International Journal of Innovative
Computing, Information and Control
Volume **16**, Number **3**, June **2020**

 ISSN 1349-4198
pp. **1059–1075**

# A NEW COOPERATIVE CACHE OPTIMIZATION ALGORITHM FOR INTERNET OF VEHICLES BASED ON EDGE CLOUD NETWORK

Wei Bai*, Hongwu Zhang and Chen Chen

School of Mathematics and Computer Science
Ningxia Normal University
Xueyuan Road, Guyuan 756000, P. R. China
*Corresponding author: nx_bw@nxnu.edu.cn; { 89299306; nxsfcc }@nxnu.edu.cn

Received January 2020; revised April 2020

Abstract. *The cache capacity in the edge cloud is limited and it cannot provide a complete cache service for all vehicles. How to use the limited cache resources to minimize the delay of retrieving data has become the key of research. This research focuses on the cache resource allocation strategy based on edge cloud in urban crossroads scenario. Firstly, the scenario is described and modeled. Then, the above model is solved by using the link minimum cost cooperative cache algorithm based on Cournot game and the detailed flow of learning algorithm is given. In the simulation, the study builds a Long Short-Term Memory network (LSTM) neural network to predict the behavior of vehicles according to their driving data and then obtain the distribution of vehicle driving directions. Firstly, the real vehicle data was learned by using LSTM network to acquire the vehicle trajectory prediction model. Then, the performance of the proposed algorithm is verified by simulation and compared with several benchmark strategies. Simulation results show that the proposed algorithm can achieve higher returns than the benchmark strategies under different prediction accuracy rates and cache numbers. Meanwhile, it is also compared with several cache optimization algorithms. Simulation results indicate that the algorithm proposed in the paper is better in terms of average cache hit rate and average cache delay.*

**Keywords:** Cache optimization algorithm, Internet of Vehicles (IoV), Edge cloud, Cournot game, Long Short-Term Memory network (LSTM), Vehicle trajectory prediction

1. **Introduction.** Mobile Edge Computing (MEC) is an emerging technology proposed in recent years to meet the increasing computing needs of mobile applications. It was developed by the European Telecommunications Standards Institute (ETSI). MEC extends the boundaries of cloud computing to provide cloud service with higher computing power and lower delay by deploying services at the edge of the network near the user side [1-4]. With the development of the Internet of Vehicles (IoV), the growing demand for non-safety related businesses has brought tremendous pressure to the IoV. Due to the low capacity and stability of the wireless link and the fast-moving speed of the vehicle, the transmission of a large amount of data from the content provider server to the vehicle side takes a huge delay and reduces the quality of service. Besides, the frequent switching of vehicles between edge clouds has caused vehicles to establish links with remote servers frequently. It further increases the difficulty of data acquisition. To improve the service quality of data services, service providers can cache popular content to the edge cloud side by using a cache method. In this way, vehicles can directly obtain the required data from the edge cloud without accessing the remote cloud frequently.

DOI: 10.24507/ijicic.16.03.1059

Most of the services in the IoV are required to be completed within a low delay due to the high mobility of the vehicle. Besides, because the vehicle is equipped with a large number of sensor devices, it will generate large-scale data that needs to be analyzed and processed. And, the introduction of MEC in the Internet of Vehicles is an inevitable trend with the increasing demand for entertainment services. In particular, MEC is indispensable for autonomous driving technology in the IoV. Automated driving usually requires millisecond-level real-time control and a wider driving horizon, which can be achieved through MEC technology. Therefore, MEC provides strong support for autonomous driving technology. In a word, the combination of MEC and the IoV is the main direction and basis for future research on the Internet of Vehicles.

For the caching problem in the IoV, some articles mainly researched how to cache the data in the vehicle and transfer the data to other vehicles through the Vehicle to Vehicle (V2V) link when it meets other vehicles. For example, in [5], the authors proposed a data caching mechanism for Vehicular ad-hoc networks (VANETS). They cluster the vehicles and use V2V to continuously share data between vehicles. In this way, the data related to the area can be stably stored in the corresponding geographic area. [6] proposed a content distribution strategy based on edge cloud with the goal of saving energy. The authors used the theory of auction game to study the collaborative caching of edge nodes, considering cache loss and size and obtaining an edge node selection strategy with both performance and security.

In addition to the research of computing resource allocation algorithms, another type of resource optimization problem for IoV is to study the cache resource allocation strategy for cache applications. Many studies have focused on improving the Quality of Service (QoS) of content services using caching services in the IoV, while other studies have used pre-caching to improve the caching services of the IoV. By predicting vehicle mobility or the popularity of content, data service providers can cache data in advance. [7] modeled the vehicle mobility and studied the method of data offloading under different requirements. And it proposed a file placement algorithm in terms of the statistical priority of the data and user needs. In [8], the authors proposed a secure cache strategy based on the auction game for the disaster recovery problem of the IoV. The development of deep learning technology has greatly improved the accuracy of vehicle trajectory prediction. Therefore, many researchers have begun to use it to assist pre-caching strategy. Among them, a lot of researches are directed at the pre-caching problem in cellular networks. For example, [9] proposed a pre-caching algorithm based on Echo State Networks (ESNs). It performs cached distribution based on predictions of the content. The authors also proposed the pre-cache placement problem based on Unmanned Aerial Vehicle (UAV) in [10] to improve the QoS of the cache service in the cellular network.

Aiming at the problems of insufficient resources and single-point failure for single-point cache, researchers have proposed the method of collaborative caching. Generally, collaboration may occur between vehicles, or vehicle cloud networks, or even a vehicle cloud network and an edge cloud network within a vehicle cloud network. [11] studied collaborative caching strategies based on vehicle mobility prediction. The authors first pointed out that traditional collaborative caching strategies cannot be directly transferred to the Internet of Vehicles due to vehicle mobility. Then, the authors studied how to place caches and select nodes in hotspots where vehicles frequently visit and stay. [12] proposed an edge caching strategy based on cross-entropy to provide a cooperative caching method between base stations which improved the cache service QoS.

Based on the above research status, there is a large amount of literature on the Internet of Vehicles, but there are still some problems that need to be solved. Firstly, the traditional cloud computing architecture cannot meet the business needs of its application due to

the particularity of the Internet of Vehicles. A mobile cloud architecture for the Internet of Vehicles scenario is needed to integrate and manage various resources and the services that meet the needs of IoV applications are also necessary. Secondly, in the scenario with high-speed synchronous streaming, the vehicle needs to offload some computing tasks to surrounding vehicles, because the computing capacity of the vehicle is limited while the demand for multimedia services of vehicle users is high. In this process, how to ensure the delay requirements of the business is very necessary. It needs to adopt an appropriate computing resource allocation strategy. Thirdly, in the urban intersection scene, the data requested by users may not be acquired completely within the coverage of one base station because of the switch of vehicle between base stations. And in the coverage of the next base station, the user needs to obtain the remaining data from the remote cloud again, which leads to an increase of the acquisition delay. Therefore, the data is pre-cached to the target base station according to the driving trajectory which can reduce the delay of data acquisition efficiently. In the end, in the high-speed free flow scene, users may not be able to obtain complete data in a base station. At this time, the opposing vehicle and the base station can cooperate to cache and send the data to the target vehicle together when they meet the target vehicle. Thereby, it reduces the delay of vehicle users. In this process, the allocation of cache resources and communication resources will affect the final service delay. How to allocate resources reasonably has become the key to improving service quality. These existing problems of the IoV have driven us to carry out research work from the above aspects. The paper proposes a link minimum cost cooperative cache algorithm based on Cournot game. It fully considers practical factors such as transmission overhead, distribution of vehicle driving direction and delay. It also makes full use of the storage capacity of the edge server node to store user terminal requests, which reduces repeated downloads of the same content and link consumption efficiently. The main contributions of this study are as follows.

1) The cache resource allocation strategy of edge cloud network in the urban crossroads scene is studied and the real scene is described and mathematically modeled.

2) In the simulation experiment, the LSTM network was used to train and test 600,000 real data of vehicle trajectory in the real world and obtain the real results of vehicle trajectory prediction.

## 2. System Model.

2.1. **Scene description.** The scenario considered in this paper is an urban crossroads scene with $\boldsymbol{I} = \{1, 2, \ldots, t, \ldots, I\}$ fork road, as shown in Figure 1. Each fork is covered by a buffered edge cloud. The edge cloud is connected to the remote cloud through the backhaul link. $N$ vehicles are driven within the coverage of an edge cloud and the $N$ vehicles are about to cross an intersection. These vehicles make up a collection, denoted as $V = \{v_1, \ldots, v_n, \ldots, v_N\}$. Each element in $V$ represents the meta data of each vehicle, which is expressed in detail as $v_n = \left\{\overrightarrow{v_n}, \overrightarrow{a_n}, \theta_n\right\}$. $\overrightarrow{v_n}$ represents the driving speed and $\overrightarrow{a_n}$ represents the acceleration vector. In addition, $\theta_n$ indicates the azimuth of the vehicle $n$. These metadata can be used to make behavioral predictions about which direction the vehicle will go next. We define the probability distribution of the vehicle's next choice of $l$ as $F(l)$. Some vehicles may initiate requests of downloading data, such as getting a video.

To provide download services better, content providers can cache some data in the nearest edge cloud in advance. Assume that all potential content sets are $\boldsymbol{L} = \{L_1, \ldots, L_N\}$, where $L_i$ is the content size. With proper in-file encoding technology, a complete content can be divided into multiple independent fragments, which are initially stored in the

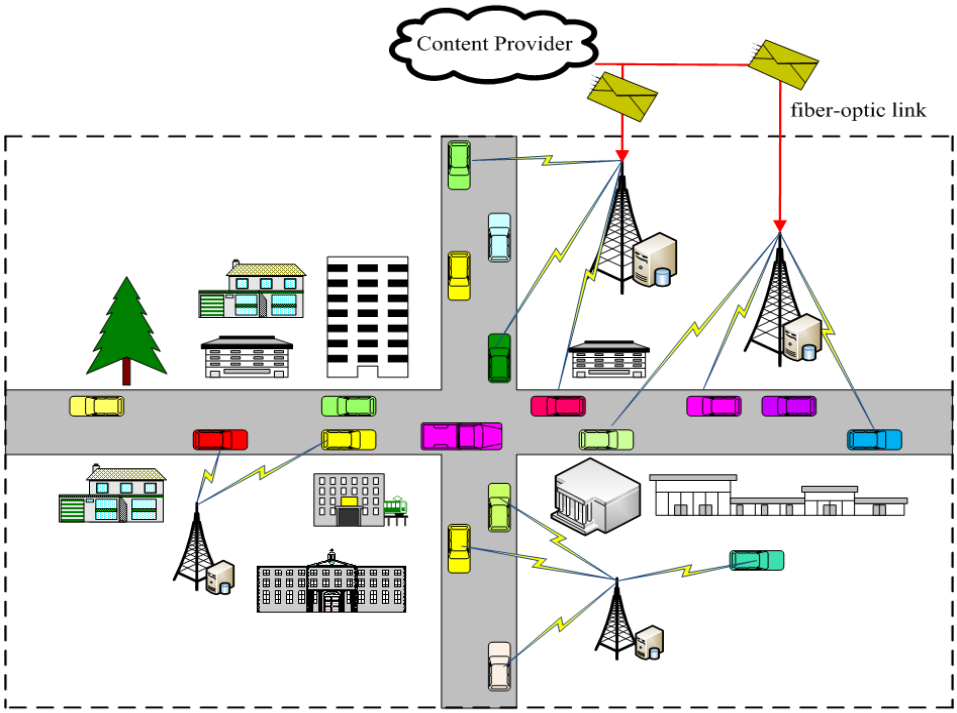


FIGURE 1. Schematic diagram of research on cache resource allocation strategy based on edge cloud network

content server of remote cloud network [13]. We suppose that the size of a single piece of content is $u$ and the number of fragments that can be split for each type of content is $M = \frac{L}{u}$, $L \in \boldsymbol{L}$. When a car makes a request to get some data, data may not be transmitted within the coverage of an edge cloud due to vehicle mobility. Therefore, content providers can send some fragments to the vehicle directly through the current edge cloud and then cache the remaining into the adjacent edge cloud [14]. When the vehicle reaches the next coverage of edge cloud, it can get the remaining data directly through the cache without visiting the remote cloud again. Assuming the number of cache units occupied by the request $i$ is $c_i$, the data of $uc_i$ size will be cached to the adjacent edge cloud while the data of $u(M - c_i)$ size will be directly transmitted to the vehicle. Our goal is to find a cache strategy to ensure lower delay of data acquisition by determining the location and size of the cache. Based on the above description, we give the model definition in the following.

## 2.2. **Model building.**

2.2.1. *State space.* The state space of the system can be expressed as follows:

$$s = \{s_1, \ldots, s_l, \ldots, s_I\} \tag{1}$$

Assume that all requests are processed sequentially, that is, during a time $\Delta t$ ($\Delta t \ll t$), there is one and only one request being processed. $s_l$ represents the number of cache units that have been occupied in the edge cloud $l$. We suppose that the state space of each system can be expressed as follows: each edge cloud has a maximum capacity of a cache, which is denoted as $K$ and $0 \leq s_l \leq K$. All possible states constitute the state space, denoted as $S$. We focus on a finite MDP problem where a system has $N$ time slices at most. The state of the system at time period $n$ is represented as $s^n$, and we can obtain the sequence of states $S = \{s^1, \ldots, s^n, \ldots, s^N\}$.

2.2.2. *Decision space.* The system can decide whether to cache content in a particular edge cloud. The behavior in a determined state can be characterized as:

$$a(s) = \begin{cases} 0, & \text{no cache} \\ (l, c), & \text{edge cloud } l \text{ will cache } c \text{ units} \end{cases} \tag{2}$$

where the distribution of $l$ obedience is what we are about to learn by using neural networks. We denote $\boldsymbol{F}(\boldsymbol{I})$ as this distribution and then $l \sim \boldsymbol{F}(\boldsymbol{I})$. Besides, $1 \leq c \leq K - s_n$. All feasible actions constitute a set of actions, recorded as A. The number of elements in A is represented as A.

2.2.3. *State transition.* In this model, when the state $s^n$ and behavior $a^n$ are determined, the state $s^{n+1}$ of the next stage can be determined. The state transition equation is expressed as:

$$s^{n+1} \leftarrow h\left(s^n, a\right) = \{s_1, \ldots, s_l + c, \ldots, s_N\} \tag{3}$$

where $a = (l, c)$.

2.2.4. *Profit.* The profit $Q$ is a mapping of space $S \times \mathrm{A}$ to the real number domain, that is $Q : S \times \mathrm{A} \to \mathbb{Q}$. In this system, we mainly take the negative value of the time loss for vehicles to obtain the data they need as a benefit. There may be three situations here: cache hits, cache misses, and no cache. Firstly, each vehicle takes some time to initiate a request, and this time is recorded as $\tau$. Note that $\tau$ can be subdivided into two parts: $T = \tau_1 + \tau_2$, where $\tau_1$ represents the transmission delay of the service request of V2I and $\tau_2$ represents the delay between the edge cloud and the remote cloud. It can be seen that $\tau_1 \ll \tau_2$. We assume that the data volume of the request packet is $Q$ and $C_1$ and $C_2$ are the transmission rates of the V2I and the backhaul link respectively.

$$C_1 = B \log(1 + x) \tag{4}$$

Here, $x$ is the signal-to-noise ratio of the V2I link and $B$ is the bandwidth. The backhaul link is an optical fiber link. We suppose that the transmission rate is fixed. However, there is a complicated routing and forwarding process in the remote cloud network and assume that the delay consumed by this process is $\tau_1$. Then we can get:

$$\tau_1 = \frac{Q}{C_1} \tag{5}$$

$$\tau_2 = \frac{Q}{C_2} + \tau_1 \tag{6}$$

If the content is not cached or a cache misses, the vehicle needs to re-initiate the request to the content server after entering the adjacent edge cloud. In addition, all content is obtained from Internet content providers, which will lead to huge delays and poor quality of service [15]. We take the complete time spent in this case as $t_m$, which can be expressed as:

$$t_m = 2\tau + \frac{L}{C_1} + \frac{L}{C_2} \tag{7}$$

And if the cache hits, the vehicle can obtain the previous part of the data like other vehicles firstly. Then, the remaining data is obtained directly from near edge clouds. At this time, the time consumption can be calculated as:

$$t_h(a) = 2\tau_1 + \tau_2 + \frac{uc}{C_1} + u(M - c)\left(\frac{1}{C_1} + \frac{1}{C_2}\right) \tag{8}$$

In this way, the time spent by each car can be characterized as $t_n(a) = t_h o(a) + t_m(1 - o(a))$, where

$$o(a) = \begin{cases} 1, & \text{cache hits and } c > 0 \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

We use constant $\beta$ to represent time value. The revenue function can be defined as:

$$r(s, a) = E - \beta t_n(a) \tag{10}$$

Here, $E$ is a large positive number. The revenue function indicates that the smaller the delay for vehicle users to obtain data, the greater the revenue.

2.2.5. *Target.* The cache policy is defined as a mapping of $S$ to A, that is

$$\pi : S \to \mathrm{A} = \{a_1, \ldots, a_n, \ldots, a_N\}, \quad s^n \in S \tag{11}$$

The strategy gives the probability distribution of the behavior in the state $s^n$, which can be expressed as:

$$\pi(a|s) = P\left[A_n = a \,|S_n = s\right] \tag{12}$$

In this model, the caching strategy is not affected by the environment. Therefore, it is a deterministic strategy and $\pi(a|s) = \pi(s)$. For a certain caching strategy, we can determine the entire system state and income sequence, that is,

$$\{s_1, \pi(s_1), \tau_1, \ldots, s_N, \pi(a_N), r_N\} \tag{13}$$

We denote that $G_n^\pi$ is the total discounted revenue obtained by taking strategy $\pi$ from state $s_n$ to end state $s_N$, then

$$G_n^\pi = r_n + \gamma r_{n+1} + \cdots + \gamma^{N-n} r_N \tag{14}$$

Our goal is to find an optimal strategy $\pi^*$ to make $G_n^{\pi^*} = \sup\left\{E_\pi\left(G_n^\pi\right)\right\}$.

3. **Cache Resource Allocation Strategy of Edge Cloud Network Based on Cournot Game.** This section will mainly introduce the collaborative caching process in edge computing, analyze the link cost incurred when a user terminal makes a content request and introduce a collaborative storage method between nodes to improve storage efficiency.

3.1. **Cache system model.** The cache architecture in edge computing is mainly divided into three parts: cloud content center, MEC computing node cluster and user terminal. Now suppose that in the case of only one cloud content center, the MEC computing service node cluster is composed of $F = \{F_1, F_2, \ldots, F_M\}$, where $F_i$ $(i \in 1, 2, \ldots, M)$ represents the MEC server node $i$. $U = \{U_1, U_2, \ldots, U_N\}$ represents a collection of user terminals, where $U_j$ $(j = 1, 2, \ldots, N)$ represents the $j$ user terminal.

Assume that in a given geographic range, the MEC service node which each user terminal belongs is divided according to the number of user terminals in the range. The number of user terminals is the number of vehicles $N$ and the number of MEC server nodes $M$. The MEC service is that the user's request will pass through its own MEC server node firstly. Considering the mobility between user terminals and the influence of social communication, this paper uses a social relationship-based division method to divide MEC server nodes for user terminals. The social relationship entity $E_{i,j}$ of user terminals $U_i$ and $U_j$ is calculated as follows:

$$E_{i,j} = \varpi \lambda_{i,j} + \varphi \frac{1}{1 + \mu D_{i,j}} \tag{15}$$

where $\mu$ is the scale factor. $\lambda_{i,j} \in [0, 1]$ represents the influence factor of the user terminal $U_i$ and $U_j$. $\varpi$ and $\varphi$ are the influence factor of the content information and the geographic

location respectively. $D_{i,j} \in \{0, 1\}$ indicates whether the user terminals $U_i$ and $U_j$ are within the influence range.

The cache size $C$ of the MEC server is divided into two parts: cooperative and non-cooperative (Figure 2). The part of non-cooperative $C(1 - R)$ is used to store frequently accessed content in the autonomous domain of the MEC server node while the cooperative part CR caches the content with high access frequency in the entire cache system by controlling the cooperation ratio $R$. Too small cooperation ratio $R$ will cause storage redundancy in the entire cache system and too large will cause the increasing cost of user terminal getting content through other MEC servers. The paper will focus on choosing a suitable cooperation ratio $R$ to reduce link overhead.

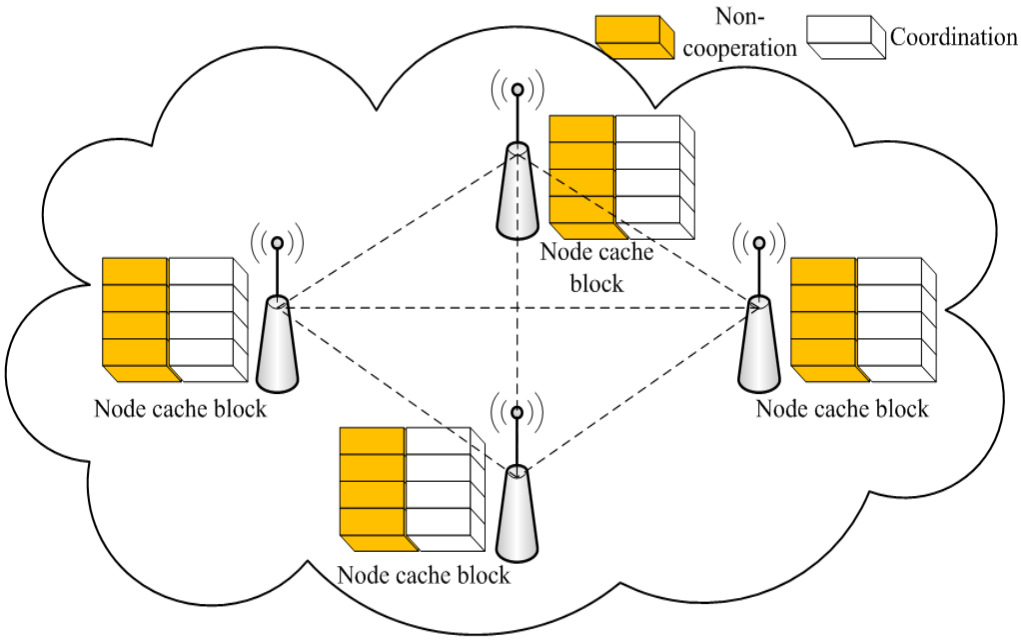


FIGURE 2. Composition of MEC server cache space

3.2. **Link cost analysis.** As can be seen from the model structure in the above section, there are mainly three cases where the user terminal obtains content sources: the cloud content center, the edge server node to which it belongs, and other collaborative parts. In this paper, the unit link transmission cost of user terminals accessing content through different content sources is different. The cost of unit link transmission is the extra consumption required for the transmission of 1-byte data in the network. It is assumed that $P_{i,j}$ is the unit overhead of the MEC server nodes $F_i$ and $F_j$. $P_i$ is the unit overhead of its autonomous domain and the unit overhead between the point of the MEC server section and the cloud content center is $P_c$, which is satisfied $P_c > P_{i,j} > P_i$.

The following discusses the cost of obtaining content from the above three cases.

1) MEC server node has no storage content $\vartheta$. User terminals will be able to get content through the cloud content center and the total overhead for requesting that content in a cache period is:

$$P_{total} = \sum_{i \in M} T_i^{\vartheta} C_{\vartheta} (P_c + P_i) \tag{16}$$

where $T_i^{\vartheta}$ $(i = 1, 2, \ldots, M)$ indicates the number of times that content $\vartheta$ is accessed on MEC server node $F_i$ and $C_{\vartheta}$ is the size of the content $\vartheta$.

2) Some MEC server nodes cache content $\vartheta$. Users can access content $\vartheta$ directly from the MEC server nodes they belong to. If its MEC server node has not stored content $\vartheta$, it obtains content from the nearest MEC server node. Then the total cost of getting the content is:

$$P_{total} = save\left(N^{\vartheta}\right) P_s C_{\vartheta} + \sum_{I \in M\left\{\backslash M^{\vartheta}\right\}, j \in M^{\vartheta}} T_i^{\vartheta} C_{\vartheta} (\min(P_{i,j}) + P_i) \tag{17}$$

where $M\left\{\backslash M^{\vartheta}\right\}$ represents the set of MEC server nodes without storing content $\vartheta$. $save\left(M^{\vartheta}\right)$ is the number of MEC server nodes that have stored the content $\vartheta$. $P_s$ represents the storage overhead of the unit data of the MEC server node and $\min(P_{i,j})$ is the minimum link cost between MEC server nodes.

3) Each MEC server node stores the content $\vartheta$. Users can access the content $\vartheta$ directly at the MEC server nodes they belong to, and then the total cost of acquiring the content is:

$$P_{total} = save(M) P_s C_{\vartheta} \tag{18}$$

Here, $save(M)$ is the number of MEC server nodes of the entire cache system.

3.3. **Calculation of cache value.** Based on the analysis of the link cost in the previous section, the reduced link cost for obtaining the content $\vartheta$ on the MEC server node $F_i$ in this section is called the cache value $\text{value}_{\text{i}}^{\vartheta}$ for caching the content. According to the storage situation of content $\vartheta$ in the entire cache system, the specific expression of its cache value $\text{value}_{\text{i}}^{\vartheta}$ is as follows.

1) Each MEC server node has not stored content $\vartheta$, so the cache value $\text{value}_{\text{i}}^{\vartheta}$ is:

$$\text{value}_{\text{i}}^{\vartheta} = T_i^{\vartheta} P_i + \sum_{j \in \left\{\backslash M^{\vartheta}\right\}} T_i^{\vartheta} (P_c + P_i - P_{i,j}) - P_s C_{\vartheta} \tag{19}$$

where $T_i^{\vartheta} P_i$ represents the saved overhead because user terminals in its autonomous domain of the MEC server node $F_i$ can obtain content $\vartheta$ directly. $\sum_{j \in \left\{\backslash M^{\vartheta}\right\}} T_i^{\vartheta} (P_c + P_i - P_{i,j})$ indicates the saved cost because the user terminals in its autonomous domain of MEC server node $F_i$ obtain content $\vartheta$ from other collaborative MEC server nodes. $P_s C_{\vartheta}$ is storage cost of MEC server node $F_i$ storing the content $\vartheta$.

2) Some MEC servers store content $\vartheta$, so the cache value $\text{value}_{\text{i}}^{\vartheta}$ is:

$$\text{value}_{\text{i}}^{\vartheta} = T_i^{\vartheta} \min(P_{i,j}) + \sum_{k \in \left\{\backslash M^{\vartheta}\right\}} T_i^{\vartheta} \left[\min(P_{k,j}) - \min(P_{i,j}) + P_i\right] - P_s C_{\vartheta} \tag{20}$$

where $T_i^{\vartheta} \min(P_{i,j})$ is the reduced cost that terminals in their autonomous domain of MEC server node $F_i$ obtain the content $\vartheta$ directly without accessing the content from other MEC server node $F_j$. The latter half is the cost saved by other MEC server node $F_k$ accessing the content from the MEC server node $F_i$ nearby. $P_s C_{\vartheta}$ represents storage cost of $F_i$ for storing content $\vartheta$.

3.4. **Link minimum cost caching algorithm.** The process of decision is shown in Figure 3. Each MEC server node caches some of the more popular content in the system by cooperating, and the cloud provides request content, computing and cooperative scheduling for the entire edge computing. The implementation of the cooperative scheduling for the Cournot game will be introduced in the next section.

In Section 3.1, it is described that the cooperation part of each MEC server node constitutes a cache space and a market monopoly model can be established for the cooperation cache space. That is, when different MEC server nodes do not store the same content

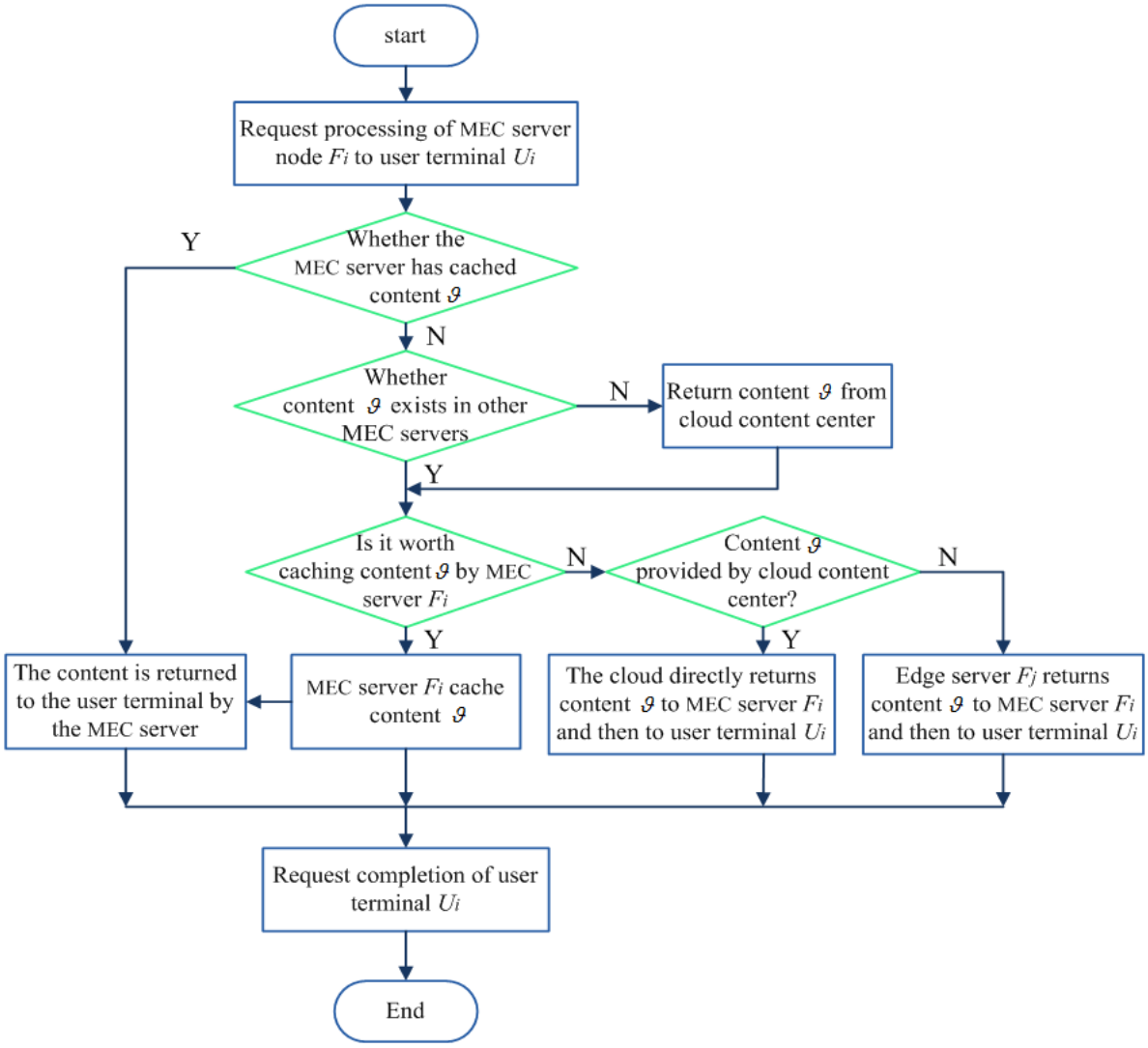


FIGURE 3. Cache decision process

required, the requested content can be regarded as a commodity. To ensure that the content is not repeatedly stored in the entire cache system, each MEC server node hopes to be able to store its cache space nearby and then the nodes can obtain the maximum benefit by competing. Finally, the cloud content center delivers the content to the MEC server node, which can minimize the link cost of the entire cache system. This is the typical Cournot game.

Based on the Cournot game, each MEC server node is set as the bidder. Each bidder gives its bid for the requested content, and the cloud content center acts as the content source provider to weigh the bids of each MEC server. Therefore, the corresponding pricing function is obtained as follows:

$$c(P) = \varpi + \varphi \left( \sum_{i=1}^{M} P_i \right)^{\rho} \tag{21}$$

where $\varpi$, $\varphi$ and $\rho$ all represent normal numbers and $P = \{p_1, p_2, \ldots, p_M\}$ is the bid set of different MEC server nodes. When $\tau \geq 1$, the pricing function is a convex function and it ensures that the cloud content center delivers the same requested content to the MEC server node that minimizes the link cost of the entire cache system.

From the link cost analysis of Section 4.2, the link cost saved by each MEC server node is:

$$\Upsilon^{save} = \Upsilon^{nocache} - \Upsilon^{cache} = n_i \cdot \frac{n_i^{cache}}{n_i} \cdot P_G \cdot C_i - c(P) \cdot d_i \tag{22}$$

where $n_i^{cache}/n_i$ is the proportion of the content stored in the MEC server node $F_i$ to the sum of the requested content in its autonomous domain. $P_G$ is the unit transmission overhead from the cloud content center to the user terminal, that is $P_G = P_i + P_c$. $c(P)$ represents the unit cache cost and $d_i = M_i \cdot C_i$ is the sum of size of the requested content.

According to [16], $n_i^{cache}/n_i$ of Formula (22) is converted into $\sum_{i=1}^{M_i} i^{-\sigma} \approx \frac{M_i^{1-\sigma}}{1-\sigma}$, where $\sigma$ is the distribution factor of content popularity. And the utility function based on the cache model of Cournot game of each MEC server node can be obtained.

$$\pi_i(P) = n_i \cdot \frac{M_i^{1-\sigma}}{(1-\sigma)\omega_K} \cdot P_G \cdot C_i - c(P) \cdot d_i \tag{23}$$

$\omega_K$ is the total probability of the requested content.

4. **Mobility Prediction Based on LSTM.** An LSTM neural network was constructed to predict the behavior of vehicles according to their driving data $v$, which can obtain a distribution of vehicle driving direction. The schematic diagram of the LSTM network is shown in Figure 4. This network takes account of the $I$ different bifurcation directions at the intersection, denoted as $D_I$. We use a vector $\mathbf{m}$ of length $I$ to express $D_I$. To make the weights in each direction be the same, the vector of direction $d \in \{1, 2, \ldots, I\}$ is expressed as:

$$\mathbf{m}_d = m_d(i) = \begin{cases} 0, & i \neq d \\ 1, & i = d \end{cases} \tag{24}$$

Under this definition, $\forall d_1, d_2 \in [1, I]$, $d_1 \neq d_2$, $\|\mathbf{m}_{d_1} - \mathbf{m}_{d_2}\|_2 = \sqrt{2}$. All $\mathbf{m}$ have the same Euclidean distance. The state of the $n$th car is $v^n$ and all elements in $v^n$ are used as feature vectors of the LSTM network. Our goal is to build a behavior map $F$ of vehicle from the time step $l$ to the current moment $t$:

$$m^t = F\left(x^{t-1}, x^{t-l+1}, \ldots, x^{t-1}\right) \tag{25}$$

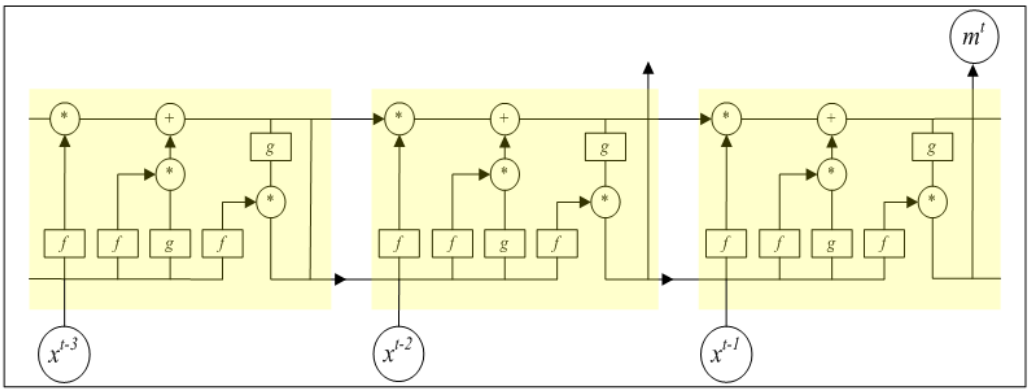


FIGURE 4. Schematic diagram of LSTM network structure

The LSTM network consists of three layers: the input layer, the hidden layer and the output layer. The number of neurons in the input layer depends partly on the number of elements of $v$. Each element of $v$ corresponds to a neuron of input layer. The other part depends on the LSTM itself. It will recursively use subsequent output as input at the current moment. The number of these neurons is configurable, as is the number of

neurons in the hidden layer. The neurons in the output layer are consistent with $I$. An example of the network structure is shown in Figure 4, where $f(\cdot)$ and $g(\cdot)$ are the gate structure of the LSTM network. To make predictions in different kinds of direction, we use cross-entropy as loss function of the network. Therefore, the loss function $J(\theta)$ can be expressed as:

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{I} y_k^{(i)} \log\left(h_\theta\left(x^{(i)}\right)_k\right) + \left(1 - y_k^{(i)}\right) \log\left(1 - \left(h_\theta\left(x^{(i)}\right)_k\right)\right)\right] \tag{26}$$

Here, $\theta$ is the set of trainable parameters. $m$ represents the number of samples and $I$ is the number of categories. $x$ and $y$ are input and output respectively. $h_\theta(x_k)$ is the Softmax function of input $x_k$, which shows the mean probability of $x_k$ in $I$ output. The expression of $h_\theta(x_k)$ is:

$$h_\theta(x_k) = \frac{e^{x_k}}{\sum_{i=1}^{I} e^{x_i}} \tag{27}$$

The main goal of LSTM is to minimize $\frac{\partial}{\partial\theta}J(\theta)$ and find the best $\theta$ through gradient descent. After sufficient training, the LSTM network can output $\boldsymbol{F}(\boldsymbol{I})$ for subsequent learning processes. The prediction accuracy of the network shows the hit probability of cache accurately because the cache position is determined by the prediction [17]. Assuming that the number of records and correct predictions in the test set are $T$ and $b$ respectively, we can calculate the prediction accuracy as $\eta = b/T$. In addition to behavior, the network also has an impact on system revenue, because high prediction accuracy means higher instantaneous revenue.

In this chapter, vehicle trajectory data is provided by a shared travel company Didi Chuxing in the real world. We used a total of 600,000 records as training and test set. The raw data is the Global Positioning System (GPS) information of vehicles traveling on the roads of city X, and the sample interval is one GPS data every 3 seconds. We calculate the distance of latitude and longitude by making a difference between the front and back of the vehicle and using the spherical hemi-squared formula. The formula is as follows:

$$d = \frac{r\pi}{180}\arcsin\left(\sqrt{\sin^2\left(\frac{\varphi_2 - \varphi_1}{2}\right) + \cos(\varphi_1)\cos(\varphi_2)\sin^2\left(\frac{\lambda_1 - \lambda_2}{2}\right)}\right) \tag{28}$$

After obtaining the distance, we can find out the average speed, acceleration and direction angle during the time interval. Based on the above three types of data, we mark the vehicle as four behaviors: going straight, turning left, turning right and turning around. Finally, we get the data form $\mathrm{v} = (v_{long}, v_{lat}, \alpha_{long}, v_{lat}, \theta)$ that we needed. To improve performance, the number of vehicles for each behavior was balanced. The data set was finally divided into three subsets, 80% of which were used for training, 10% for cross-validation and the last 10% for testing.

Then, we built a four-layer LSTM network to train the above data. The input layer includes five data neurons and three historical outputs. We use the Adam optimizer as the gradient descent algorithm. The detailed parameters of the LSTM network are shown in Table 1. The entire network is built on Keras of the TemorFlow framework.

## 5. **Simulation Experiment.**

### 5.1. **Simulation experiment.**

After using the LSTM network to obtain the distribution of vehicle driving directions, we performed simulation experiments. Relevant parameter values are as follows: $I = 4$, $K = \{5, \ldots, 15\}$, $\beta = 1$, $N = 20$, $\tau_1 = 1$s, $B = 5$MHz, $E = 15$, $L = 10$MB, $\mu = 1$MB, $x = 10$dB. After training, we simulated the $N = 20$ car and 100

TABLE 1. The value table LSTM network parameter based on cache resource allocation strategy of edge cloud network

| LSTM network layer | Number of neural clouds | Activation function |
|---|---|---|
| Input layer | 3 * 5 where 5 is the number of input nodes and 3 is the time step | – |
| Hidden layer I (LSTM) | 50 | ReLU |
| Hidden layer II (LSTM) | 100 | ReLU |
| Hidden layer III (LSTM) | 200 | ReLU |
| Output layer | 4 output node represents $m^t$ | Softmax |

cycles to evaluate performance. In each cycle, the system attempts to service download requests for all vehicles and records the total revenue of the system. In addition, the behavior of the system is also recorded for subsequent analysis. We selected the following benchmark algorithms for comparison.

1) Greedy strategy: The greedy strategy will choose the behavior that can reach the local optimum when making decisions in each state. With this strategy, edge cloud will try to cache as much content as possible for each vehicle and it ignores possible future requests. When the cache resources are sufficient, the greedy strategy will get the highest real-time benefits. However, with the large occupation of the cache, the cache resources will be consumed at a faster rate, which results in higher delay of the remaining vehicles. Finally, the total profit obtained by the greedy strategy may not be optimal.

2) No-cache strategy: In this strategy, content is not cached in the edge cloud, which causes severe degradation of service quality for vehicle users. They have to repeat the process of content request through Internet to get the remaining content from the content provider.

3) Full hit strategy: One of the goals of this research is to explore the impact of the LSTM network on the subsequent process of cache allocation. The best case comes when the prediction accuracy reaches 100%, which is usually impossible. When the prediction accuracy rate is 100%, the system can avoid wasting resources and obtain higher returns.

4) Minimum hit strategy: When the edge cloud cannot predict the direction of the vehicle at all, it can only assume that the vehicle is traveling in all directions with equal probability. Therefore, a discrete distribution $n^{\min} \sim D(I)$ is used here to replace $\boldsymbol{F}(\boldsymbol{I})$. The probability of each $\iota$ is $f(\iota) = \frac{1}{I}$. Then, the long-term expected return becomes the minimum value of the strategy.

## 5.2. **Results and analysis.**

5.2.1. *Prediction accuracy.* The prediction accuracy and loss of the LSTM network are shown in Figure 5. During the first few iterations, the LSTM network converged at a faster speed and the prediction accuracy also improved quickly. When the accuracy exceeds 0.696, the learning rate drops to a lower level, which also leads to a slowdown of the rate of improvement in prediction accuracy. For the total training, the highest prediction accuracy of LSTM can reach 86.7%.

5.2.2. *Average stage profit.* Figure 6 and Figure 7 show the average revenue of every stage for different amounts of cache capacity and prediction accuracy. As cache capacity increases, more cache resources can be allocated and the average return is higher. The minimum strategy has only slightly improved because of being affected by the accuracy of prediction. And for the impact of prediction accuracy, the worst case occurs when the

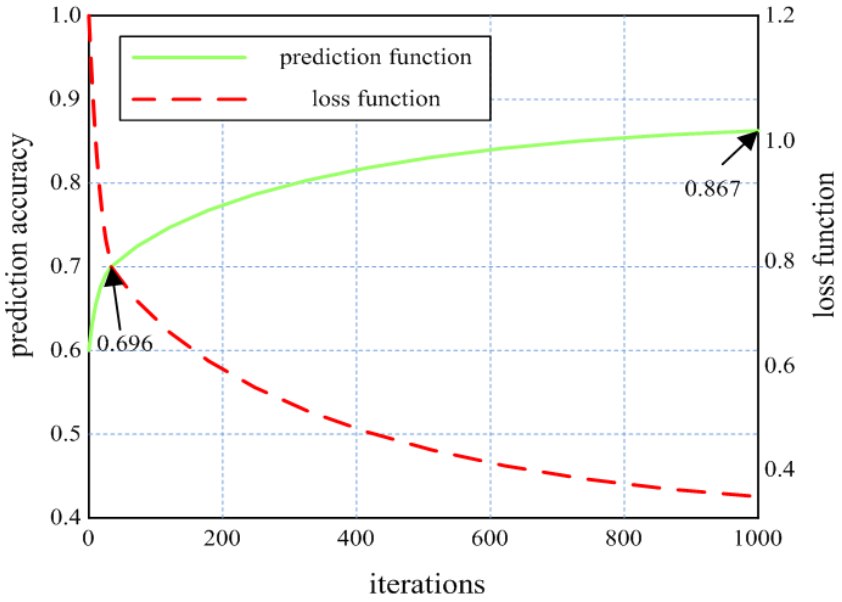


FIGURE 5. Prediction accuracy picture of vehicle trajectory

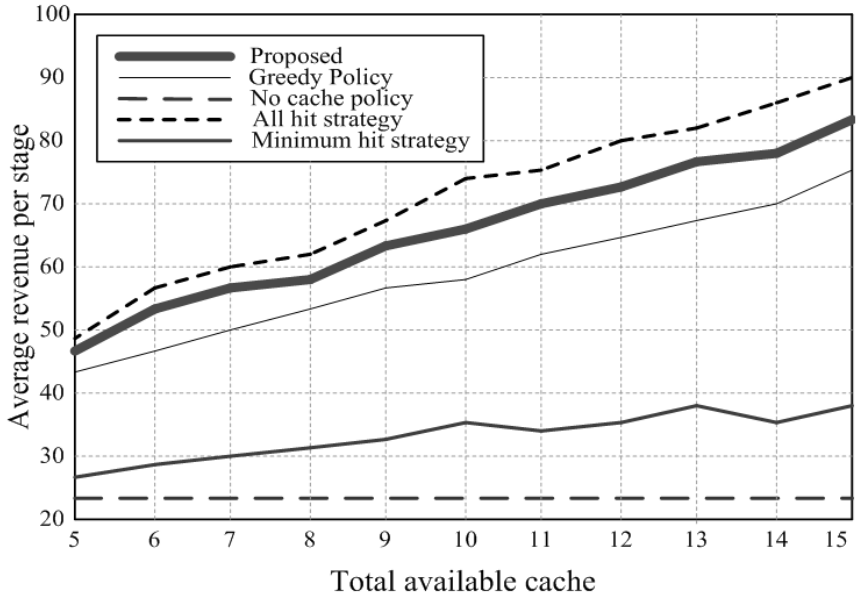


FIGURE 6. Profit of each stage with different number of caches

system has no cache. At this time the vehicle will have to consume $t_m$ time to obtain the data and the system will gain $r_{NC}$ benefits.

In this simulation, $r_{NC} = 20$ and it does not change with the prediction accuracy. The minimum strategy shows the performance of the system when prediction accuracy has no effect. In this case, the edge cloud of each direction has a probability $(1/I)$ of being selected to cache the data. And the hit ratio of cache is also $(1/I)$. This value is not enough to support the system to obtain higher revenue. However, the minimum strategy is still better than the no-cache strategy because the cache hit rate is low, but there is still a probability to hit which leads to a higher profit.

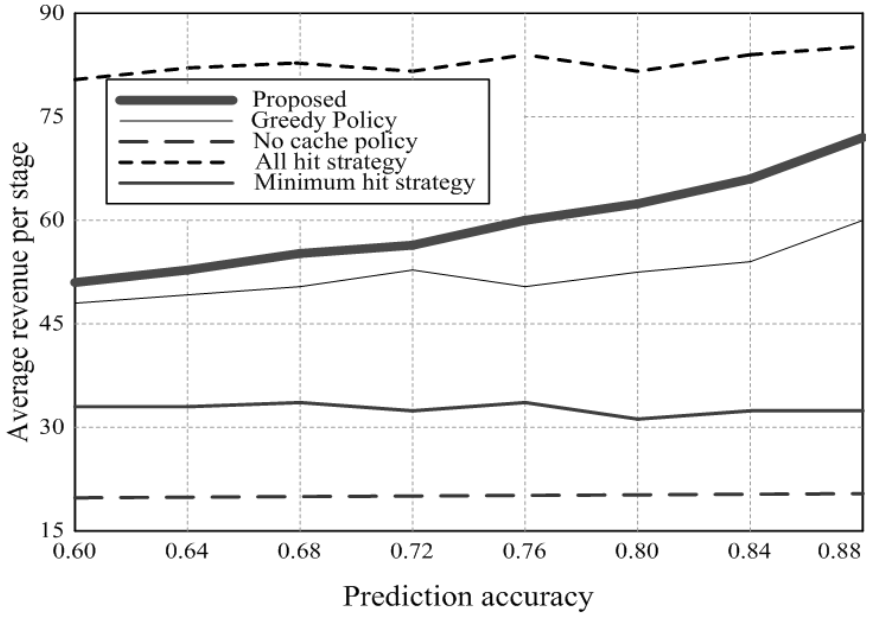


FIGURE 7. Profit of each stage with different prediction accuracy

The system can get the maximum benefit as long as the cache resources are sufficient when the cache hit rate reaches 100%. Then all vehicles can enjoy the cache service, and its overall revenue will naturally reach the maximum.

Besides, the paper also compares this strategy with greedy strategy, as is shown in Figure 7. As can be shown from the results, with the improvement of prediction accuracy, the system tends to cache more content which can reduce delay and increase revenue. This indicates that the prediction accuracy can affect the system revenue, and the greedy strategy has the same trend with the proposed strategy. In the all hit strategy, using more caches can reduce the latency of the system. However, this will increase the memory cost and easily lead to data inconsistency, which will affect the subsequent vehicles to enjoy the cache service. In addition, when the remaining resources are not sufficient, the proposed strategy will allocate caching more and more cautiously so as to obtain a higher overall revenue; thus the performance of the proposed strategy is better than the greedy strategy.

5.2.3. *Performance comparison with advanced algorithms.* To verify the effectiveness of the new collaborative cache optimization algorithm proposed in this paper, the algorithms of [6,8,12] are selected as comparison algorithms. It is assumed that all requests from all user terminals satisfy the Zipf distribution [18]. Meanwhile, to evaluate the performance of the algorithm, two parameters are defined as references: hit rate and cache delay. And the simulation parameters are as follows: $D = 1/\text{km}^2$, $\vartheta = 2500$, $F \in [2, 8]$, $N = 1000$, $R \in [0.4, 0.8]$, $C \in [100, 200]$ and parameter of Zipf $\sigma \in [0.7, 1.2]$.

As can be seen from Figure 8, the average cache hit rate of each algorithm increases as $\sigma$ increases, because when $\sigma$ increases, the probability of the vehicle terminal requesting the content increases and the frequency of replacement of the requested content by the MEC server node is reduced. Therefore, the hit rate raises. Compared with several other advanced algorithms, the new collaborative cache optimization algorithm proposed in this paper makes the hot content stay in the cache for a longer time to increase the cache hit rate through the collaborative cache between the MEC servers.

It can be seen from Figure 9 that the average buffer delay of each algorithm increases with decreasing of parameter $\sigma$ of Zipf. Because the parameter $\sigma$ increases, then the

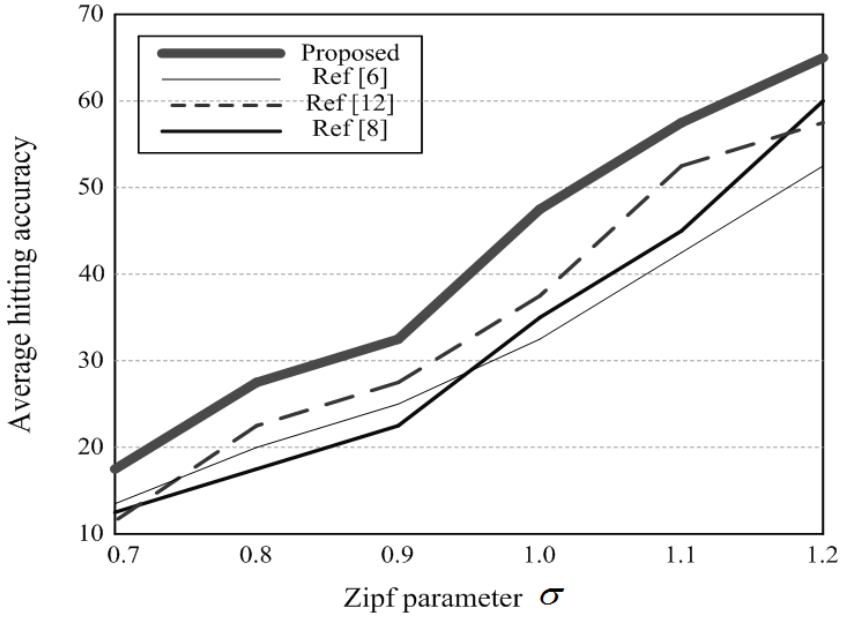


FIGURE 8. Comparison of cache hit ratios of different cache algorithms

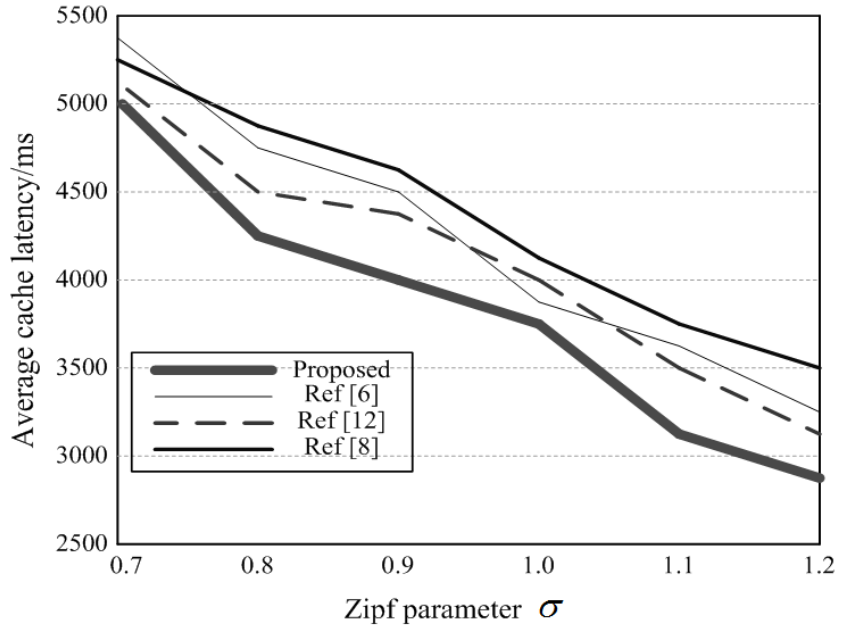


FIGURE 9. Comparison of cache delay between different cache algorithms

probability of the vehicle terminal requesting the same content raises. The corresponding content may be obtained directly from the local server node MEC, reducing transmission delay and link cost center incurred when acquiring content from the cloud content center. Compared with several other advanced algorithms, the new collaborative cache optimization algorithm enriches the content of the type of the entire cache system and reduces delay through cooperative caching.

6. **Conclusion.** This paper studies the cache resource allocation strategy based on edge cloud network in urban intersection scenarios, and models and analyzes the problem. Then, the link minimum cost cooperative cache algorithm based on Cournot game fully

considers the practical factors such as transmission cost, distribution of vehicle driving directions and delay. It obtains a better cache resource allocation strategy. In the simulation analysis, the paper uses LSTM network to analyze and predict the driving trajectory of real vehicles firstly. Based on the prediction results, the paper simulates and compares the performance of the proposed strategy and some benchmark strategies. The results show that the proposed strategy can obtain the highest benefits under different prediction accuracy and the number of caches.

However, due to the limitations of experimental scenarios and conditions, the resource optimization algorithms proposed in this paper have only analyzed performance through simulation of the scenario of IoV and the experiments have not been performed in real and complex scenarios. Experimental conditions in all aspects have been idealized or approximated. The object of this paper, that is, the particularity of the layered cloud architecture of the IoV, has no feasible and easy-to-use equipment for actual testing at this stage. Therefore, the performance verification of this paper does not have a measurement basis and it is difficult to verify the performance in actual scenarios. It can only be simulated through simulation. In future research, we also need to design, optimize and verify the algorithm for real and complex scenarios.

**Acknowledgement.** This research is supported by Funded project of first-class discipline construction (pedagogy discipline) of Ningxia institution of higher learning (Grant No. NXYLXK2017B11); Ningxia hui autonomous region – computer application technology funding project, Guyuan municipal science and technology program for industry (Grant No. 2019GKGY038).

## REFERENCES

[1] A. Mahmood, W. E. Zhang and Q. Z. Sheng, Software-defined heterogeneous vehicular networking: The architectural design and open challenges, *Future Internet*, vol.11, no.3, 2019.

[2] H.-C. Hsieh, C.-S. Lee and J.-L. Chen, Mobile edge computing platform with container-based virtualization technology for IoT applications, *Wireless Personal Communications*, vol.102, no.4, pp.1-16, 2018.

[3] J. Ahn, J. Lee, S. Park et al., Power efficient clustering scheme for 5G mobile edge computing environment, *Mobile Networks and Applications*, vol.24, no.4, pp.1-10, 2018.

[4] L. Fernández-Maimó, A. Huertas, M. G. Pérez et al., Dynamic management of a deep learning-based anomaly detection system for 5G networks, *Journal of Ambient Intelligence & Humanized Computing*, vol.10, no.8, pp.3083-3097, 2019.

[5] C. Wu, T. Yoshinaga, Y. Ji et al., A reinforcement learning-based data storage scheme for vehicular ad hoc networks, *IEEE Trans. Vehicular Technology*, vol.66, no.7, pp.6336-6348, 2017.

[6] Q. Xu, Z. Su, Q. Zheng et al., Secure content delivery with edge nodes to save caching resources for mobile users in green cities, *IEEE Trans. Industrial Informatics*, vol.14, no.6, pp.2550-2559, 2018.

[7] W. Shin, B.-Y. Min and D. K. Kim, VehiCaching: Embracing user request on vehicle route with proactive data transportation, *IEEE Vehicular Technology Conference*, pp.1-5, 2015.

[8] Z. Su, Q. Xu, J. Luo et al., A secure content caching scheme for disaster backup in fog computing enabled mobile social networks, *IEEE Trans. Industrial Informatics*, vol.14, no.10, pp.4579-4589, 2018.

[9] M. Chen, W. Saad, C. Yin et al., Echo state networks for proactive caching in cloud-based radio access networks with mobile users, *IEEE Trans. Wireless Communications*, vol.16, no.6, pp.3520-3535, 2017.

[10] H. Dai, Y. Yang, J. Yin, H. Jiang and C. Li, Improved greedy strategy for cloud computing resources scheduling, *ICIC Express Letters*, vol.13, no.6, pp.499-504, 2019.

[11] L. Yao, A. Chen, J. Deng et al., A cooperative caching scheme based on mobility prediction in vehicular content centric networks, *IEEE Trans. Vehicular Technology*, vol.67, no.6, pp.5435-5444, 2017.

[12] Z. Su, Y. Hui, Q. Xu et al., An edge caching scheme to distribute content in vehicular networks, *IEEE Trans. Vehicular Technology*, vol.67, no.6, pp.5346-5356, 2018.

[13] X. Wang, K. Wang, S. Wu et al., Dynamic resource scheduling in mobile edge cloud with cloud radio access network, *IEEE Trans. Parallel & Distributed Systems*, vol.29, no.11, pp.2429-2445, 2018.

[14] R. Muñoz, R. Vilalta, N. Yoshikane et al., Integration of IoT, transport SDN, and edge/cloud computing for dynamic distribution of IoT analytics and efficient use of network resources, *Journal of Lightwave Technology*, vol.36, no.7, pp.1420-1428, 2018.

[15] P. Bellavista, D. Belli, S. Chessa et al., A social-driven edge computing architecture for mobile crowd sensing management, *IEEE Communications Magazine*, vol.57, no.4, pp.68-73, 2019.

[16] G. P. Zhang, J. Gu, P. Liu et al., Cooperative communication strategy for wireless sensor networks based on cooperative game theory, *Journal of Wuhan University of Technology*, vol.32, no.9, pp.133-136, 2010.

[17] G. Jain, M. Sharma and B. Agarwal, Optimizing semantic LSTM for spam detection, *International Journal of Information Technology*, vol.11, no.2, pp.239-250, 2019.

[18] D. Niyato and E. Hossain, A game-theoretic approach to competitive spectrum sharing in cognitive radio networks, *IEEE Wireless Communications and Networking Conference (WCNC2007)*, pp.16-20, 2007.