of the request packet in the queue of node i and δ denotes the time taken for the packet to be transmitted from node j to node i.

Definition 7 (Q Value Updating): The Q value is updated by the following formula

$$Q_j\,(i, d)_{new} = Q_j\,(i, d)_{old} + \gamma\left(f + r\,(j, i) - Q_j\,(i, d)_{old}\right) \tag{3}$$

where γ [35] is the discount factor.

IV. ADAPTIVE DELIVERY TREE (Q-ADT) BUILDING ALGORITHM BASED ON Q-CPM

Based on the Q-CPM mentioned above, we design an adaptive delivery tree building algorithm (Q-ADT) for CCDNs. First, we define a Q-table and introduce two kinds of data packets for Q-ADT. Next, a Q-learning based adaptive routing algorithm is used to select a path from each destination node to its termination node. Then, the corresponding Q-value in Q-table is updated by the Q-value updating algorithm according to the feedback learning packets. Finally, the path from the termination node to the destination node is found by reverse routing information and the Q-ADT is constructed.

TABLE 2. Q-Table.

Current Node D_1			
Next Node	Path	Q value	Termination Node
1	$(D_1,1)$	3.5	S
2	$(D_1,2)$	4.7	S
D_2	(D_1, D_2)	2.5	D_2
3	$(D_1,3)$	5.2	S

A. Q-TABLE AND PACKETS IN CCDNs

In our Q-learning based algorithm, we use a table to store the Q value, called Q-table. We set up a Q-Table for each node in CCDNs. As shown in Table 2, each node's Q-Table consists of four fields: Next Node, Path, Q value, and Termination Node. The Next Node field corresponds to the next state space of Q-learning. The Path corresponds to the action space, representing the path from the current node to the next hop. The Q value represents the expected congestion cost from the node to the destination node through the next hop. The Termination Node corresponding to the node that arrives at the termination state and the current node represents the current state.

There are two kinds of packets in the network, request packets and learning packets. The request packet format is:

$$< sender\ location, node\ sequence, next\ hop >$$

The *sender location* is the geographic location of the sender. The *node sequence* is the sequence of nodes through which the request packet passes, and the *next hop* field is the next hop node selected by the current node according to the Q-table. The learning packet format is:

$$< sender\ location, respondent\ location, path\ delay, future\ delay, termination\ node\ flag >$$

where *sender location* indicates the location of the sender. *respondent location* is the location of the node that sends the learning packet. *path delay* is the delay from the last hop to the current node, it's equivalent to the δ mentioned above. *future delay* represents the estimated value of future congestion cost from the current node to the destination node, it's equivalent to the f mentioned above. *termination node flag* indicates whether the current node is the termination node. If it is, set the value to 1, otherwise to 0.

Algorithm 1 Q-Learning Based Adaptive Routing

```
For the node $i$ that received the request packet
    $termination\ node\ flag \leftarrow 0$
Add itself to the node sequence
 $path\ delay \leftarrow$ Delay sent from last hop to node $i$
IF node $i$ is the source node
    termination node flag $\leftarrow 1$
    Send a learning packet to the last hop
Else IF node $i$ is the destination node
    $SenderSourceDist \leftarrow$ distance from sender
    to source node
    $NodeSourceDist \leftarrow$distance from node
    $i$ to source node
    IF $NodeSourceDist < SenderSourceDist$
    && Existing data
        termination node flag $\leftarrow 1$
        Send a learning packet to the last hop
    Else
        future delay$\leftarrow \min_{k \in N(i)} Q_i(k, d)$
        Send a learning packet to the last hop
        Continue to send request packets to next hop $k$
Else
    future delay$\leftarrow \min_{k \in N(i)} Q_i$ (k,d)
    Send a learning packet to the last hop
    Continue to send request packets to next hop $k$
```

B. Q-ADT BUILDING ALGORITHM

During the building of Q-ADT, each destination node is used as a sender to send the request packet. So, a path connecting it to the terminating node can be derived. When all destination nodes find paths to connect themselves to their termination nodes, an adaptive delivery tree is constructed and the connections among the source node and all the destination nodes can be achieved.

We divide the building process of Q-ADT into three phases. The first phase is responsible for path selection. In this phase, each destination node is used as a sender and the neighbor node with the smallest Q value is selected to send the request packet. As shown in Algorithm 1, when a node receives a request packet, it adds itself to the sequence of nodes and calculates the delay δ sent from the last hop to the local node. Then, it determines whether it is the terminating node of the destination node. If so, set the *termination node flag* to 1 and send the learning packet to the last hop according to the node sequence. If not, calculate the future delay according to Equation (1), and send the learning packet to the last hop. Then, we keep looking for the next hop according to the Q-table. The operations are repeated until it reaches the terminating node. In Algorithm 1, "Existing data" means the node has obtained a copy of the data.

The second phase is responsible for the learning phase. As shown in Algorithm 2, node j sends the request packet to the downstream node i firstly. If the node j receives

Algorithm 2 Q-Value Update

```
Node $j$ sends a request packet to its downstream node $i$
IF the learning packet from node $i$ is received within time
interval $T$
    $\delta \leftarrow$ path delay
    $f \leftarrow$ future delay
    $t_i \leftarrow$Wait time in the packet queue of node i
    IF $termination\ node\ flag$
        $r(j, i) = r(j, i) - 1/dis^2$
        Update $Q_j(i, d)$ according to Equation (3)
    Else
        $r(j, i) = \delta + t_i$
        Update $Q_j(i, d)$ according to Equation (3)
Else
    $r(j, i) \leftarrow r(j, i) + c$
    Update $Q_j(i, d)$ according to Equation (3)
    $k = \arg\min_k Q_j(k, d)$
    Resend the request message to the downstream node $k$
```

the learning packet sent by i in a certain time interval T, it judges whether i is the terminating node according to the *termination node flag*. If so, we change the reward function to $r(j, i) = r(j, i) - 1/dis^2$($dis$ indicates the distance from the sender to the terminating node. $-1/dis^2$ ensures that the smaller the distance, the smaller the return value, and the easier it is to select the path to that terminating node the next time. It can also ensure that the sender eventually finds a terminal node closest to itself). Then, the Q value is updated according to Equation (3). If it is not a terminating node, the queuing waiting time of the request packet in node i is calculated. The reward function is calculated by the *path delay* and *future delay* feedback from the learning packet according to Equation (2).

If the node j does not receive the learning packet sent by the node i in the time interval T, the node i is considered to be absent and the node j changes the reward function to $r(j, i) = r(j, i) + c$(c is a large constant, so that we can get a large value of the reward function, make the corresponding Q value larger, and avoid congestion caused by sending data to the node next time). Then, the Q value is changed according to the Equation (3) and the downstream node k is reselected to send the request packet according to the Q-Table.

For the calculation of the reward function, we can further explain in Figure 5, the reward function calculation can be divided into three situations:

1) As shown in Figure 5, if the node k receives the learning packet sent by source node O in a certain time interval T and O is the terminating node of sender D_j, then k calculates the reward function according to $r(j, i) = r(j, i) - 1/dis^2$.
2) If the node k receives the learning packet sent by node i and i is not the terminating node of sender D_j, then k calculates the reward function according to $r(j,i) = \delta + t_i$.

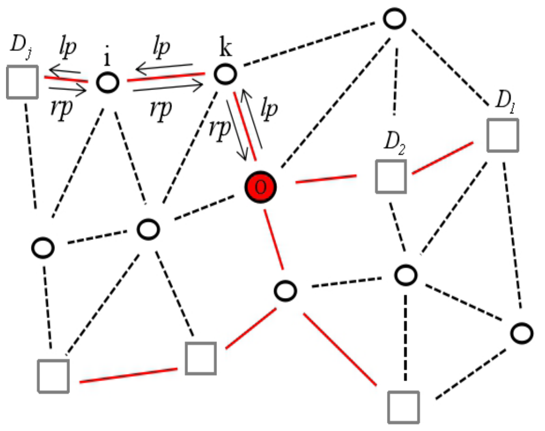

FIGURE 5. Example diagram of the Q-ADT build process.

3) If the node k does not receive the learning packet sent by the node i in the time interval T, then k calculates the reward function according to $r(j, i) = r(j, i) + c$.

The third phase focus on the delivery tree construction. If the destination node finds a path to its terminating node, its terminating node sends data to the destination node through a reverse route according to the node sequence. Finally, when all destination nodes find paths to the termination node, a Q-ADT is constructed.

Figure 5 is a sketch of the Q-ADT building process, where O represents the source server node and $D_1, D_2, \ldots, D_j$ represents the destination node in the network. The circle represents the relay nodes in the network. *rp* represents the request packet, *lp* represents the learning packet, and the red solid line represents the selected adaptive path. In the algorithm of this paper, each destination node sends a packet to request a path to the termination node.

The routing process is explained in detail by taking Figure 5 as an example:

1) The node D_j is used as the sender to select the neighbor node i with the smallest Q value according to the Q-table to send the request packet.
2) After receiving the request packet, the neighbor node i sends the learning packet to D_j, including its best estimate f to the terminating node d and the delay δ from the D_j to the node i.
3) After receiving the learning packet sent by the neighbor node i, D_j calculates its congestion cost function according to Equation (2), and updates the corresponding congestion cost estimate of the node i in the Q-Table according to Equation (3).
4) The neighbor node i selects the next hop node according to the Q-table, and continues to send the request packet to the next hop node until reaching the terminating node.

When all the destination nodes find a path to their terminating node through the congestion-aware process, a Q-ADT is constructed.

V. EXPERIMENTAL ANALYSIS

We tested the performance of the experiment in different network topologies. We select 5 destination nodes each time, set the parameters γ to 0.9. The time interval T is set to the maximum time for feedback learning packets when the network load is heavy, and this paper is set to 0.01s. The c is set to a larger constant, and in this paper 10 is selected as the c value. Here we set a reasonable value according to the actual situation. We define the total congestion costs for data delivery as follows:

$$Cost = \sum_{D_j \in D} \sum_{i \in q(D_j)} r\left(D_j, i\right), \tag{4}$$

where $q\left(D_j\right)$ is the node sequence from node D_j to its terminating node.

Formally, given a network $G = (V, E)$, the weight of each edges, and a set of terminals $D \subseteq V$, the Steiner tree problem is to find a tree in G that spans D with the minimum total weight. This problem has been proven to be NP-hard. For the Steiner tree problem, the best one can hope for is an approximation algorithm with a small but constant approximation guarantee. Many people model delivery tree optimization problems as Steiner tree problems. The Toward Source Tree (TST) [28] is a Steiner tree approximation algorithm to generate approximate Steiner Trees. Therefore, we compare the proposed algorithm with TST. Simulation results show that Q-ADT can effectively reduce the congestion cost of data delivery, and can better adapt to different network load conditions compared with TST.

A. CONGESTION COST ANALYSIS

This paper tests our algorithm using the communication network topology used by Boyan and Littman [36], including an irregular 6×6 network and a 116 node LATA communication network. Communication network is an abstract representation of real life system, such as Internet or transmission network. It consists of a set of isomorphic nodes and links between them. Figure 6 (a) is a schematic diagram of an irregular 6×6 network topology. The network consists of two well-connected parts and a bridge link between the two parts. The bridge link in the middle is prone to network congestion. Figure 6 (b) is a schematic diagram of a 116 node LATA communication network topology. Compared with the 6×6 network, the LATA communication network has an irregular network connection and the network is more complicated.

Figure 7 (a) is a comparison of the total congestion cost of Q-ADT and TST in an irregular 6×6 network. The load in this paper corresponds to the parameter value of the Poisson arrival process for the average number of injected packets per time unit. As can be seen, in Figure 7(a), the total congestion cost of TST is smaller than that of Q-ADT although the difference between them is not significant. As the network load increases, the congestion cost of TST increases significantly, much higher than Q-ADT, while the Q-ADT can also achieve lower congestion costs under high load conditions. In addition to the irregular 6×6 network, we also take a

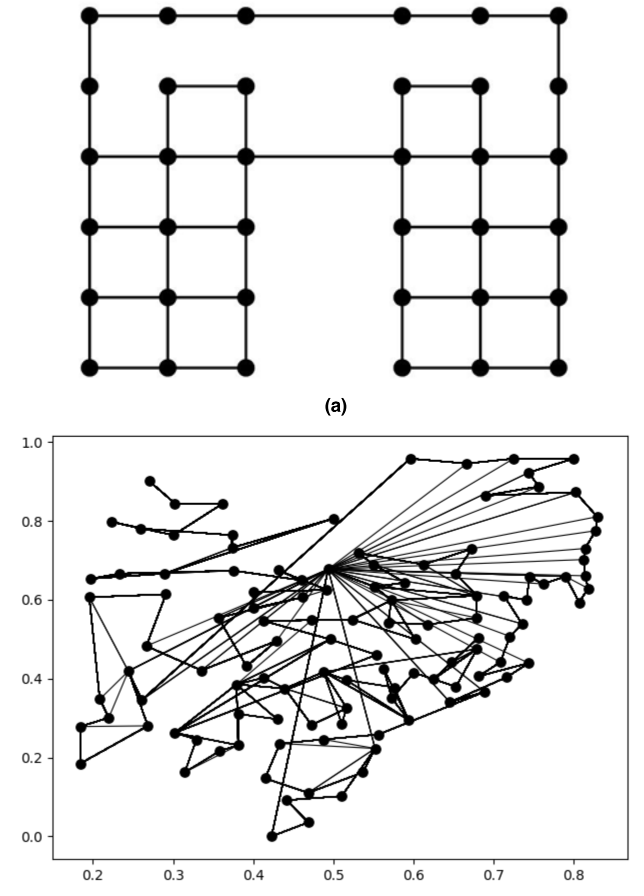

FIGURE 6. Two network topologies. (a) Irregular 6 × 6 network topology. (b) LATA communication network topology.

test in the 116-node LATA communication network. As can be seen in Figure 7(b), the Q-ADT still has a low congestion cost. It can be seen that Q-ADT can effectively reduce the congestion cost of the network with different network topologies.

B. EXPERIMENTAL PERFORMANCE ANALYSIS WITH DIFFERNET LOAD

As shown in Figure 8 and Figure 9, we compare the variation of the congestion cost over time between Q-ADT and TST under the LATA network topology. We conduct experiments under high and low loads. Low load means fewer data packets are injected per unit time and network congestion is small. High load means more data packets are injected per unit time and network node congestion is serious. From Figure 8, we can see that the congestion cost of Q-ADT converges to a certain value with low load when the time reaches 2000. However, the congestion cost of TST remains low. As shown in Figure 9, in the case of high load, the congestion cost of Q-ADT gradually converges to a smaller value as the load increases, while the congestion value of TST is always in a higher level. Therefore, TST cannot reduce network congestion globally when the load is high, and Q-ADT can obtain lower congestion cost in the long run regardless of the load.

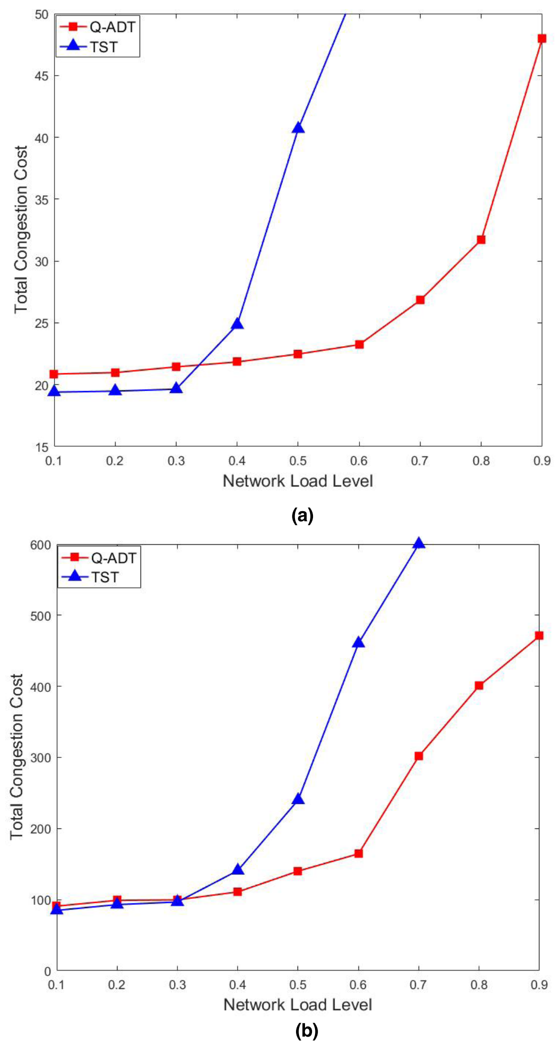

FIGURE 7. Congestion cost comparison. (a) Comparison of congestion cost for irregular 6 × 6 network. (b) Comparison of congestion cost for LATA communication network.

Figures 8 and 9 have peak patterns, this is because at the initial stage of the algorithm, the Q-table has not been updated, so the resulting congestion cost is not optimal before the Q-table is updated to the optimal state, and peaks may occur, as time increases, congestion costs gradually reach a steady state. Q-learning needs a learning phase as a kind of learning based algorithm. In this phase, the algorithm does not return any optimal result as already proven in theory. It reaches the steady state after such a learning phase, causing the optimal result. In Figures 8 and 9, we can see the characteristic.

C. TREE LENGTH ANALYSIS

Steiner Tree Problem is to find a tree with the minimum total weight. For the Steiner tree problem, the best one can hope for is an approximation algorithm with a small but constant approximation guarantee. Figure 10 shows the comparison

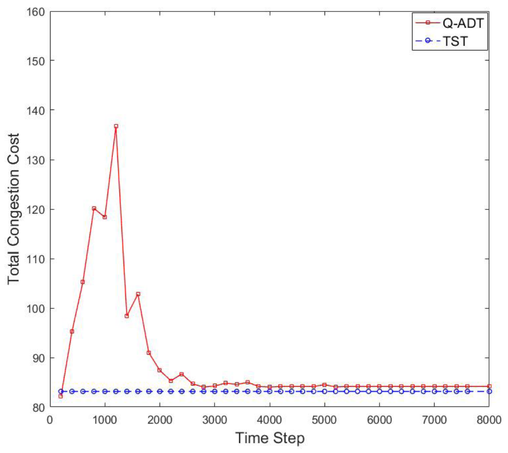

FIGURE 8. Comparison of congestion cost with low load.

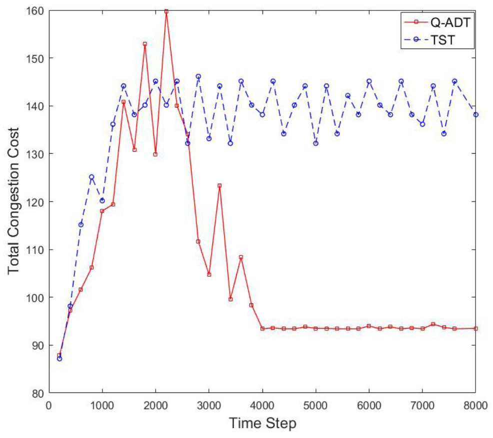

FIGURE 9. Comparison of congestion cost with high load.

between the tree length of the Q-ADT and TST as the load increases. It can be seen from the graph that the length of Q-ADT is basically the same as TST when the load is low. With the load increasing, the length of Q-ADT increases, indicating that the Q-ADT adjusts the path to accommodate increasing network congestion. Although the Q-ADT constructed by this paper is not as good as TST, it is very similar, indicating that the method can minimize the cost of congestion for content delivery along the delivery tree.

D. DISCOUNT FACTOR SELECTION

As shown in Figure 11, the congestion cost changes when the discount factor increases from 0.1 to 0.9. As can be seen from the figure, the congestion cost decreases with the increase of discount factor. When the discount factor is 0.9, the cost is the lowest. And the values in the experiment are based on actual conditions. Figure 12 shows the change of congestion cost over time when the discount factors are 0.5 and 0.9 respectively. It is shown that when the discount factor is taken as 0.5, the congestion cost does not change after the time reaches 6000. When the discount factor is set 0.9, the congestion cost will not change after the time reaches

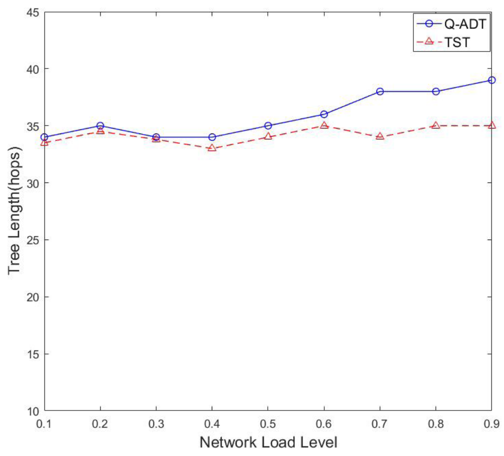

FIGURE 10. Tree length comparison.

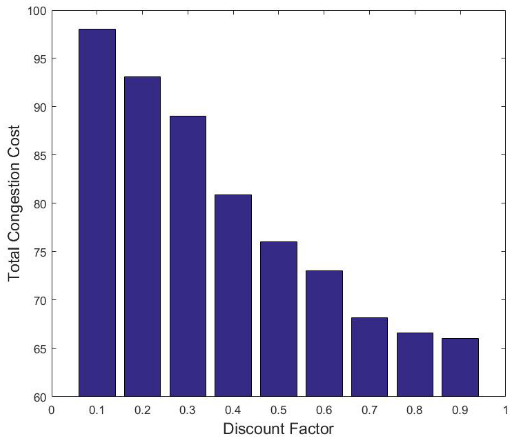

FIGURE 11. Comparison of congestion cost with different discount factors.

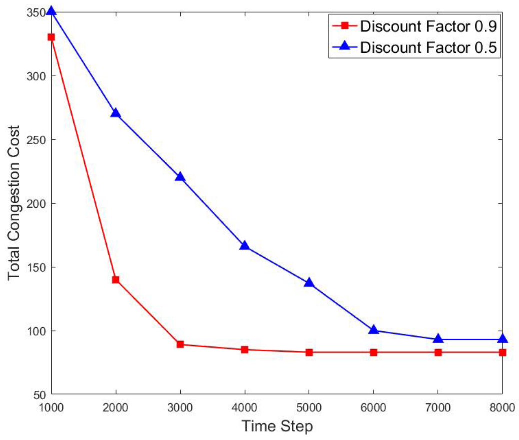

FIGURE 12. Changes of congestion cost over time with different discount factors.

3000. It can be seen that taking the appropriate discount factor can effectively reduce the convergence time of the algorithm.

VI. CONCLUSION

In order to reduce the cost of content placement in cloud content delivery network and adapt to the dynamic characteristics of CCDNs, we propose a content placement model based on Q-learning in this paper. This model uses the Q-learning approach which can lead to better routing decisions due to up-to-date and more reliable congestion values. So it can adapt to the dynamic characteristics of the CCDNs. Furthermore, we present an adaptive delivery tree construction algorithm to select the optimal paths with the lowest cost. The simulation results show that our method can adapt to the change of network and save congestion cost greatly. In future, we will further study the optimization of network performance jointly considering the consumption of energy and bandwidth.

REFERENCES

[1] B. M. Maggs and R. K. Sitaraman, "Algorithmic nuggets in content delivery," *SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 3, pp. 52–66, Jul. 2015.

[2] D. Lu, X. Huang, G. Zhang, X. Zheng, and H. Liu, "Trusted device-to-device based heterogeneous cellular networks: A new framework for connectivity optimization," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 11219–11233, Nov. 2018.

[3] F. Chen, K. Guoy, J. Liny, and T. La Porta, "Intra-cloud lightning: Building CDNs in the cloud," in *Proc. IEEE INFOCOM*, Mar. 2012, pp. 433–441.

[4] M. Wang, P. P. Jayaraman, and R. Ranjan, "An overview of cloud based content delivery networks: Research dimensions and state-of-the-art," *Transactions on Large-Scale Data- and Knowledge-Centered Systems* (Lecture Notes in Computer Science). Berlin, Germany: Springer, 2016, pp. 131–158.

[5] R. Ranjan, K. Mitra, and D. Georgakopoulos, "MediaWise cloud content orchestrator," *J. Internet Services Appl.*, vol. 4, no. 2, pp. 1–14, 2013.

[6] M. A. Salahuddin, J. Sahoo, R. Glitho, H. Elbiaze, and W. Ajib, "A survey on content placement algorithms for cloud-based content delivery networks," *IEEE Access*, vol. 6, pp. 91–114, Feb. 2018.

[7] B. Pierre, B. A. Walid, and G. Eric, "Cache location in tree networks: Preliminary results," in *Proc. Int. Conf. Netw. Optim.* New York, NY, USA: Springer-Verlag, 2011, pp. 517–522.

[8] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking deep reinforcement learning for continuous control," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1329–1338.

[9] J. Yan, H. He, X. Zhong, and Y. Tang, "Q-learning-based vulnerability analysis of smart grid against sequential topology attacks," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 1, pp. 200–210, Jan. 2017.

[10] Y. Shilova, M. Kavalerov, and I. Bezukladnikov, "Full echo Q-routing with adaptive learning rates: A reinforcement learning approach to network routing," in *Proc. IEEE NW Russia Young Res. Elect. Electron. Eng. Conf.*, Feb. 2016, pp. 341–344.

[11] G. Zhang, D. Lu, and H. Liu, "Strategies to utilize the positive emotional contagion optimally in crowd evacuation," *IEEE Trans. Affective Comput.*, to be published. doi: 10.1109/TAFFC.2018.2836462.

[12] D. Lu, X. Huang, W. Zhang, and J. Fan, "Interference-aware spectrum handover for cognitive radio networks," *Wireless Commun. Mobile Comput.*, vol. 14, no. 11, pp. 1099–1112, 2014.

[13] J. Kangasharju, J. Roberts, and K. W. Ross, "Object replication strategies in content distribution networks," *Comput. Commun.*, vol. 25, no. 4, pp. 376–383, Mar. 2002.

[14] X. Jia, D. Li, H. Du, and J. Cao, "On optimal replication of data object at hierarchical and transparent Web proxies," *IEEE Trans. Parallel Distrib. Syst.*, vol. 16, no. 8, pp. 673–685, Aug. 2005.

[15] P. Wendell *et al.*, "Donar: Decentralized server selection for cloud services," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4, pp. 231–242, 2011.

[16] O. V. Aram, S. Yousefi, and M. Solimanpur, "Joint server selection and replica placement in urban content delivery networks," *Int. J. Oper. Res.*, vol. 25, no. 3, pp. 288–306, 2016.

[17] K. Xu, X. Li, S. K. Bose, and G. Shen, "Joint replica server placement, content caching, and request load assignment in content delivery networks," *IEEE Access*, vol. 6, pp. 17968–17981, Mar. 2018.

[18] A. A. Haghighi, S. S. Heydari, and S. Shahbazpanahi, "Dynamic QoS-aware resource assignment in cloud-based content-delivery networks," *IEEE Access*, vol. 6, pp. 2298–2309, Dec. 2018.

[19] L. Zeng, S. Xu, Y. Wang, K. B. Kent, D. Bremner, and C. Xu, "Toward cost-effective replica placements in cloud storage systems with QoS-awareness," *Softw., Pract. Exper.*, vol. 47, no. 6, pp. 813–829, 2017.

[20] J. Broberg, R. Buyya, and Z. Tari, "MetaCDN: Harnessing 'storage clouds' for high performance content delivery," *J. Netw. Comput. Appl.*, vol. 32, no. 5, pp. 1012–1022, 2009.

[21] H. Hu *et al.*, "Community based effective social video contents placement in cloud centric CDN network," in *Proc. IEEE Int. Conf. Multimedia Expo*, Jul. 2014, pp. 1–6.

[22] M. A. Salahuddin, H. Elbiaze, W. Ajib, and R. Glitho, "Social network analysis inspired content placement with QoS in cloud based content delivery networks," in *Proc. IEEE Global Commun. Conf.*, Dec. 2015, pp. 1–6.

[23] C.-F. Lin, M.-C. Leu, C.-W. Chang, S.-M. Yuan, "The study and methods for cloud based CDN," in *Proc. Int. Conf. Cyber-Enabled Distrib. Comput. Knowl. Discovery*, Oct. 2011, pp. 469–475.

[24] D. Lu, B. Hu, X. Zheng, H. Liu, and G. Zhang, "Session-based cloud video delivery networks in mobile Internet," *J. Internet Technol.*, vol. 18, no. 7, pp. 1561–1571, 2017.

[25] J. Sung, M. Kim, K. Lim, and J.-K. K. Rhee, "Efficient cache placement strategy in two-tier wireless content delivery network," *IEEE Trans. Multimedia*, vol. 18, no. 6, pp. 1163–1174, Jun. 2016.

[26] J. Sung, K. Kim, J. Kim, and J.-K. K. Rhee, "Efficient content replacement in wireless content delivery network with cooperative caching," in *Proc. 15th IEEE Int. Conf. Mach. Learn. Appl. (ICMLA)*, Dec. 2016, pp. 547–552.

[27] Z. Zheng and Z. Zheng, "Towards an improved heuristic genetic algorithm for static content delivery in cloud storage," *Comput. Elect. Eng.*, vol. 69, pp. 422–434, Jul. 2018.

[28] H. Gong, L. Fu, X. Fu, L. Zhao, K. Wang, and X. Wang, "Distributed multicast tree construction in wireless sensor networks," *IEEE Trans. Inf. Theory*, vol. 63, no. 1, pp. 280–296, Jan. 2017.

[29] M. Aibin, R. Goścień, and K. Walkowiak, "Multicasting versus anycasting: How to efficiently deliver content in elastic optical networks," in *Proc. Int. Conf. Transparent Opt. Netw.*, Jul. 2016, pp. 1–4.

[30] X. Fu, Z. Xu, Q. Peng, L. Fu, and X. Wang, "Complexity vs. optimality: Unraveling source-destination connection in uncertain graphs," in *Proc. IEEE INFOCOM*, May 2017, pp. 1–9.

[31] V. Narayanan and S. Jagannathan, "Distributed adaptive optimal regulation of uncertain large-scale interconnected systems using hybrid Q-learning approach," *IET Control Theory Appl.*, vol. 10, no. 12, pp. 1448–1457, 2016.

[32] M. Abu Alsheikh, D. T. Hoang, D. Niyato, H.-P. Tan, and S. Lin, "Markov decision processes with applications in wireless sensor networks: A survey," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 3, pp. 1239–1267, 3rd Quart., 2015.

[33] J. Gu, W. Wang, A. Huang, H. Shan, and Z. Zhang, "Distributed cache replacement for caching-enable base stations in cellular networks," in *Proc. IEEE Int. Conf. Commun.*, Jun. 2014, pp. 2648–2653.

[34] K. Psounis, A. Zhu, B. Prabhakar, and R. Motwani, "Modeling correlations in Web traces and implications for designing replacement policies," *Int. J. Comput. Telecommun. Netw.*, vol. 45, no. 4, pp. 379–398, 2004.

[35] F. Farahnakian, M. Ebrahimi, M. Daneshtalab, P. Liljeberg, and J. Plosila, "Q-learning based congestion-aware routing algorithm for on-chip network," in *Proc. IEEE 2nd Int. Conf. Netw. Embedded Syst. Enterprise Appl.*, Dec. 2011, pp. 1–8.

[36] J. A. Boyan and M. L. Littman, "Packet routing in dynamically changing networks: A reinforcement learning approach," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 1994, pp. 671–678.

YUJIE LIU is currently pursuing the M.S. degree with the School of Information Science and Engineering, Shandong Normal University, Jinan, China. Her research interests include content delivery networks and cloud computing.

DIANJIE LU received the Ph.D. degree in computer science from the Institute of Computing Technology, Chinese Academy of Science, Beijing, China, in 2012. He is currently an Associate Professor with the School of Information Science and Engineering, Shandong Normal University, Jinan, China. His research interests include cognitive wireless networks, heterogeneous cellular networks, and cloud computing.

GUIJUAN ZHANG received the Ph.D. degree in computer science from the Institute of Computing Technology, Chinese Academy of Science, Beijing, China, in 2011. She is currently an Associate Professor with the School of Information Science and Engineering, Shandong Normal University, Jinan, China. Her research interests include distributed computing and computational intelligence.

JIE TIAN received the Ph.D. degree in communication and information systems from the School of Information Science and Engineering, Shandong University, China, in 2016. She is currently a Lecturer with the School of Information Science and Engineering, Shandong Normal University, Jinan, China. Her research interests include cross-layer design of wireless communication networks, radio resource management in heterogeneous networks, and signal processing for communications.

WEIZHI XU received the Ph.D. degree in computer architecture from the Institute of Computing Technology, Chinese Academy of Sciences. He was a Postdoctoral Researcher with the Institute of Microelectronics, Tsinghua University. He is currently an Assistant Professor with the School of Information Science and Engineering, Shandong Normal University. His research interests include video processing and high performance computing.