RESEARCH ARTICLE OPEN ACCESS

Multi-cell Collaborative Caching Based on Game Theory in Mobile Edge Computing

Niangtao Zhuang*, Jipeng Zhou**
*(Department of Computer Science, Jinan University, Guangzhou 510632,China
Email: niangtao4826@foxmail.com)
**(Department of Computer Science, Jinan University, Guangzhou 510632,China
Email: jpzhoucn@sohu.com)

--************************----------------------------------

Abstract:

Mobile edge computing cache and computing services are pushed to the edge network closer to user, separating network data from the remote cloud network to provide better service to users. At present, most of the researches on edge computing barely consider cooperation between the cells. In order to study the benefits brought by the cooperation between multiple cells, this paper proposes a multi-cell cellular network model developed as a Stackelberg game problem for the multi-cell cache optimization problem. The cell control center and the base station groups are regarded as the dominant and follower of the game model, respectively, and their respective revenue functions are formulated. Because their income function is non-continuous function, the traditional game theory solution can't be used to obtain the Nash equilibrium solution. This paper proposes an iterative alternating algorithm to solve the problem. The control center and the base station group respectively use the improved hybrid frog hopping algorithm (SMSA) and the greedy exchange algorithm (GSA) to solve the problem, and the two alternately iterate and finally obtain the approximate solution of the optimal solution of the model. Through numerical simulation experiments, we verify that the proposed algorithm outperforms the greedy algorithm proposed by other researches.

Keywords — **Mobile edge computing, Multi-cell cache optimization problem, Stackelberg game, iterative alternatin algorithm.**

--************************----------------------------------

I. INTRODUCTION

In recent years, with the rapid development of smart terminal devices and the emergence of many colorful multimedia content and applications, it is expected that the future traffic will expand tremendously and bring substantial pressure on mobile backhaul capacity. According to a Cisco report--"Cisco visual networking index: Global mobile data traffic forecast update, 2017-2022", global mobile data traffic will grow seven-fold between 2017 and 2022, and by 2022, monthly global mobile data The traffic will reach 77 exabytes, the annual traffic will reach nearly 1 terabyte, and nearly three-fifths of the traffic will be offloaded from the cellular network. At present, many scholars have proposed various solutions to meet these challenges [2][3], but in order to be able to meet the need of the large-scale traffic scenarios foreseen in future 5G networks, in addition to improving network capacity, it requires higher layers Innovations (eg, network architecture, backhaul and application). Mobile Edge Computing (MEC) is considered a viable solution deploying the server at the base station location closest to the user. In addition to providing users with high-quality computing and storage services with low latency and high bandwidth, it can also alleviate the link congestion problem of the core network with the advantages of low latency, high bandwidth, real-time radio network information and position sensing [4]. Reference [5]-[7] mainly discusses the concepts, application scenarios, features and challenges of MEC.

The report also shows that video traffic will account for 79% of global mobile data traffic by 2022, and most of the traffic load on the Internet comes from repeated viewing and downloading of the same set of video files, and edge caching can be seen as an effective solution to traffic repeat access problems. The mobile edge computing deployment server provides caching to the base station to support caching of popular video or files, which improves user quality of experience (QoE) and improves network performance (such as transmission delay, link load congestion, and energy effectiveness). Recently, there have been a lot of works on collaborative caching of edge networks. Reference [8] solves the utility maximization of MEC's collaborative service cache with a distributed algorithm based on Gibbs sampling, and develops a federation game to stimulate cooperation between base stations. Reference [9] studied the collaborative caching problem of fog-infinite access network (f-ran), formulating the network model into a weighted graph, whose weight was set as the unloading traffic increment of the corresponding complete subgraph, thus transforming the original clustering optimization problem into a 0-1 integer programming problem and solving it with the greedy algorithm. Reference [10] proposed a cooperative multicast aware cache (CMAC) method to solve the edge cache problem of MEC, and proved by experimental simulation that the average content access delay of CMAC was reduced by nearly 13% compared with the non-cooperative multicast aware cache (MAC).

Most of the existing researches on edge cache consider the network model of single cell, but the cooperation between cells are frequently ignored. In order to study the benefits of multi-cell collaboration, a multi-cell cellular network (MCN) model based on mobile edge computation is brought forth in this paper. In order to find a solution with both the delay and link load optimization conditions, this paper defines the multi-cell collaboration problem as the Stackelberg master-slave game model. The control center of each cell is regarded as the dominant player of the game, mainly focusing on the backhaul link load optimization of the model, and the base station group composed of all the base stations of the cell is regarded as the follower of the game party, mainly responsible for the delay optimization of the model. We formulate the income function of different players separately. Since the income function is a nonlinear function, we can't solve it with the traditional game theory solution. This paper proposes Stackelberg Model Alternating Iterative (SMAI) algorithms composed of two kinds of sub-algorithms. The algorithm is used to reach an approximate solution to the problem, and numerical simulation experiments are used to evaluate the pros and cons of the algorithm. The experimental results show that the proposed SMAI algorithm has better performance than other greedy algorithms, and it proves that compared with the single-cell collaboration model, multi-cell collaboration brings lower delay cost and backhaul link load to the network.

The rest of this paper is as follows: the second part describes the system model and caching incentives; The third part describes the model characteristics of the system and gives the problem description formula. The fourth part transforms the cache problem into a Stackelberg game model and formulates the cost function of the game players. In the fifth part, the algorithm for solving Stackelberg model is introduced. The sixth part analyze the performance of the algorithm. The seventh part is the final summary of this paper.

II. SYSTEM MODEL

We consider a multi-cell cooperative cellular network with a cache of m cells and the objects included in the network are Macro Base Station (MBS) Small Base Station (SBS) and Mobile User (MU), the model is shown in Fig. 1.

Assuming each cell has a MBS, then cell sets denoted as $C=\{c_1,c_2,\cdots,c_m\}$ and MBS sets denoted as $M=\{mb_1,mb_2,\cdots,mb_m\}$ where c_i denotes cell i and mb_i denotes MBS of cell i. Each MBS is equipped with an edge cloud server that allows MBS to have computing, storage and control functions to store a limited amount of content to improve the utility of the entire network. S_{mbs}^{i} denote the cache capacity of MBS of cell i. In addition, MBS can also coordinate and manage the SBS group of the cell as the control center of the cell. The MBS between different cells is connected through a high-capacity link (such as optical fiber), and each MBS is also connected to the remote internet through a backhaul link. Therefore, it is different from the previous collaborative buffering between single cells [8]-[10], the model of this paper can also obtain the requested file through the cooperation between cells.

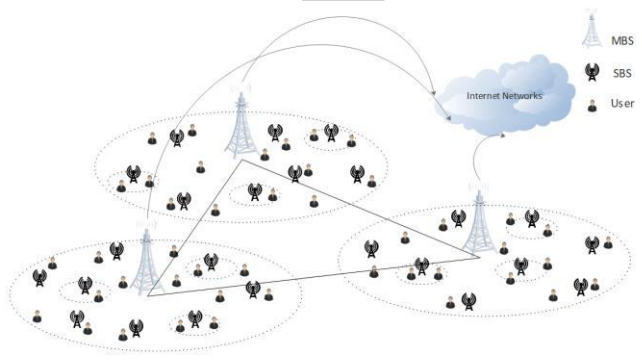


Fig. 1. Multi-cell heterogeneous network model

In addition to having one macro base station, each cell has s small base station (SBS) and h mobile user terminals. $B^i=\{mb_i,b_1^i,b_2^i,\cdots,b_s^i\}$ denotes the set of all base stations including MBS and SBS where mb_i denotes the MBS of cell i and b_j^i denotes the SBS j of cell i. $U=\{U^1,U^2,\cdots,U^m\}$ denotes the set of mobile user terminals of cell i where u_k^i denotes the user k of cell i. Furthermore, $B=\{B^1,B^2,\cdots,B^m\}$ and $U=\{U^1,U^2,\cdots,U^m\}$ denote the set of base stations and users in the global network respectively. Each user may be within the service scope of multiple SBSs, and the SBS can provide services to users within the service range and $S_{b_j}^i$ denotes the cache capacity of SBS j of cell i, but at the same time, the user can only connect with one SBS to experience services.

We model its active Topology by a undirected graph $G(V,E)$ which is shown in Fig. 2, where the vertices V denotes network nodes including MBS, SBS and user, and for every connected pair of nodes $i, j \in V$ there exist edges $e(i,j)\in E$ if they can establish a connection. To make a difference, V_{bs} denotes all the base stations including MBS and SBS in global network, and A_i, $i\in V_{bs}$ denote the set of files cached by node i.

Based on the model proposed in this paper, the rule that the user access request is served is summarized as: when the adjacent SBS of the user's location stores the file, it is directly sent to the user, and if the content is not in the adjacent SBS set, the file will be found in the cache location of the cell through the MBS and transmitted to the user. If the file does not exist in the cell, the file will be obtained through the cooperation of other cells and sent to the user. Otherwise, the request is sent to the remote cloud core network.

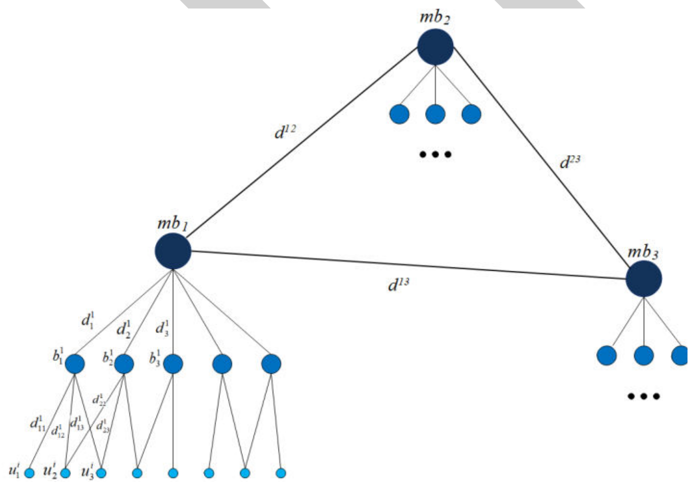


Fig. 2. Network model topology

III. Model Characteristics and Problem Formulation

A. Cache Content Characteristics

In order to solve the problem, we assume that the user's demand for the file will not change for a period of time, and the model of this paper does not consider the user's mobility, and the connection between the user and the base station is not interrupted during the transmission of the file. $F=\{f_1, f_2, \cdots, f_n\}$ denotes a library of n files. We use the content popularity to indicate the probability that the user requests specific content, and assume that the content popularity meets the requirement of the Zip-f distribution with the parameter γ. We can define the content popularity of file f at SBS j of cell i as

$$\mathrm{p}_f^{i,j} = \frac{(q_f^{i,j})^{-\gamma}}{\sum_{f\in F}(q_f^{i,j})^{-\gamma}}, \quad \forall i\in C, j\in B^i, \gamma\ge 0 \qquad (1)$$

where $q_f^{i,j}$ denotes the descending order of the access frequency of file f in SBS j of cell i. In order to calculate the backhaul link load of the network, we also need to calculate the popularity distribution of files under the global network. Reference [11] and [12], we have

$$\mathrm{p}_f^i = \frac{\sum_{j\in B^i} p_f^{i,j}}{|B^i|}, \quad \forall i\in C, f\in F, \qquad (2)$$
$$\mathrm{p}_f = \frac{\sum_{i\in C} p_f^i}{|C|}, \quad \forall f\in F$$

where p_f^i defines the content popularity of file f in cell i and p_f define the content popularity of file f in global network. Let $N(u)$ denote all SBS sets that can establish a reliable connection with user u, denoted as the neighborhood of user u, then the probability that user u accesses the file f can be expressed as

$$\mathrm{p}_{u,f}^{i}=\frac{\sum_{j\in N(u)}p_f^{i,j}}{|N(u)|},\quad \forall i\in C,u\in U^i,f\in F \tag{3}$$

We denote the set of files in the cache of node j in cell i as A_j^i, $\forall i\in C, j\in B^i$, and use set $A=\{A_j^i \mid \forall i\in C, j\in B^i\}$ to denote the set of file sets cached by each base station in global network. For the convenience of follow-up work, we denote the union of files cached by cell i as $A^i=\bigcup_{j\in B^i}A_j^i$, $\forall i\in C$, and denote the union of files cached by global network as $A_{total}=\bigcup_{i\in C}A^i$. Then we can define the union of files cached by the neighborhood of user u as

$$A_{N(u)}^i=\bigcup_{j\in N(u)}A_j^i,\quad \forall i\in C \tag{4}$$

B. Backhaul Link Load

Once the user's request is served by the base station of the cell, there is no need to send the request to the remote internetwork and the backhaul links traffic can be seen as offloaded. Given a cache placement A, we can define the backhaul links load as

$$L(A)=\sum_{f\in\bar{A}_{total}}p_f\cdot S_f \tag{5}$$

where p_f denotes the popularity of file f in global network, S_f denotes the size of file f, $\bar{A}_{total}=\{f \mid f\in F\cap f\notin A_{total}\}$ denotes the complementary set of A_{total} .

All paragraphs must be indented. All paragraphs must be justified, i.e. both left-justified and right-justified.

C. Delay Cost Model

Since the size of the request message sent by the user is much smaller than the file size of the file f, the system delay of this model only considers the transmission delay generated by the file during transmission. Inspired by [13], we use file transmission delay as the weight of the edge $e(i,j)\in E$, i.e., $w(i,j)$. We denote the set of the feasible path from node i to node j as $P(i,j)=\{P_1(i,j),P_2(i,j),\cdots\}$ where any feasible path is expressed as $P_k(i,j)=\{e(i,v_1),e(v_1,v_2),\cdots,e(v_{|P_k(i,j)|-1},j)\}$ where $v_1,v_2,\cdots,v_{|P_k(i,j)|-1}$ denote the intermediate nodes of node i to path j, then the minimum file transmission delay transfer from node i to node j can be expressed as

$$t(i,j)=\min_{P_k(i,j)\in P(i,j)}\sum_{e(u,v)\in P_k(i,j)}w(u,v) \tag{6}$$

We denote the set of files requested by mobile user as O, the files requested by user u in cell i is denoted as $o_u^i\in O$. Given a cache files placement, we can define the file transmission delay to server requested $o_u^i\in O$ as

$$d(o_u^i,A)=\begin{cases}\min\limits_{\{j|j\in V_{bs},o_u^i\in A_j\}}t(u,j), & if\ o_u^i\in A_{total}\\ t(u,mb_i)+t_0, & else\end{cases} \tag{7}$$

where t_0 denotes the delay cost of transferring files from the Internet to MBS of the target cell, $t(u,mb_i)$ denotes the delay cost of transferring files from MBS to the target user. Then, the delay cost generated by user u can be define as

$$D_u^i(A)=\sum_{f\in o_u^i}p_{u,f}^i\cdot d(o_u^i,A),\quad \forall i\in C,u\in U^i \tag{8}$$

Finally, we can define the total delay cost of the global network as

$$D(A)=\sum_{i\in C}\sum_{u\in U^i}D_u^i(A) \tag{9}$$

IV. Multi-cell Edge Cloud Collaborative Cache Model

In order to achieve a cache placement with low latency and low link load, we formulate the cooperative caching problem of multi-cell as a Stackelberg game model, where the players of the game model are the control center of MBS and the base station group of the cell. On the one hand, the control center of MBS plays as the leader that is mainly used to reduce the backhaul link load of the network, and at the same time take into account the network delay state, and give a cache file placement for filling all cached entities. On the other hand, based on the cache placement policy given by the leader, the base stations in the cell play as the followers that exchange the files beneficially to further optimize the network delay caused by the cache placement. In the following part, we will analyze the cost function and formulation of the leaders and the followers respectively.

A. *Cost Model of Leader and Follower*

The decision made by the control center is to obtain a cache placement with low backhaul link load. However, solely consideration of the optimized backhaul link load is likely to reduce the number of copies of files in the cell and affecting the performance of the cache strategy in delay cost. Thus, the cost function of the control center can be defined as

$$T_{leader}(A_{leader}) = \alpha \cdot L(A_{leader}) + (1-\alpha)\cdot D(A_{leader}) \qquad (10)$$

where A_{leader} is the cache placement made by the control center, function L and function D are defined in (5) and (9), respectively. $\alpha \in (0,1)$ is a weighting factor.

Based on the cache placement policy given by the leader, the base station group exchange the files beneficially to further optimize the network delay. The cost function of base station group can be defined as

$$T_{follower}(A_{follower}) = D(A_{follower}) \qquad (11)$$

where function D are defined in (9).

B. *Problem formulation of Stackelberg game model*

Given a cache placement A, the control center changes the strategy A in order to reduce the cost function of the control center of this strategy. The optimal A of control center is obtained by solving the following problem.

$$\begin{aligned} \arg\min_{A} \quad & T_{leader}(A) = \alpha \cdot L(A) + (1-\alpha)\cdot D(A) \\ s.t. \quad & A = \{A_j^i \mid \forall i \in C, j \in B^i\} \\ & \sum_{f \in A_j^i} S_f \le S_j^i, \quad \forall i \in C, j \in B^i \\ & A_j^i, o_u^i \in F, \quad \forall i \in C, j \in B^i, u \in U^i \\ & \alpha \in (0,1) \end{aligned} \qquad (12)$$

the utility of base station group is affected by the total delay of the network model. The base station of each cell adjusts the cache file for the purpose of maximizing the overall network benefit, so we build the following optimization model for the follower's cost function:

$$\begin{aligned} \arg\min_{A} \quad & T_{follower}(A) = D(A) = \sum_{i \in C} \sum_{u \in U^i} \sum_{f \in o_u^i} p_{u,f}^i \cdot d(o_u^i, A) \\ s.t. \quad & A = \{A_j^i \mid \forall i \in C, j \in B^i\} \\ & \sum_{f \in A_j^i} S_f \le S_j^i, \quad \forall i \in C, j \in B^i \\ & A_j^i, o_u^i \in F, \quad \forall i \in C, j \in B^i, u \in U^i \end{aligned} \qquad (13)$$

The equilibrium of the Stackelberg game indicates that the decision of the leader and the follower is close to each other and the solution is satisfactory to both parties. Given the initial cache strategy A, the optimal decision of the base station group can solve the problem (13), and the optimal decision of the control center is the solution (12), so the Stackelberg model Nash equilibrium solution of this paper can be expressed as a strategy (A, B). Nash equilibrium has another way of expression. Take the control center as an example. For any cache strategy A_{leader}, if A^{*}_{leader} is the Nash equilibrium of the control center, then there is always

$$T_{leader}(A^{*}_{leader}) \le T_{leader}(A_{leader}) \qquad (14)$$

V. STACKELBERG GAME SOLUTION

The optimization problem in section IV belongs to the base station buffer problem. [9] has proved that this kind of optimization problem belongs to the NP-complete problem that finding the optimal solution of the problem with a single algorithm is difficult. Combining the characteristics of the model and the two-layer characteristics of the Stackelberg game, we design an iterative alternating Stackelberg Model Solution Algorithm (SMSA) to solve the problem.

The solution process of the model is as follows: The model is divided into two sub-problems: the control center sub-problem and the base station group sub-problem. First, the control center executes the Improved Shuffled Frog Leaping Algorithm (ISFLA) with the random cache placement A_{init} as the initial solution, and the obtained new cache placement A_{leader} is used as the initial solution of the Greedy Swapping Algorithm (GSA) of the base station group. Performing a file exchange yields a new cache placement $A_{follower}$ as a new round of initial resolution execution loop alternate iterations. When the number of iterations reaches the maximum number or the solution no longer changes, the algorithm will terminate and output cache placement A_{last} as the equalization solution for the Stackelberg game. The SMSA is shown in Algorithm 1.

Algorithm 1: Stackelberg Model Solution Algorithm (SMSA)

Input: $C, M, B, U, G(V,E), F$

Output: A_{last}

```
1:   $A \leftarrow \varnothing,\ k \leftarrow 1,\ \max\_iter\_num \leftarrow 1000$
2:   $A_{init} \leftarrow LTGA(C,B,U,F)$
3:   $A_{last} \leftarrow A_{init}$
4:   While $k < \max\_iter\_num$ do
5:     $A_{leader} \leftarrow ISFLA(A_{last},C,B,U,F)$
6:     $A_{follower} \leftarrow GSA(A_{leader},C,B,U,F)$
7:     If $A_{last} = A_{follower}$ then
8:       Exit
9:     Else
10:      $A_{last} \leftarrow A_{follower}$
11:      $k \leftarrow k+1$
12:    End if
13:  End while
14:  Return $A_{last}$
```

A. Improved Shuffled Frog Leaping Algorithm (ISFLA)

Traditional SFLA is a group-based collaborative search algorithm generated by natural biological imitation, imitating the process of frog foraging in a limited space [19]. The wetland where frogs live represents the solution space. Each frog represents a solution to the solution space, and then these frogs are partitioned into different subgroup, each performing a local search. Within each subgroup, every individual frog has its own idea that may be infected by other frogs. After a certain number of evolutionary steps, ideas are passed between the subgroup in a shuffling process. The local search and the shuffling process continue until the defined convergence criteria are met [20][21].

The Improved Shuffled Frog Leaping Algorithm(ISFLA) in this paper is optimized on the traditional SFLA for both the update mechanism and the number of update solutions for the algorithm in the local search phase: (1) Improving the number of solutions in the internal search: the traditional SFLA only update one worst solution per round in the internal search phase, which has a great impact on the convergence speed of the algorithm. In order to speed up the convergence of the algorithm, we update N worst solutions each time; (2) Modifying the update mechanism of the solution: The traditional SFLA local search process is only suitable for solving continuous problems, and it is not convenient for us to deal with discrete combinatorial optimization problems. [15] proposes a discrete shuffled frog leaping algorithm for solving multi-kock problems, and proves that the algorithm can solve the feasibility and effectiveness of the multi-package problem. In order to catch up with the update of the solution of the model and reference [15], this paper proposes a stochastic cross-solution update mechanism. Application of the improved algorithm is summarized in the following steps:

Step 1: Initialize. Set N_{ethnic} and N_{group}, N_{ethnic} is the number of subgroup, and N_{group} is the number of frogs in each subgroup. Therefore, the total number of frogs in the swamp is defined as $N_{sum} = N_{ethnic} \cdot N_{group}$.

Step2: Producing an initial population. In order to make the ISFLA algorithm fully demonstrate the global search ability and avoid falling into local optimum, we randomly generate N cache placements as the initial population of the algorithm.

Step3: Subgroup division. In this paper, the cost function (10) of the control center is used as the fitness function of the algorithm. Each frog calculates the fitness value through the fitness function and sorts the N_{sum} frogs in descending order of fitness values. Allocate them evenly in N_{ethnic} arrays $P^i_{ethnic}, i = 1,2,\cdots,N_{ethnic}$ to get the subgroup set $P_{ethnic} = \{P^i_{ethnic} \mid i = 1,2,\cdots,N_{ethnic}\}$. In this process, the first frog goes to the first subgroup, the second frog goes to the second subgroup, frog N_{ethnic} to the $N_{ethnic} - th$ subgroup, frog $N_{ethnic} + 1$ to the first subgroup, and so on.

Step4: Local Search within each subgroup. First, the frog position with the global best fitness value is identified as X_{best}. In each subgroup, the frogs position with the best and worst fitness value are represented as X^i_{best} and X^i_{worst}, respectively, $i = 1,2,\cdots,N_{enthnic}$. Then, in each subgroup, a certain update process is applied to enhance N_{worst} frogs with the worst fitness value. In this update process, we randomly select the cache file of the base station with the highest fitness solution to cover the cache files of the N_{worst} worst solution corresponding base stations. If this process leads to a better solution, it will replace the worst frog. Otherwise, the update process are repeated, but in this case X^i_{best} is replaced by X_{best}. If no improvement takes place, a new solution will be randomly produced to replace that frog.

Step5: Shuffling process. After each subgroup completes the internal search, remix the solutions of all subgroups, and then return to step3 to repeat the above process until the convergence criteria are fulfilled. The output of the best fitness value solution is the final solution of the algorithm.

B. Greedy Swapping Algorithm(GSA)

Inspired by the Local Greedy Swapping algorithm of [13], we propose the Greedy Swapping Algorithm (GSA) to exchange beneficial cache between cells, thus greedily reducing the delay cost of the global network.

In each cell, the relationship between the base stations can be regarded as a tree with MBS being the root node and s SBS being the leaf nodes. We convert this tree into a tree with the depth of s+1 using the child brother method, as shown in Fig. 3. The algorithm takes the tree of Fig.4 as the reference. In each iteration, starting with the node with the depth of s, the beneficial exchange set between the node and its child node will be found and exchanged. After the execution, the node of depth s-1 node performs the above operation and so on, end this iteration until the depth is 1.

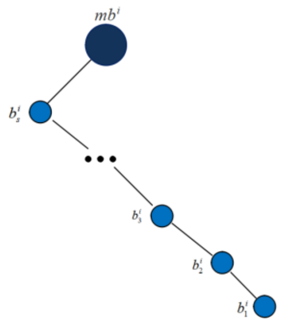


Fig. 3. A cell model expressed as a tree with a depth of s + 1

Before describing the algorithm, we first define the concept of a beneficial exchange set: set the maximum number of iterations K, we define the beneficial exchange set of the base station j of the cell i at the k-th iteration as $O_j^i(k)$. Set $O_j^i(k)$ is composed of triples that each triplet represents a beneficial exchange, e.g. if there is a triple (v, f_1, f_2), then the exchange of the f_1 of the base station j with the file f_2 of its parent node will reduce the delay cost in global network, the set of beneficial exchange can be expressed as

$$O_j^i(k) = \{(v, f_1, f_2) \mid f_1 \in A_j^i, f_2 \in A_v^i, \\ T_{follower}(A_{exchange}) < T_{follower}(A)\} \qquad (13)$$

where the function $T_{follower}$ is defined (11), indicating the delay cost of the cache placement. $T_{follower}(A)$ represents the delay cost generated by the cache placement A before the file exchange, and $T_{follower}(A_{exchange})$ represents the delay cost generated by the cache placement $A_{exchange}$ after the exchange of the file. What's more, we use $O^i(k) = \{O_j^i(k) \mid \forall j \in B^i\}$, $k \le K$ to represent the beneficial exchange set of all base stations of cell i.

The GSA algorithm takes the result of the ISFLA algorithm as the initial solution and performs iteration K times for each cell. Each iteration finds all beneficial exchange sets $O^i(k), i = 1,2,\cdots,m$ and exchanges all the files recorded in the triple and then proceeds to the next iteration. GSA is shown in Algorithm 2.

Algorithm 2: Greedy Swapping Algorithm (GSA)

Input: A_{SFLA}, C, B, F

Output: A_{GSA}

```
1:    $K \leftarrow 100, A_{GSA} \leftarrow A_{SFLA}$
2:    For each $i \in C$ do
3:        $k \leftarrow 0$
4:        While $k \le K$ do
5:            For each $i \in B^i$ do
6:                $O_j^i(k) = \{(pa(j), f_1, f_2) \mid f_1 \in A_j^i, f_2 \in A_{pa(j)}^i, T_{follower}(A_{change}) < T_{follower}(A)\}$
7:                For each $(pa(j), f_1, f_2) \in O_j^i(k)$ do
8:                    $A_j^i = A_j^i \cup \{f_2\} \setminus \{f_1\}$
9:                    $A_{pa(j)}^i = A_{pa(j)}^i \cup \{f_1\} \setminus \{f_2\}$
10:               End for
11:           End for
12:           $k \leftarrow k+1$
13:       End while
14:   End for
15:   Return $A_{GSA}$
```

where $pa(j)$ represents the parent node of node j in the tree. The output of the GSA algorithm will be the new round of initial solution to repeat SMSA until the convergence criteria are satisfied, and the final result of SMSA is the final cache decision of the global network.

VI. Simulation Results

In this section, we use simulations to evaluate the proposed Stackelberg Model Solution Algorithm(SMSA) on the three aspects of total delay cost, backhaul link load and inter-cell cooperation. Unless otherwise stated, the simulation consists of the following initial settings:

1. We implement 3 cells in a 5000 x 5000 m2 regular grid. Each cell has a coverage range of 600m where MBS is located in the center of cell and 8 SBS are evenly scattered over the coverage region. The communication range of MBS is 600m and the communication range of SBS is 250m.

2. We use the YouTube request trace data collected by the campus of the University of Massachusetts Amherst during the day of March 12, 2008 (refer to: http://traces.cs.umass.edu/index.php/Network/Network) and select the top 200 files with the most requested requests to join the file library F. We assume the file popularity p follows the Zipf distribution with parameter 1.5 and the request ranking of the file is based on the YouTube request trace data. We set: $S_f = 10Mb$, $S^i_{b_j} = 50Mb,$

$, i = 1,2,3,\ \ j = 1,2,\cdots,8$ and $S^i_{mbs} = 100Mb$ $, i = 1,2,3$.

3. We set the latency from a SBS to a mobile user as 10ms. The latency from MBS to a SBS (or from a SBS to MBS) in the same cell is denote to be 10ms, and the latency from a MBS to other MBS of different cell is assigned to be 50ms, the latency from the Internet of remote cloud to a MBS assigned to be 130ms.

In order to demonstrate the advantages and disadvantages of the proposed SMSA algorithm, this paper compares the SMSA algorithm with the random cache algorithm, [4] proposed the Depth First Greedy(DFG) algorithm and the Local Greedy Swapping(LGS) algorithm. Then, we evaluate the impact of the number of mobile user and the number of SBS on the performance of algorithm through Simulation.

A. Impact of the Number of Users

We set the number of users from 100 to 200 gradually, and the other environments are recorded the experimental results of each scheme under the initial settings to evaluate the impact of changes in the number of users on various algorithms.

Fig. 4a shows the change in system cost caused by cache placement under the change in the number of users. And Fig. 4b and Fig. 4c show the change in delay cost and linked load cost respectively caused by cache placement under the change in the number of users. We can observe that the proposed SMSA scheme outperforms the other three schemes in terms of delay cost and backhaul link load optimization, resulting in the lowest network delay and minimum backhaul link load.

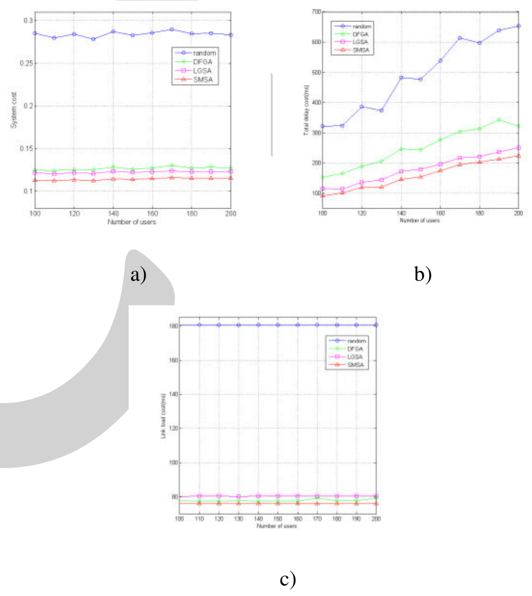
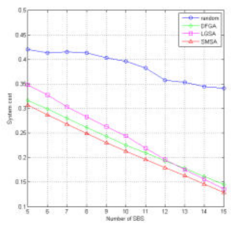


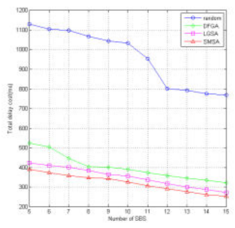
Fig. 4 Numerical results for a) System cost with respect to the number of users b) Total delay cost with respect to the number of users c) Linked load cost with respect to the number of users

B. Impact of the Number of SBSs

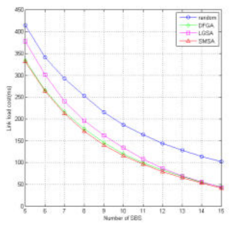
As shown in Fig. 5a, the SMSA cache scheme proposed in this paper can maintain the optimal performance with lowest system cost as the number of SBS changes. The performance of the DFG algorithm is close to our algorithm. Observing Fig.5b and Fig.5c, it can be found that the performance of the DFG algorithm is close to our SMSA in terms of backhaul link load, but the DFG algorithm is not good enough in optimizing the total system latency cost.



a)



b)

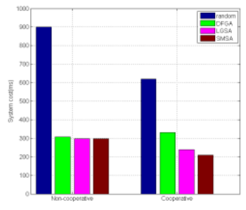


c)

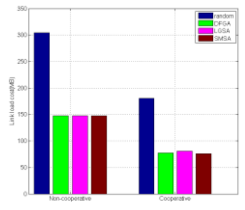
Fig. 5. Numerical results for a) System cost with respect to the number of SBS b) Total delay cost with respect to the number of SBS c) Linked load cost with respect to the number of SBS

C. *Impact of cell collaboration*

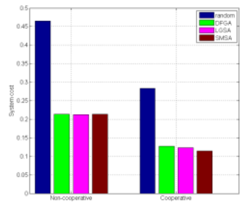
Most of the existing researches on mobile edge cloud caching focus on collaborative caching in single-cell scenarios [8]-[10]. In order to verify the effectiveness of the multi-cell cooperative caching proposed in this paper, we evaluate the performance of each solution in terms of system cost, total latency cost, and backhaul link load, respectively, under conditions where inter-cell support is coordinated and the simulation results are shown in Fig. 6a, 6b and 6c respectively. It can be seen from the figure that the performance of all algorithm supporting inter-cell cooperative caching is much stronger than that of non-cell cooperative caching. For example, the system cost of SMSA solution is reduced by 46% compared to the non-cooperative condition, the total system time delay is reduced by 29.9%, and the backhaul link load by 48%. In addition, it can be found that SMSA is more suitable for solving the cooperative cache problem of multi-cells than the greedy algorithm.



a)



b)



c)

Fig. 6. Numerical results for cell collaboration

VII. Conclusion

In this paper, we propose a multi-cell cellular network model, which uses mobile edge technology and is equipped with an edge server with caching function in each base station. Therefore, it is considered to be able to solve the optimization problem of multi-cell cooperative caching. We formulate the problem as a Stackelberg game problem and decompose it into sub-problems of two types of players. Then, we design an iterative alternating Stackelberg Model Solution Algorithm (SMSA) to solve the problem, and we use the numerical simulation experiment to evaluate the performance of the proposed algorithm. The simulation results show that the proposed algorithm outperforms the greedy algorithm proposed by other researches in terms of delay cost optimization and reduction of backhaul link load, and proves the effectiveness of our proposed algorithm in solving multi-cell cooperative caching problem.

REFERENCES

[1] Liu J , Yan H , Li Y , et al. (2016). Cache Behavior Characterization and Validation over Large-scale Video Data. IEEE Transactions on Circuits and Systems for Video Technology, 1-1.

[2] Condoluci, M. , Dohler, M. , Araniti, G. , Molinaro, A. , & Zheng, K. . (2015). Toward 5g densenets: architectural advances for effective machine-type communications over femtocells. IEEE Communications Magazine, 53(1), 134-141.

[3] Liu, H. , Eldarrat, F. , Alqahtani, H. , Reznik, A. , De Foy, X. , & Zhang, Y. . (2017). Mobile edge cloud system: architectures, challenges, and approaches. IEEE Systems Journal, 1-14.

[4] Wang, S. , Zhang, X. , Zhang, Y. , Wang, L. , Yang, J. , & Wang, W. . (2017). A survey on mobile edge networks: convergence of computing, caching and communications. IEEE Access, 1-1.

[5] Ahmed, A. , & Ahmed, E. . (2016). A Survey on Mobile Edge Computing. 10th IEEE International Conference on Intelligent Systems and Control, (ISCO 2016). IEEE.

[6] Mao, Y. , You, C. , Zhang, J. , Huang, K. , & Letaief, K. B. . (2017). A survey on mobile edge computing: the communication perspective. IEEE Communications Surveys & Tutorials, 19(4), 2322-2358.

[7] Roman, R. , Lopez, J. , & Mambo, M. . (2016). Mobile edge computing, fog et al.: a survey and analysis of security threats and challenges. Future Generation Computer Systems, S0167739X16305635.

[8] Chen, L., & Jie, X. (2017). Collaborative service caching for edge computing in dense small cell networks.

[9] Cui, X. , Jiang, Y. , Chen, X. , Zheng, F. C. , & You, X. . (2018). Graph-based cooperative caching in fog-ran.

[10] Huang, X. , Zhao, Z. , & Zhang, H. . (2016). Latency analysis of cooperative caching with multicast for 5G wireless networks. the 9th International Conference. ACM.

[11] Li, X. , Wang, X. , Zhu, C. , Cai, W. , & Leung, V. C. M. . (2015). Caching-as-a-Service: Virtual caching framework in the cloud-based mobile networks. Computer Communications Workshops. IEEE.

[12] Shanmugam, K. , Golrezaei, N. , Dimakis, A. G. , Molisch, A. F. , & Caire, G. . (2013). Femtocaching: wireless content delivery through distributed caching helpers. IEEE Transactions on Information Theory, 59(12), 8402-8413.

[13] Pacifici, V. , Josilo, S. , & Dan, G. . (2016). [ieee 2016 28th international teletraffic congress (itc 28) - w ü rzburg, germany (2016.9.12-2016.9.16)] 2016 28th international teletraffic congress (itc 28) - distributed algorithms for content caching in mobile backhaul networks. 313-321.

[14] Yue, M. , Hu, T. , Guo, B. , & Guo, X. . (2010). The research base on memetic meta-heuristic Shuffled Frog-Leaping Algorithm. 2009 2nd International Conference on Power Electronics and Intelligent Transportation System (PEITS). IEEE.

[15] Ma, Z., & Shu, S. . (2011). Shuffled Frog Leaping Algorithm for Solving Multiple Knapsack Problem. Computer & Digital Engineering, 19-21.

[16] Du, J., Yuan, Z., & Wang, J. . (2017). Shuffled Frog Leaping Algorithm Based on Grey Prediction Theory. Transactions of China Electrotechnical Society(15).

[17] Cui, W., Liu, X., Wang, W., Wang, J. . (2012). Survey on shuffled frog leaping algorithm. Control and Decision(04), 3-8+15.

[18] Chen, Z. , Lee, J. , Quek, T. Q. S. , & Kountouris, M. . (2017). Cooperative caching and transmission design in cluster-centric small cell networks. IEEE Transactions on Wireless Communications, 1-1.

[19] Eusuff, M. M. , & Lansey, K. E. . (2003). Optimization of water distribution network design using the shuffled frog leaping algorithm. , 129(3), 210-0.

[20] Fallah-Mehdipour, E. , Mari?O, M. A. , Bozorg Haddad, O. , & Orouji, H. . (2013). Estimation of muskingum parameter by meta-heuristic algorithms. Proceedings of the ICE - Water Management, 166(6), 315-324.

[21] Li, Y. , & Yan, Z. . (2019). Improved shuffled frog leaping algorithm on system reliability analysis. Brain Informatics, 6(1).