A cache placement algorithm based on comprehensive utility in big data multi-access edge computing

Yanpei Liu*, Wei Huang, Li Han, Liping Wang
Zhengzhou University of Light Industry, School of Computer and Communication Engineering, Zhengzhou, 450002, China
[E-mail: liuyanpei@zzuli.edu.cn]
*Corresponding authors: Yanpei Liu

Received June 13, 2021; revised August 2, 2021; accepted August 29, 2021; published November 30, 2021

Abstract

The recent rapid growth of mobile network traffic places multi-access edge computing in an important position to reduce network load and improve network capacity and service quality. Contrasting with traditional mobile cloud computing, multi-access edge computing includes a base station cooperative cache layer and user cooperative cache layer. Selecting the most appropriate cache content according to actual needs and determining the most appropriate location to optimize the cache performance have emerged as serious issues in multi-access edge computing that must be solved urgently. For this reason, a cache placement algorithm based on comprehensive utility in big data multi-access edge computing (CPBCU) is proposed in this work. Firstly, the cache value generated by cache placement is calculated using the cache capacity, data popularity, and node replacement rate. Secondly, the cache placement problem is then modeled according to the cache value, data object acquisition, and replacement cost. The cache placement model is then transformed into a combinatorial optimization problem and the cache objects are placed on the appropriate data nodes using tabu search algorithm. Finally, to verify the feasibility and effectiveness of the algorithm, a multi-access edge computing experimental environment is built. Experimental results show that CPBCU provides a significant improvement in cache service rate, data response time, and replacement number compared with other cache placement algorithms.

Keywords: Multi-access edge computing, comprehensive utility, big data, replacement cost, cache placement

The authors thank the editor and the anonymous reviewers for their helpful comments and suggestions. The work was supported by the National Natural Science Foundation (NSF) under grants (No. 61802353), Henan Provincial Department of Science and Technology (NO. 192102210270, NO.212102210407), and Dr Fund of Zhengzhou University of Light Industry.

http://doi.org/10.3837/tiis.2021.11.002 **ISSN : 1976-7277**

1. Introduction

The large-scale popularization of smart phones and mobile devices in recent years, combined with the explosive growth of rich media applications, is generating massive data in the communication system. According to a prediction report by Cisco VNI in 2019, global mobile data traffic will reach 930 EB by 2022, at which point mobile video traffic will account for 79% of the total mobile traffic [1]. The rapid growth of mobile traffic, especially the development of video based broadband and low delay services, has created significant challenges to mobile networks. The pressure of network bandwidth is increasing dramatically and heavy mobile application traffic places huge pressure on mobile backhaul core networks. Additionally, with the increase of network bandwidth, the probability of network congestion increases greatly and can lead to a rise in packet loss rate and end-to-end network delay, directly impairing user experience [2].

Deploying the content of computing processing capacity and user needs on edge devices such as base stations closer to users is an effective way to alleviate peak traffic congestion and achieve low latency [3]. Thus come into being the technology of multi-access edge computing. As edge devices such as base stations are very close to users in geographical location, they are highly suitable for providing edge services. Deploying content caching on edge devices such as base stations and providing edge services has also become an important research direction for both academia and industry [4].

An effective edge caching architecture provides numerous benefits, including [5-7]: (1) When a user requests, if the cache node near the user has cached the user's request content, the user can obtain the content directly from the cache node nearby, instead of establishing a connection with the core server at the far end. This reduces data transmission pressure of the core network; (2) Mobile users deriving the request content from the adjacent cache nodes can greatly reduce user delay; (3) The use of edge cache technology can lower the number of user requests to the remote core server and reduce user request processing pressure on the remote server. The cache architecture in big data multi-access edge computing environment is illustrated in **Fig. 1**.

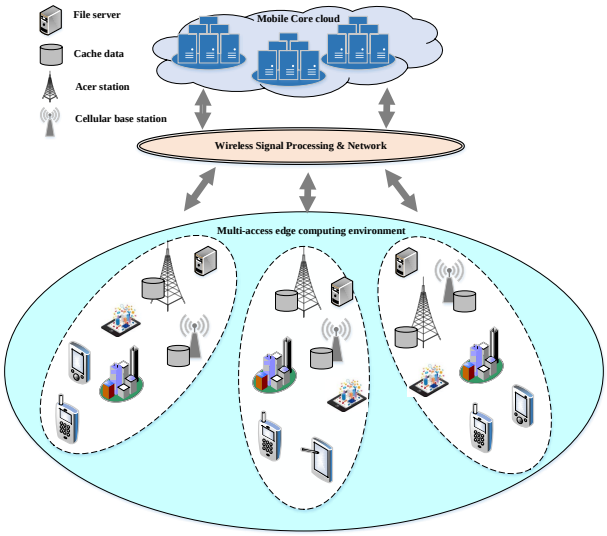

Fig. 1. Cache architecture in big data multi-access edge computing environment

In the actual deployment of cache content, the user's access behavior and the update of cache content are highly dynamic [8]. The main problems to address for cache placement decisions are which content should be selected for caching and which data nodes should cache content be placed on. Therefore, how to select the appropriate cache content according to the actual needs and find the most appropriate location to optimize the cache performance must still be determined.

This paper analyzes the characteristics of cached data objects and data nodes. By using the three factors of cache object acquisition, cache value, and replacement cost, a comprehensive utility model is obtained and the best data node is calculated by tabu search algorithm. The main contributions of this paper are as follows:

(1) The cache value generated by cache placement is calculated using cache capacity, data popularity, and node replacement rate. The cache placement problem is then modeled according to cache value, data object acquisition, and replacement cost;

(2) The cache placement model is transformed into a combinatorial optimization problem. Combined with the concept of the tabu search algorithm, a cache placement algorithm based on comprehensive utility in multi-access edge computing environment is proposed;

(3) A multi-access edge computing environment is built to verify the feasibility and effectiveness of the proposed algorithm. Experimental results show that the cache placement algorithm based on comprehensive utility in multi-access edge computing environments (CPBCU) provides a significant improvement in cache service rate, data response time, and the number of replacement data compared with other cache placement algorithms.

The rest of the paper is organized as follows: Section 2 reviews related works. Section 3 describes the cache placement model in edge computing environment. Section 4 presents the cache placement algorithm based on comprehensive utility in multi-access edge computing environment. Section 5 describes the details of our proposed algorithms. Section 6 provides comparison and analysis of experiment results, followed by the conclusion in Section 7.

2. Related work

Cache placement strategies in multi-access edge computing have become a popular research topic in recent years [9~10]. This section introduces its development status in China and abroad, and points out some problems identified in the research.

2.1 Cache placement for improving hit rate

Many scholars have studied methods to improve the hit rate through file cache with limited cache capacity. Pantisano F et al. [11] proposed a novel cache-aware user association algorithm, which calculates the request distribution of each base station file, caches the file according to the popularity of the file, and maximizes the hit probability by using the optimal user access based on the matching algorithm. Zheng C et al. [12] studied network edge caching using big data and machine learning methods to estimate content popularity and design active caching strategy. In this way, the performance of the network can be improved and the growing demand for wireless resources can be alleviated. Müller S et al. [13] proposed a novel algorithm for context-aware proactive caching, updating the cache content by learning the context information of connected users regularly so as to improve the hit rate. Wang X et al. [14] proposed an integration of the deep reinforcement learning technology and federated learning framework with mobile edge systems to optimize mobile edge computing, caching, and communication. Lei L et al. [15] presented a viable alternative to the conventional methods for caching optimization, employing an algorithm which uses deep neural network to

analyze the optimization algorithm of cache content distribution, facilitating a mobile edge network achieve cache content distribution with the lowest energy consumption. Tao M et al. [16] proposed a file caching strategy to improve the hit rate and reduce the pressure of backhaul bandwidth. In this work, it was determined that in cooperative transmission, multiple base stations can provide access services for the same user, but the base stations must obtain the same file from the core network at the same time, increasing the backhaul pressure. Qu J et al. [17] proposed a greedy algorithm for cache content placement which transforms the problem of cache placement into the problem of maximizing the single tone submodule function.

2.2 Cache placement to reduce latency

With the increase of cache file hit rate, users are more likely to obtain files directly from the nearby base station and other edge devices, so file transmission delay will be greatly reduced. Numerous studies have explored the effect of file caching on reducing file transmission delay. Wang Y et al. [18] proposed a distributed algorithm with polynomial complexity. This algorithm reduces the delay of file transmission by file caching and transforms the optimization problem into a facility location problem. Spivak A et al. [19] proposed an approach for the improvement of data placement. The algorithm considers the memory capacity, CPU number, and other attributes, and uses Hadoop Distributed File Systems (HDFS) cache to improve the task performance. Xie R et al. [20] studied the cooperation between core network cache and base station cache in 5G, proposing a heterogeneous cooperation cache strategy for energy universities to achieve energy efficiency optimization of network system. Liao J et al. [21] studied the optimization of content cache placement in which file and cache sizes are different and multicast transmission is used to minimize the average return rate. Ren D et al. [22] proposed a group based cache strategy which considers the allocation of storage resources to reduce the average delay and total energy consumption of the content. Wei J et al. [23] studied the cooperation scheme between multi access edge computing (MEC) servers to optimize the performance of content caching and delivery between MEC and mobile devices. In this work, the cooperative cache problem is formalized as an integer linear programming problem and solved by subgradient optimization algorithm. Yu R et al. [24] explored the application of scalable video coding technology in collaborative video caching and inter cell scheduling to further improve the cache capacity of system collation.

In addition, the current cache placement methods contain the following problems: (1) Few studies comprehensively consider the cache capacity of nodes and the number of node replacements; (2) The cache price value of the combination of data popularity, node replacement number, and node cache capacity is rarely considered; In response to these limitations, a cache placement algorithm based on comprehensive utility in multi-access edge computing environments (CPBCU) is proposed.

3. Cache Placement Model for Multi-access Edge Computing

3.1 Cache placement for improving hit rate

In the multi-access edge computing environment, the architecture of cache placement based on comprehensive utility is shown in **Fig. 2**. The system mainly includes three parts: acquisition of cache objects, cache value, and replacement cost of cache data. The acquisition of cache object is the transmission of cache data from the data node storing the data to the data node to be cached; the cache value comprehensively considers the data popularity, replacement rate,

and cache capacity of the data node, so that the value of data caching to the data node is the largest; the replacement cost refers to the possible replacement cost when the data is transferred to the corresponding data node. This method can improve the utilization of cache data and reduce the cost of cache replacement and system transmission. The main notations are summarized in **Table 1**.

Table 1. Summary of main notations.

Notations	Definition
$B(n)$	Data block set
$D(m)$	Data node set
b_i	the i data block
d_i	the j data node
r_i^{cac}	The available cache capacity of data node i
r_i^{mem}	The cache space size of data node i
r_i^{cpu}	the CPU speed of data node i
r_i^{disk}	the memory read and write speed of data node i
Cap	Cache capacity of data node
$R^{(1)}$	The cache availability of data node
$R^{(2)}$	The CPU speed
$R^{(3)}$	The memory read-write speed of the data node
Rep_i	Replacement rate of data object i
Pop_i	Popularity of data object i
$h(d_i,d_k)$	The network distance between data nodes d_i and d_k
$Value_i^j$	The cache value of data i buffered on the j data node
$Penalty_i^j$	The cache replacement cost generated by caching data i to data node j

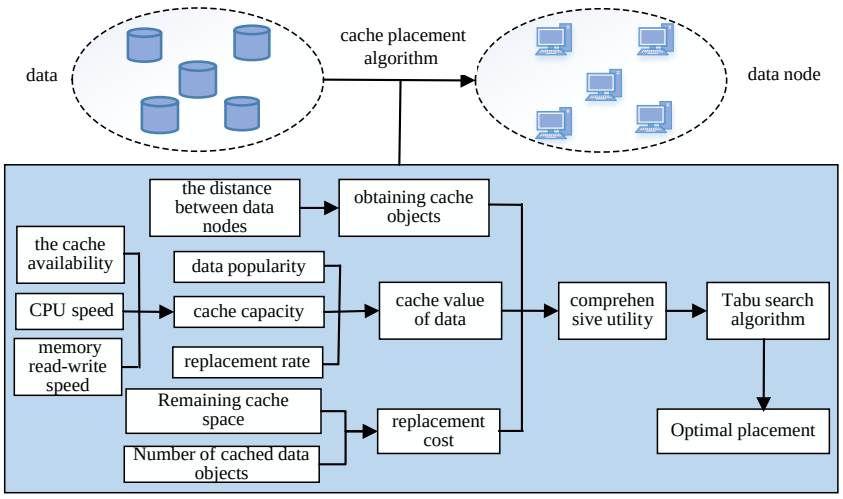

Fig. 2. Cache placement architecture based on comprehensive utility in big data multi-access edge computing

3.2 Factors affecting cache placement

In this paper, CPBCU algorithm is employed to consider the popularity of cache objects, cache capacity of data nodes in edge devices, replacement rate of data nodes, network distance of data nodes, and replacement cost, integrating these factors for unified quantification.

(1) Cache capacity of data nodes

Suppose the data block set to be cached is represented as $B(n)=\{b_i \mid i=1,2,...,n\}$ by n data blocks, where b_i represents the i data block. It is assumed in this work that when the edge server cluster caches data, the data block size is the same and the data block size is set to mc. The cluster data node set is composed of m data nodes, which is expressed as $D(m)=\{d_j \mid j=1,2,...,m\}$, Where d_j is the j data node and each data node has limited cache space.

In multi-access edge computing, the purpose of caching data to data nodes is to improve the efficiency of task execution and speed up service requests. Node cache capacity Cap is determined by cache availability $R^{(1)}$, CPU speed $R^{(2)}$ and memory read and write speed $R^{(3)}$. As the three indicators use different calculation units, they must be homogenized. In this paper, range method is used to measure all indicators.

The cache availability of data nodes is quantified by the following methods:

1) In an edge server cluster, the available cache capacity of each data node is r_i^{cac}, and cache space size is r_i^{mem}, where $i=1,2,...,n$.

2) The cache availability of each data node is calculated as: $r_i^{use}={r_i^{cac}}/{r_i^{mem}}$, the arithmetic mean of the available cache rates of the cluster data nodes is: $avg_{use}=\sum_{i=1}^{n} r_i^{use} \Big/ n$.

3) The normalization formula of the available cache rate of each data node on the cluster is as follows:

$$R_i^{use}=\frac{r_i^{use}}{avg_{use}} \tag{1}$$

4) Format the available cache rate of data nodes according to range method. The calculation formula is:

$$R_i^{(1)}=\frac{R_i^{use}-\min\limits_{i\in n} R_i^{use}+1}{\max\limits_{i\in n} R_i^{use}-\min\limits_{i\in n} R_i^{use}+1} \tag{2}$$

In a similar way, CPU speed and The memory read-write speed can be obtained.

$$R_i^{(2)}=\frac{R_i^{cpu}-\min\limits_{i\in n} R_i^{cpu}+1}{\max\limits_{i\in n} R_i^{cpu}-\min\limits_{i\in n} R_i^{cpu}+1} \tag{3}$$

$$R_i^{(3)}=\frac{R_i^{disk}-\min\limits_{i\in n} R_i^{disk}+1}{\max\limits_{i\in n} R_i^{disk}-\min\limits_{i\in n} R_i^{disk}+1} \tag{4}$$

Therefore, the cache capacity of each data node can be expressed as:

$$Cap_i = \sqrt[3]{R_i^{(1)} * R_i^{(2)} * R_i^{(3)}} \tag{5}$$

(2) Replacement rate of data nodes

This paper introduces the cache replacement rate of data nodes, which can accurately express the cache state and demand degree of data nodes and explain the timeliness of cache data. The cache replacement rate Rep of a node represents the data size of the cache replacement of a data node in the unit storage resource. The calculation formula is:

$$Rep_i = \sum_{j=1}^{k} data_j^{\,i} \Big/ r_i^{mem} \tag{6}$$

Where k represents the number of cache replacement of data node i, $data_j^{\,i}$ is the data size of data node i in the j replacement, and r_i^{mem} is the size of the cache space of data node i. The larger the value of data node replacement rate is, the higher the missing rate of cached data requests of the data node.

(3) Data popularity

The factors that affect the popularity of data mainly include the frequency of data access, the average access time interval, and the recent nature of the access. The calculation formula of data popularity Pop can be expressed as follows:

$$Pop_i = \frac{A_i}{\sum_{j=1}^{n} A_j} * \frac{1}{T^{now} - T_i^{last}} \Big/ \frac{T_i^{last} - T_i^{first}}{A_i} \tag{7}$$

where A_i is the number of times the data object i has been accessed, T_i^{last} is the last time that data object i was accessed, T_i^{first} is the first time that data object i was accessed, T^{now} is the current time.

3.3 Cache placement model

The cache placement model is determined by three components: data object acquisition, cache value, and replacement cost.

(1) Obtaining cache objects

Cache data must be acquired before it is placed in the cache. In this paper, the distance between data nodes is used to represent cache data acquisition. The calculation formula of data node j acquiring data block i is as follows:

$$Acq_i^{\,j} = h(d_i, d_k) \tag{8}$$

where $h(d_i, d_k)$ is the network distance between data nodes d_i and d_k, that is, the data transmission overhead between two data nodes. In the multi-access edge computing environment, the data node of cache data and the data node of storage copy cannot be the same node, so $h(d_i, d_k) > 0$.

(2) Cache value of data

The calculation formula of cache value of data i buffered on the j data node is as follows:

$$Value_i^{\,j} = \frac{Pop_i}{Rep_j / Cap_j} = \frac{Pop_i * Cap_j}{Rep_j} \tag{9}$$

For increased convenience, variable $x_{i,j}$ is defined, which means that data block i is placed on data node j, and the calculation formula is as follows:

$$x_{i,j}=\begin{cases}1, \ data\ block\ i\ cached\ on\ data\ node\ j\\ 0, \qquad\qquad otherwise\end{cases} \tag{10}$$

Therefore, the formula for calculating the total cache value of all data to be cached on the data node can be expressed as follows:

$$Value=\sum_{i=1}^{n}\sum_{j=1}^{m}Value_i^{\ j}=\sum_{i=1}^{n}\sum_{j=1}^{m}\frac{Pop_i*Cap_j*x_{i,j}}{Rep_j} \tag{11}$$

(3) Replacement cost

In the multi-access edge computing environment, when the cache is placed, it is assumed that the data i is cached on the data node j. If the available cache capacity of the data node can accommodate the data, the resulting replacement cost is 0. If the available cache capacity of the data node cannot accommodate the data, the resulting replacement cost is $mc/band_j$, where $band_j$ is the network bandwidth of data nodes and mc is the size of data. Therefore, the calculation formula of cache replacement cost generated by caching data i to data node j is as follows:

$$Penalty_i^{\ j}=\begin{cases}0, \ mc\le r_j^{\ cac}\\ val=\dfrac{mc}{band_j}, \ otherwise\end{cases} \tag{12}$$

During cache placement, the replacement cost of data node j is calculated as follows:

$$penalty_j=\begin{cases}0, \ k=\sum_{i=1}^{n}x_{i,j}-\dfrac{r_j^{\ cac}}{mc}\le 0\\ \sum_{p=1}^{k}\dfrac{mc}{band_j}, \ k=\sum_{i=1}^{n}x_{i,j}-\dfrac{r_j^{\ cac}}{mc}>0\end{cases} \tag{13}$$

(4) Comprehensive utility

The calculation formula of the overall mathematical model of cache placement based on comprehensive utility is as follows:

$$\sum_{i=1}^{n}\sum_{j=1}^{m}x_{i,j}*(Value_i^{\ j}-Acq_i^{\ j}-Penalty_i^{j})=\sum_{i=1}^{n}\sum_{j=1}^{m}x_{i,j}*\left(\frac{Pop_i*R_j}{Rep_j}-Acq_i^{\ j}-Penalty_i^{j}\right) \tag{14}$$

$$s.t.\begin{cases}x_{i,j}\in\{0,1\},\forall i\in[1,n],\ j\in[1,m]\\ \sum_{j=1}^{m}x_{i,j}=n,\forall i\in[1,m]\\ 1\le\left|\left\{\sum j\,|\,x_{i,j}=1,\forall i\in[1,\mathrm{n}]\right\}\right|\le m\end{cases}$$

Therefore, in the multi-access edge computing environment, the objective function of cache placement problem is calculated as follows:

$$\max[\sum_{i=1}^{n}\sum_{j=1}^{m} x_{i,j} * (\frac{Pop_i * R_j}{Rep_j} - Acq_i^{\ j} - Penalty_i^j)] \tag{15}$$

$$s.t. \begin{cases} x_{i,j} \in \{0,1\}, \forall i \in [1,n],\ j \in [1,m] \\ \sum_{j=1}^{m} x_{i,j} = n, \forall i \in [1,m] \\ 1 \le \left| \{ \sum j \mid x_{i,j} = 1, \forall i \in [1,n] \} \right| \le m \end{cases}$$

4. Determine the Initial Solution of Cache Placement Based on the Placement Strategy of Replacement Rate

4.1 Equations

In the tabu search algorithm, the step of solving the optimal solution of the cache placement problem is a process of locating the optimal solution in the tabu search process. Based on the mathematical model of cache placement, the objective function of cache placement algorithm is defined as:

$$f(s) = E_1 - E_2 - E_3 \tag{16}$$

Where E_1 is the cache value of data placement, E_2 is the cost of data acquisition, and E_3 represents the replacement cost of data nodes. At the end of tabu search, the optimal solution of cache placement is obtained.

4.2 Initial solution of cache placement based on placement strategy of replacement rate

The initial solution of cache placement is obtained based on the priority placement algorithm of replacement rate. The basic steps are as follows:

(1) The data object set to be cached is $B(n) = \{b_i \mid i = 1,2,...,n\}$, and the data node set in the edge server cluster is $D(m) = \{d_j \mid j = 1,2,...,m\}$;

(2) Calculate the popularity of each data object as Pop_i, and sort it into a set $Popset$. The replacement rate of each data node is calculated as Rep_j and its composition set is $Repset$;

(3) Perform steps 4, 5 and 6 for each cache data object in sequence and place the corresponding cache data object on the corresponding data node;

(4) According to the popularity of each cached data object, the corresponding span of the data block is calculated as $span = (Pop_i - \min Pop) / (\max Pop_i - \min Pop_i)$;

(5) According to step 4, the span $span$ is obtained and the replacement rate of the data node placed by the data is calculated as $rep = span * (\max Rep_i - \min Rep_i) + \min Rep_i$;

(6) By comparing the rep with the $Repset$, the corresponding data nodes and the initial solution of cache placement are both determined.

5. Implementation of Cache Placement Algorithm for big data Multi-access Edge Computing

5.1 Algorithm description

Algorithm 1 is the pseudo-code description of cache placement algorithm based on comprehensive utility in big data multi-access edge computing environments.

Algorithm 1: cache placement algorithm based on comprehensive utility

Input: $B(n)$ is the data set to be cached, $D(m)$ is the collection of data nodes in Hadoop cluster

Output: Cache data placement result $Result$

for $i=1$ to n do
  calculate the popularity Pop_i of each data block i
  save Pop_i to set $Popset$
end for
for $j=1$ to m do
  Calculate the cache capacity Cap_j of each data node j
  Calculate the replacement rate Rep_j of each data node J
  save Cap_j and Rep_j to sets $Capset$ and $Repset$, respectively
end for
for $i=1$ to n do
  Calculate the corresponding span of the data block i
  Calculate the replacement rate of caching data block i to data node j
  $Rep = span * (\max Rep_i - \min Rep_i) + \min Rep_i$
  for $j=1$ to m do
    if $Rep_j \in [\lfloor Rep \rfloor, \lceil Rep \rceil]$
      Save data node j to set $Result$
    end if
  end for
end for
Calculate the objective function S_{init} according to $Result$
set $S_{current} = S_{init}$, $S_{best} = S_{init}$, and take the initial solution as the optimal solution
while $count < max$ do
  using $S_{current}$ to form neighborhood Table $List$
  for $k=1$ to $List$ do
    Getting local optimal solution S_{can} by objective function
  end for
  if S_{can} is better than S_{best} in the taboo table
    $S_{best} = S_{can}$

28: end if
29: if checkInList(S_{can})=True
30: $S_{current} = S_{best}$
31: else $S_{current} = S_{can}$
32: addList(S_{can})
33: end if
34: $count = count + 1$
35: end while
36: Get the optimal solution $Result$ of cache data placement

5.2 Algorithmic complexity analysis

In this paper, the total number of data placed in the cache is n, the number of data nodes is m. The time complexity of cache placement optimization algorithm mainly includes two parts:(1) Initial solution of cache placement: traverse each cache data and obtain an array of placement results of data and data nodes using a placement algorithm based on displacement rate. Therefore, the time complexity of initial solution is $o(n*m)$; (2) Solution optimization of cache placement: the initial size is n tabu array. Perform a tabu search based on the principle of tabu search algorithm. Therefore, the time complexity of solution optimization is $o(n^2)$. Generally, $n > m$, so the time complexity of cache placement is $o(n^2)$.

6. Experimental Verification and Comparison

6.1 Experimental environment and configuration

(1) Experimental environment
The multi-access edge computing environment was composed of edge servers and core clouds. The edge servers included nine local hosts and the core cloud was hosted on Alibaba Cloud. The configuration of the edge server cluster node is shown in **Table 2** and the configuration of the core cloud is shown in **Table 3**.

Table 2. Multi-access edge cloud node configuration

Host name	Configuration	IP	node function
Master	CPU:8-core(i7-9700) RAM:16GB DISK:2TB	192.168.201.20 192.168.1.2(VPN)	edge orchestrator
Slave1-Slave3	CPU:8-core(i7-9700) RAM:8GB DISK:512GB	192.168.201.21 192.168.201.22 192.168.201.23	edge servers
Slave4-Slave6	CPU:4-core(i5-9400F) RAM:8GB DISK:1TB	192.168.201.24 192.168.201.25 192.168.201.26	edge servers
Slave7-Slave9	CPU:4-core(i3-9100) RAM:8GB DISK:512GB	192.168.201.27 192.168.201.28 192.168.201.29	edge servers

Table 3. Central cloud instance configuration

Central cloud service provider	Configuration	IP	node function
Master	CPU:1-core Inter(R) Xeon(R) E5-2680 v3 2.50GHz; RAM:4GB Bandwidth:30Mbps	121.42.206.150 192.168.1.10（VPN）	Central cloud server
Slave1	CPU:1-core Inter(R) Xeon(R) E5-2680 v3 2.50GHz; RAM:2GB Bandwidth:20Mbps	121.42.206.151 192.168.1.11	Central cloud server
Slave2	CPU:1-core Inter(R) Xeon(R) E5-2680 v3 2.50GHz; RAM:1GB Bandwidth:10Mbps	121.42.206.152 192.168.1.12（VPN）	Central cloud server

(2) Evaluation index

1) Cache service rate: refers to the probability that the requested data is responded to by the data node cache. This indicator is used to reflect the advantages and disadvantages of the cache placement algorithm. The calculation formula is:

$$\gamma = \alpha / \beta \tag{17}$$

Where α is the number of accessed data stored in the cache and β is the total amount of data accessed.

2) Data response time: refers to the time required to access data, which is predominantly used in this paper to test the impact of the cache placement algorithm on data access.

3) Displacement number: refers to the ratio of the number of evicted operations in the data node's cache to the cache capacity. This indicator reflects the number of data cached in the data node's cache and the performance of the placement strategy. Its calculation formula is:

$$\lambda = \sum_{i=1}^{n} \frac{\delta_i}{r_i} \Big/ m \tag{18}$$

Where δ_i is the number of cache eviction operations of data node i, r_i is the cache capacity of data node i, and m is the total number of data nodes.

6.1 Experimental results and analysis

To verify the feasibility and effectiveness of the algorithm, the CPBCU algorithm proposed in this paper was compared with D2D-CCP algorithm [17] and the original centralized cache management (CCM) in HDFS. The experiment in this section adopts the control variable method, that is, only one variable is changed in each experiment, while the other variables are the same. Each experiment is repeated 10 times under the same conditions, and the average value is taken as the final experimental result.

(1) Influence of cache capacity on algorithm performance

This group of experiments mainly explored the effect of different cache capacity on the performance of the algorithm. The range of cache capacity of each data node was 20~50 data blocks. The experimental simulation placed 100 cache data blocks and randomly read 200 data blocks in HDFS. The experimental results are provided in **Fig. 3, 4, and 5**.

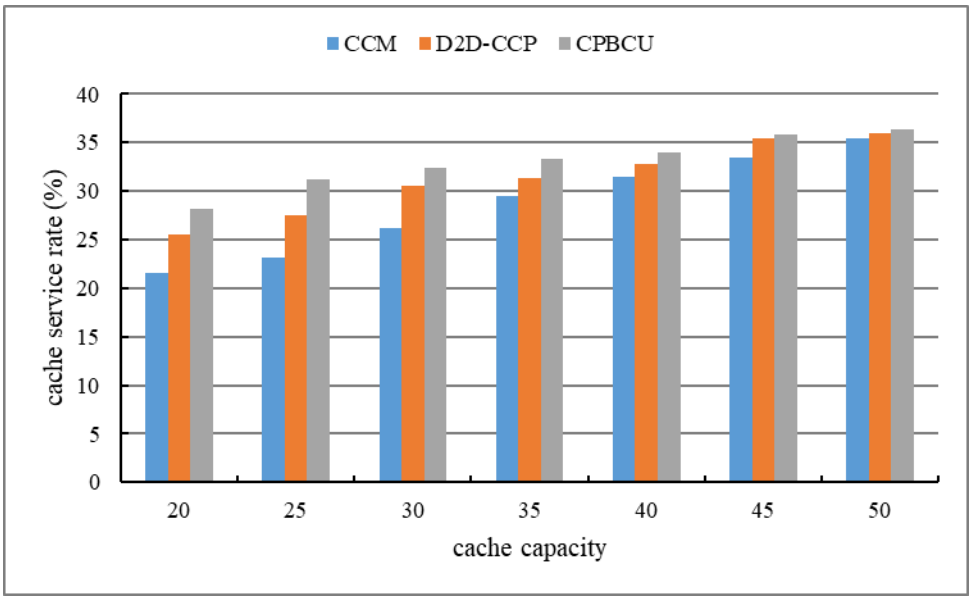

Fig. 3. Comparison of cache service rates in different cache capacities

Fig. 3 shows the change of cache service rate in different cache capacity. It can be seen that CCM algorithm has the lowest cache service rate, CPBCU algorithm has the highest cache service rate. When the cache capacity is increased from 20 to 80, the cache service rates of the three algorithms also increase. Among them, the cache service rate of D2D-CCP algorithm rises by about 5% and the cache service rate of the algorithm proposed in this paper is increased by about 10%. This is because with the increase of cache capacity, the cache of data nodes can hold more data objects and the cache hit rate will increase, so the cache service rate of data nodes will also rise.

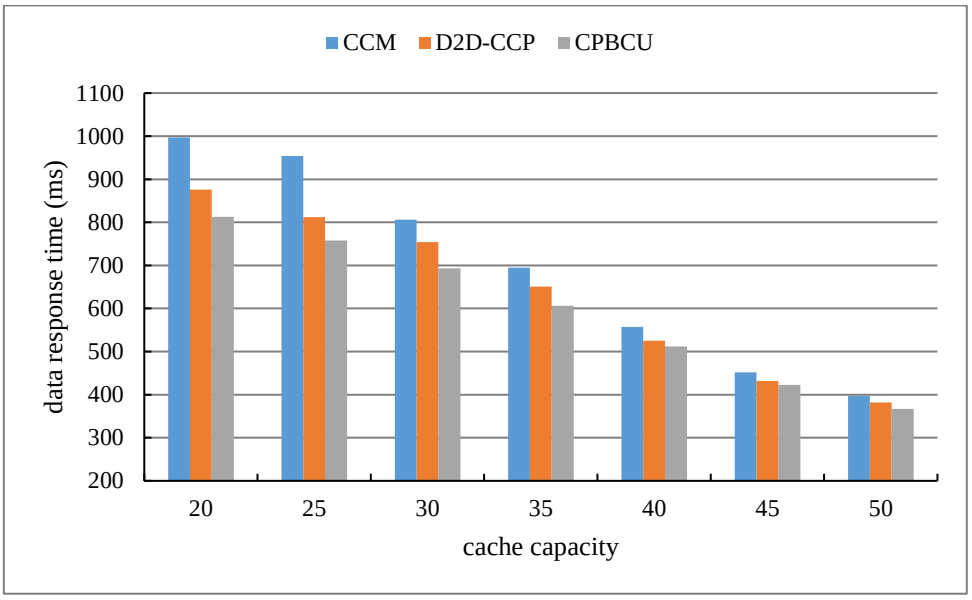

Fig. 4. Comparison of data response time in different cache capacities

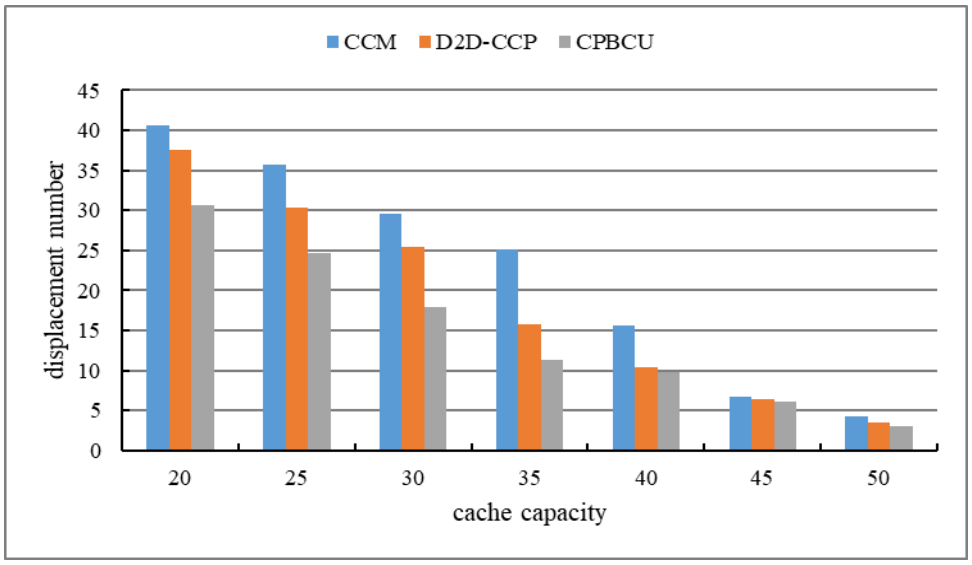

Fig. 5. Comparison of replacement number in different cache capacities

Fig. 4 shows the change of data response time in varying cache capacity. It can be seen from the figure that with the increase of cache capacity, the data response time of the three algorithms decrease and the response time of CPBCU algorithm and D2D-CCP algorithm is less than that of CCM algorithm. This is because with the increase of cache capacity, data nodes can hold more cache data and the speed of data acquisition in cache is much faster than that in disk, so the data response time decreases with the increase of cache capacity.

Fig. 5 shows the results of the replacement number with different cache capacity. It can be seen from the figure that with the increase of cache capacity, the number of replacements of the three algorithms decreases, with the smallest number of replacements from CPBCU algorithm, the highest number of replacements by CCM algorithm. This is because the CPBCU algorithm considers the popularity and the replacement rate of data nodes, making the data objects with high popularity difficult to replace.

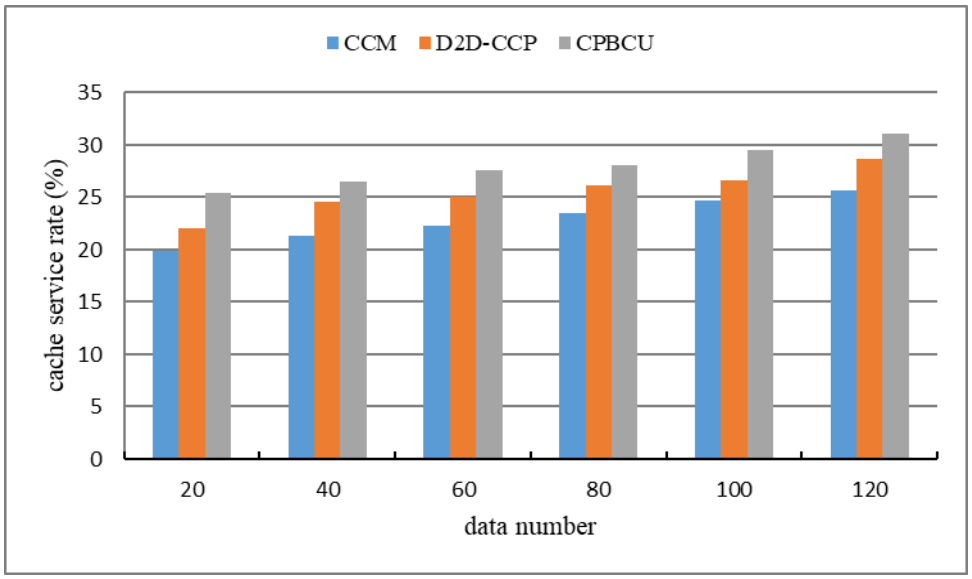

Fig. 6. Comparison of cache service rates in different data numbers

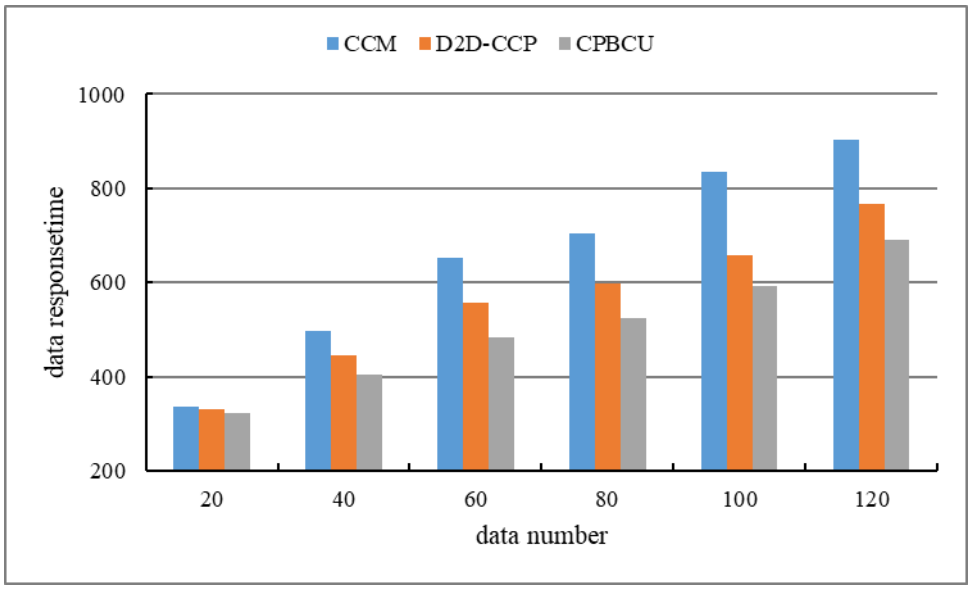

Fig. 7. Comparison of data response time in different data numbers

(2) Influence of data number on algorithm performance

This experiment focused on the effect of different data numbers on the performance of the algorithm. The number of data varied from 20 to 120, and the experimental results are shown in **Fig. 6, 7, and 8**.

Fig. 6 shows the change of cache service rate in different data numbers. It can be seen from the figure that the cache service rates of the three algorithms are augmented with an increasing number of data. In a certain number of data, CCM algorithm has the lowest cache service rate, while D2D-CCP algorithm and CPBCU algorithm have higher cache service rate. This is because in the process of cache placement, CPBCU algorithm takes into account the popularity of cache data objects and the replacement cost of data nodes. By caching the data with high popularity to the data nodes with low replacement number, it can ensure that the data with high flow is not easily replaced.

Fig. 7 shows the results of the change of data response time in different data numbers. It can be seen from the figure that as the number of data increases, the response time of the three algorithms rises. Among them, CCM algorithm has the longest response time, CPBCU algorithm has the lowest response time. This is because CPBCU algorithm places the data with high popularity on the data nodes with low replacement number, further increasing the hit rate of cache data, which makes the response time of CPBCU algorithm the shortest.

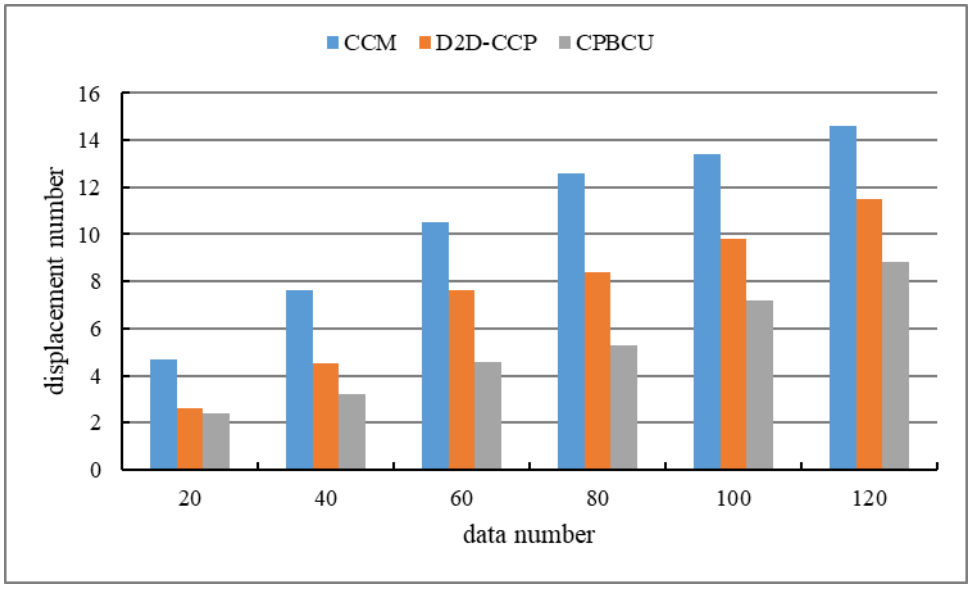

Fig. 8. Comparison of replacement numbers in different data numbers

Fig. 8 shows the change results of the replacement number in different data numbers. It can be seen from the figure that when the number of data is small, the replacement number of the three algorithms is relatively small, while when the number of data is large, the replacement number of the three algorithms is relatively large. This is because as the number of data increases, the number of data replaced in the data node cache will increase, so the number of replacements will continue to rise.

(3) Influence of data popularity on algorithm performance

This experiment mainly explored the influence of different data popularity on the performance of the algorithm. The popularity ranged from 0.3 to 0.9. In the experiment, 100 cache data blocks were placed and 200 data blocks in HDFS were randomly read. The experimental results are shown in **Fig. 9, 10, and 11**.

Fig. 9 shows the change of cache service rate in different data popularity. It can be seen from the figure that with the increasing popularity of data, the cache service rate of the three algorithms improve. Among them, CCM algorithm has the lowest cache service rate, while D2D-CCP algorithm and CPBCU algorithm have higher cache service rate. This is because with the increasing popularity of data, the number of data objects that can hit the cache grows and the cache service rate also increases. In addition, CPBCU algorithm places the data with high popularity on the data nodes with low replacement number, which further increases the hit rate of cache data and makes the cache service rate of CPBCU algorithm the highest.

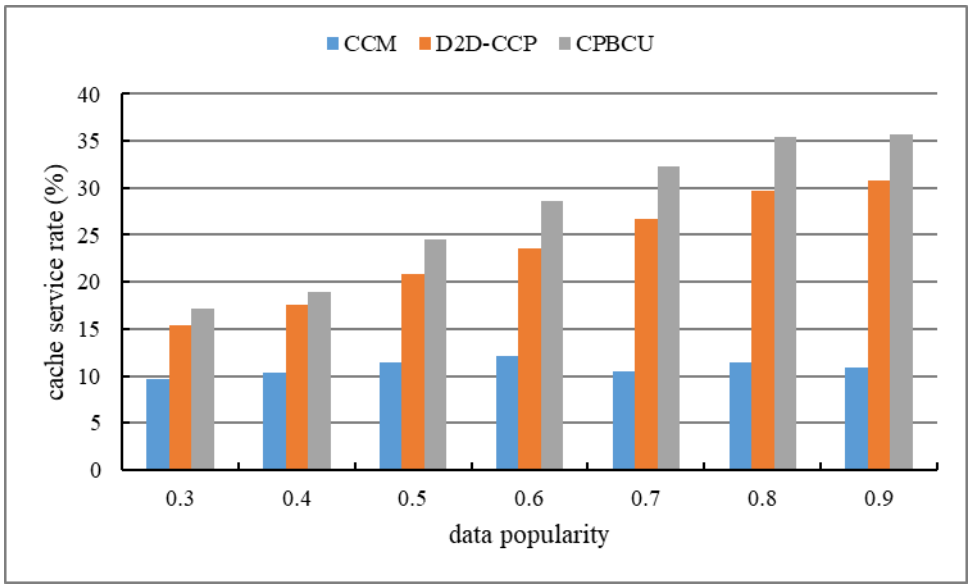

Fig. 9. Comparison of cache service rate in different data popularity

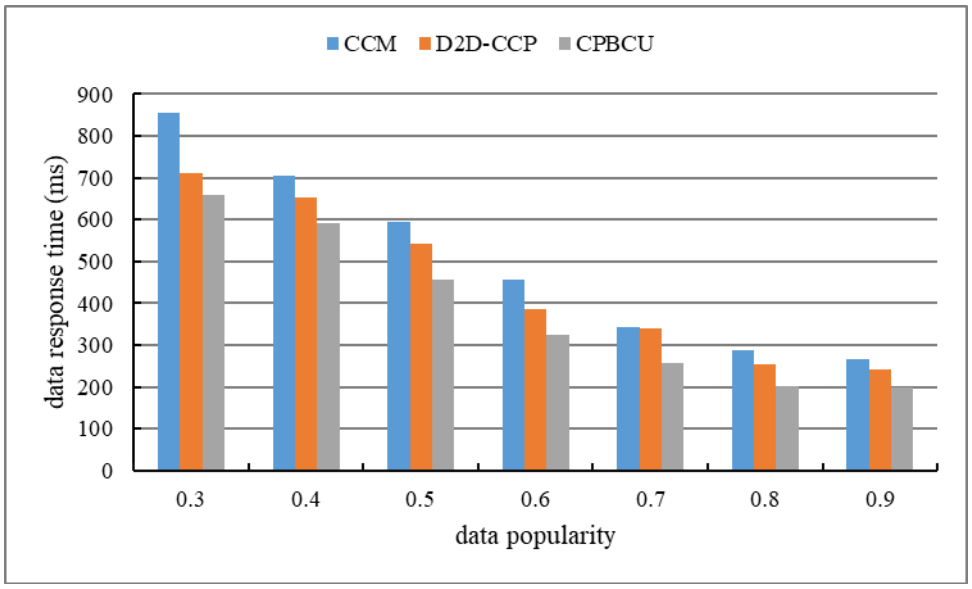

Fig. 10. Comparison of response time in different data popularity

Fig. 10 shows the results of data response time changes in different data popularity conditions. It can be seen from the figure that CCM algorithm has the longest response time, CPBCU algorithm has the shortest response time. This is because CPBCU algorithm considers data popularity and replacement cost, which ensures that data with high popularity is not easily replaced.

Fig. 11 shows the change of the replacement number in different data popularity. It can be seen from the figure that with the increase of data popularity, the replacement number of the three algorithms declines, with the replacement number of CPBCU algorithm declining the fastest. This is because D2D-CCP algorithm and CPBCU algorithm take into account the popularity of data objects. In addition, CPBCU algorithm also considers caching the data with

high popularity to the data nodes with low replacement number, which can ensure that the data is not easily replaced, thus reducing the replacement number.

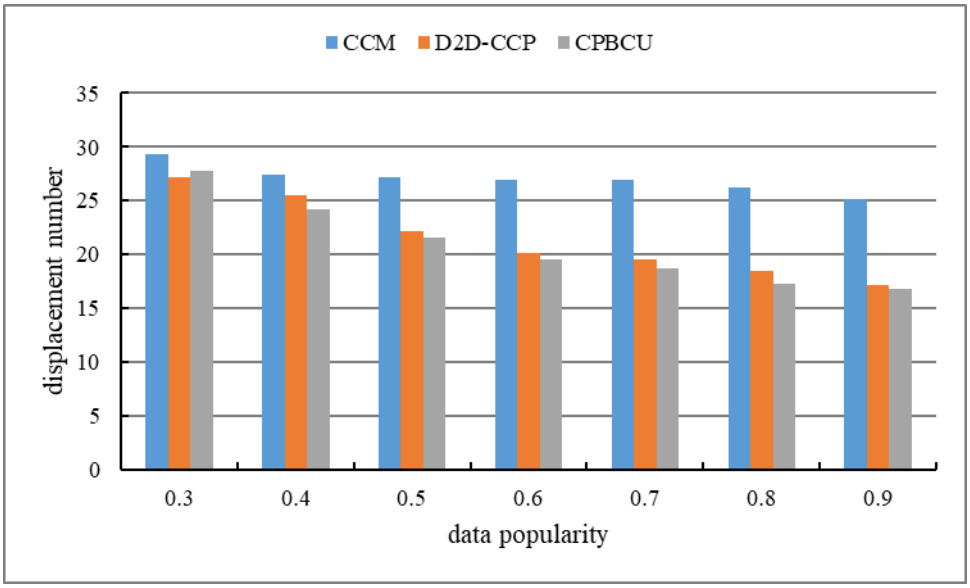

Fig. 11. Comparison of replacement number of in different data popularity

6.3 Experiment summary

Through the analysis of the above three groups of experiments, the following conclusions can be drawn: (1) The size of the cache capacity, the number of data, and the popularity of data all have a certain impact on the cache placement algorithm. The larger the cache capacity, the lower the number of data, the higher the popularity of data, and the longer the cache placement time, otherwise, the cache content should be updated frequently; (2) The CPBCU algorithm proposed in this paper considers the popularity of data and the characteristics of data nodes, obtaining superior response time and cache service rate compared to other similar algorithms.

7. Conclusion and future work

To fully explore the potential capabilities of the multi-access edge cache, this paper employed node cache capacity, data popularity, and node replacement rate to calculate the cache value generated by cache placement. The cache placement problem was modeled by considering three aspects: cache value, data object acquisition, and replacement cost. On this basis, combined with the tabu search algorithm, a cache placement algorithm based on comprehensive utility in big data multi-access edge computing environment was proposed. This algorithm can reduce network transmission overhead and replacement costs as well as improve the cache utilization. In the future, diversified scenarios bring diversified challenges to the network, and the caching technology also needs to be continuously optimized with the changes of services. For example, how to deal with the changes of wireless network state caused by mobile users switching between base stations, and how to ensure the service experience and maximize the caching benefit in the process of users moving are worthy of research.

Acknowledgement

The authors thank the editor and the anonymous reviewers for their helpful comments and suggestions. The work was supported by the National Natural Science Foundation (NSF) under grants (No. 61802353), Natural Science Foundation of Henan Province (No.202300410505), Henan Provincial Department of Science and Technology (No. 192102210270).

References

[1] CHIEN W C, WENG H Y, LAI CF, "Q-learning based collaborative cache allocation in mobile edge computing," *Future generation computer systems*, vol.102, no.1, pp.603-610, 2020. Article (CrossRef Link)

[2] Costa F R, da Rosa Righi R, da Costa C A, et al, "Nuoxus: A proactive caching model to manage multimedia content distribution on fog radio access networks," *Future Generation Computer Systems*, vol.93, pp.143-155, 2019. Article (CrossRef Link)

[3] Zhang T, Fang X, Liu Y, et al, "D2D-Enabled Mobile User Edge Caching: A Multi-Winner Auction Approach," *IEEE Transactions on Vehicular Technology*, vol.68, no.12, pp.12314-12328, 2019. Article (CrossRef Link)

[4] Assuncao M D, Silva Veith A, Buyya R, "Distributed data stream processing and edge computing: A survey on resource elasticity and future directions," *Journal of Network and Computer Applications*, vol.103, pp.1-17, 2018. Article (CrossRef Link)

[5] Biswas S, Zhang T, Singh K, et al, "An analysis on caching placement for millimeter–micro-wave hybrid networks," *IEEE Transactions on Communications*, vol.67, no.2, pp.1645-1662, 2018. Article (CrossRef Link)

[6] Xu X, Li Y, Huang T, et al, "An energy-aware computation offloading method for smart edge computing in wireless metropolitan area networks," *Journal of Network and Computer Applications*, vol.133, pp.75-85, 2019. Article (CrossRef Link)

[7] Sinky H, Khalfi B, Hamdaoui B, et al, "Adaptive edge-centric cloud content placement for responsive smart cities," *IEEE Network*, vol.33, no.3, pp.177-183, 2019. Article (CrossRef Link)

[8] Hsieh H C, Chen J L, Benslimane A, "5G virtualized multi-access edge computing platform for IoT applications," *Journal of Network and Computer Applications*, vol.115, pp.94-102, 2018. Article (CrossRef Link)

[9] Zhang Y, Li C, Luan T H, "A mobility-aware vehicular caching scheme in content centric networks: Model and optimization," *IEEE Transactions on Vehicular Technology*, vol.68, no.4, pp.3100-3112, 2019. Article (CrossRef Link)

[10] Xu D, Samanta A, Li Y, et al, "Network Coding for Data Delivery in Caching at Edge: Concept, Model, and Algorithms," *IEEE Transactions on Vehicular Technology*, vol.68, no.10, pp.10066-10080, 2019. Article (CrossRef Link)

[11] Pantisano F, Bennis M, Saad W, "Match to cache: Joint user association and backhaul allocation in cache-aware small cell networks," in *Proc. of International Conference on Communication*s, pp. 3082-3087, 2015. Article (CrossRef Link)

[12] Zheng C, Lei L, Zhenyu Z, et al, "Learn to Cache: Machine Learning for Network Edge Caching in the Big Data Era," *IEEE Wireless Communications*, vol.25, no.3, pp.28-35, 2018. Article (CrossRef Link)

[13] Müller S, Atan O, van der Schaar M, et al, "Context-aware proactive content caching with service differentiation in wireless networks," *IEEE Transactions on Wireless Communications*, vol.16, no.2, pp.1024-1036, 2016. Article (CrossRef Link)

[14] Wang X, Han Y, Wang C, et al, "In-edge ai: Intelligentizing mobile edge computing, caching and communication by federated learning," *IEEE Network*, vol.33, no.5, pp.156-165, 2019.

[15] Lei L, You L, Dai G, et al, "A deep learning approach for optimizing content delivering in cache-enabled HetNet," in *Proc. of International symposium on wireless communication systems*, pp.449-453, 2017. Article (CrossRef Link)
[16] Tao M, Chen E , Zhou H, "Content-Centric Sparse Multicast Beamforming for Cache-Enabled Cloud RAN," *IEEE Transactions on Wireless Communications*, vol.15, no.9, pp.6118-6131, 2015. Article (CrossRef Link)
[17] Qu J, Wu D, Long Y, "D2D Based Caching Content Placement in Wireless Cache-Enabled Networks," *Journal of Internet Technology*, vol.20, no.2, pp.333-344, 2019. Article (CrossRef Link)
[18] Wang Y, Tao X, Zhang X, "Joint Caching Placement and User Association for Minimizing User Download Delay," *IEEE Access*, no.4, pp.8625-8633, 2016. Article (CrossRef Link)
[19] Spivak A, Nasonov D, "Data Preloading and Data Placement for MapReduce Performance Improving," *Procedia Computer Science*, vol.101, pp.379-387, 2016. Article (CrossRef Link)
[20] Xie R, Tang Q, Huang T, "Energy-efficient hierarchical cooperative caching optimisation for 5G networks," *IET Communications*, vol.13, no.6, pp.687-695, 2019. Article (CrossRef Link)
[21] Liao J, Wong K K, Khandaker M R A, "Optimizing cache placement for heterogeneous small cell networks," *IEEE Communications Letters*, vol.21, no.1, pp.120-123, 2016. Article (CrossRef Link)
[22] Ren D, Gui X, Lu W, "GHCC: Grouping-based and hierarchical collaborative caching for mobile edge computing," in *Proc. of International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks*, pp.1-6, 2018. Article (CrossRef Link)
[23] Wei J, Gang F, Shuang Q, "Optimal Cooperative Content Caching and Delivery Policy for Heterogeneous Cellular Networks," vol.16, no.5, pp.1382-1393, 2017. Article (CrossRef Link)
[24] Yu R, Qin S, Bennis M, "Enhancing software-defined RAN with collaborative caching and scalable video coding," in *Proc. of International Conference on Communications*, pp.1-6, 2016. Article (CrossRef Link)

Yanpei Liu, born in 1982, Ph. D, Master's tutor. Her research interests include high-performance computing, edge computing, big data processing.

Wei Huang, born in 1982, ph. D, associate professor. His research interests include image processing, remote sensing image processing, machine learning.

Li Han, born in 1978, associate professor. Her major research interests include software engineering and compilation technology, intelligent information processing, and Pattern Recognition.

Liping Wang, born in 1981, Ph. D. Her research interests are the application of Internet of Things technology.