Data Caching Optimization With Fairness in Mobile Edge Computing

Jingwen Zhou, Feifei Chen, *Member, IEEE*, Qiang He, *Senior Member, IEEE*, Xiaoyu Xia, *Member, IEEE*, Rui Wang, and Yong Xiang, *Senior Member, IEEE*

Abstract—Mobile edge computing (MEC) provides a new computing paradigm that can overcome the inability of the traditional cloud computing paradigm to ensure low service latency by pushing computing power and resources to the network edge. Many studies have attempted to formulate edge data caching strategies for app vendors to optimize caching performance by caching the right data on the right edge servers. However, existing edge data caching approaches have unfortunately ignored fairness, which is an important issue from the app vendor's perspective. In general, an app vendor needs to cache data on edge servers to serve its users with insignificant latency differences at a minimum caching cost. In this paper, we make the first attempt to tackle the fair edge data caching (FEDC) problem. Specifically, we formulate the FEDC problem as a constraint optimization problem (COP) and prove its $\mathcal{NP}$-hardness. An optimal approach named FEDC-OPT is proposed to find optimal solutions to small-scale FEDC problems with integer programming technique. In addition, an approximate algorithm named FEDC-APX is proposed to find approximate solutions in large-scale FEDC problems. The performance of the proposed approaches is analyzed theoretically, and evaluated experimentally on a widely-used real-world data set against four representative approaches. The experimental results show that the proposed approaches can solve the FEDC problem efficiently and effectively.

Index Terms—Approximation algorithm, edge data caching, fairness, integer programming, mobile edge computing, optimization

1 Introduction

The number of mobile devices and Internet-of-Things (IoT) has increased explosively in the past decade. The transmission of massive data produced by and for these devices incurs heavy network traffic and consumes excessive network resources [1], [2]. As a result, it has become increasingly difficult for the traditional centralized cloud computing paradigm to fulfill various applications' demand for low service latency [3], i.e., virtual reality (VR), interactive gaming, smart vehicles, etc. Applications deployed on remote cloud servers cannot guarantee low service latency for users due to unpredictable delays over the wide-area network [4]. Moreover, it is difficult to further reduce service latency for these applications with the long physical geographic distance between remote cloud servers and users [5]. This has been acknowledged as an evident weakness of the cloud computing paradigm since mobile users are increasingly sensitive to service latency [6].

As a key 5G enabler technology, mobile edge computing (MEC) has emerged to overcome the inability of cloud computing to ensure low latency [1], [7], [8]. In the MEC environment, edge servers are deployed on base stations or attached to access points geographically close to end-users [9]. App vendors can hire resources, such as CPU, storage space, memory, and bandwidth on edge servers for hosting their applications so that their users can access their services with low latency [10].

As edge servers become the rapidly-increasing number of users' access points to online services, a tremendous amount of data replicas will be transmitted via edge servers [11]. Selecting edge servers and caching the popular data on these edge servers can ensure low data caching latency for mobile users and in the meantime, reduce the traffic pressure on the backhaul network [12], [13]. The ability of MEC to enable *edge data caching* offers app vendors many new opportunities. However, it also raises various unprecedented challenges, due to its unique constraints: 1) proximity constraint - a mobile user can only access the edge servers that cover the user [6]; 2) capacity constraint - the resources available on edge servers are constrained due to the limitation of their physical sizes [14]; and 3) latency constraint - data can only be transmitted via a fixed number of hops among different edge servers to serve a user [10]. To tackle these challenges, many researchers have attempted to formulate edge data caching strategies for app vendors with various optimization objectives, e.g., minimizing users' average data retrieval latency [15], minimizing the data caching cost incurred [16], or the combination of both [17].

- *Jingwen Zhou, Feifei Chen, and Yong Xiang are with the School of Information Technology, Deakin University, Geelong, VIC 3125, Australia. E-mail: {jingwen.zhou, feifei.chen, yxiang}@deakin.edu.au.*
- *Qiang He is with the Department of Computing Technologies, Swinburne University of Technology, Melbourne, VIC 3122, Australia. E-mail: qhe@swin.edu.au.*
- *Xiaoyu Xia is with the School of Mathematics, Physics and Computing, University of Southern Queensland, Toowoomba, QLD 4350, Australia. E-mail: xiaoyu.xia@usq.edu.au.*
- *Rui Wang is with Data61, Commonwealth Scientific and Industrial Research Organisation, Canberra, ACT 2601, Australia. E-mail: r.wang@csiro.au.*

Manuscript received 12 October 2021; revised 21 July 2022; accepted 2 August 2022. Date of publication 10 August 2022; date of current version 12 June 2023.
This work was supported in part by Australian Research Council Discovery Projects under Grants DP180100212 and DP200102491, and in part by Linkage Project under Grant LP190100594.
(Corresponding authors: Feifei Chen and Qiang He.)
Digital Object Identifier no. 10.1109/TSC.2022.3197881

Pursuing to achieve various optimization objectives, existing edge data caching approaches have unfortunately neglected the issue of fairness. Under the above-mentioned latency constraint, a maximum data retrieval latency is ensured for all the users as the minimum latency constraint. However, individual users may still receive differentiated quality-of-experience (QoE). Take VR for example. High-quality VR requires a 20 ms end-to-end latency or lower to avoid motion sickness in the long term [18]. One-hop players, i.e., players that can retrieve data from edge servers within one hop, will not experience motion sickness. Two-hop players and three-hop players may not experience motion sickness in the short term, but will in the long term. Thus, data retrieval via different hops impacts users' QoE significantly. Some users' ultralow latency may come at the cost of other users' barely-acceptable latency. This may not be an issue in the short term. However, in the long term, it will undermine users' overall quality of experience (QoE) [19] and jeopardize app vendors' long-term business goals [20], similar to the QoE fairness concern in other domains [21], [22].

Assuming that a Facebook video goes viral in an area, a straightforward caching strategy is to cache the video on every edge server for serving all the mobile users in that area. In this way, all the mobile users' received data retrieval latency is minimized because all of them can retrieve the video data from their nearby edge servers. There is no fairness issue because there is no significant difference between data retrieval latencies. However, this strategy may not be realistic because it is not guaranteed that Facebook can always hire adequate resources on every edge server in the area for caching its data due to the edge servers' limited storage resources and the tough competition for those computing resources among different app vendors [23]. Even if it is feasible, it is not likely to be cost-effective because Facebook will have to pay excessively for the computing resources hired on edge servers for caching the data replica based on the pay-as-you-go pricing model. On the other hand, if an app vendor pursues to minimize the overall data caching cost generically measured by the number of data cached on the edge servers in an area, it will cache data on a limited number of edge servers in the area. Such caching strategies are most likely to cause unfairness among users. Some of them will be able to retrieve the data replica directly from nearby edge servers while the others will have to retrieve it from edge servers multiple hops away. Users may experience undesirably differentiated QoE. Again, take VR for example example. It would be ideal for all the users to access the VR service from edge servers within one hop, which would prevent motion sickness. However, due to the capacity constraint, an edge server may not always have adequate capacity to accommodate all the users within one hop. In addition, the computing resources on edge servers are expensive and app vendors' competition for those resources is tough. It is difficult, if not impossible, for an app vendor to always secure adequate computing resources on edge servers for serving all the users within one hop. Thus, users' divergent quality-of-experience (QoE) is common in practice and ensuring long-term QoE fairness is important from the app vendor's perspective. The trade-off between fairness and caching cost further complicates the problem to formulate cost-effective edge data caching strategies, in addition to the various unique constraints in the MEC environments discussed above.

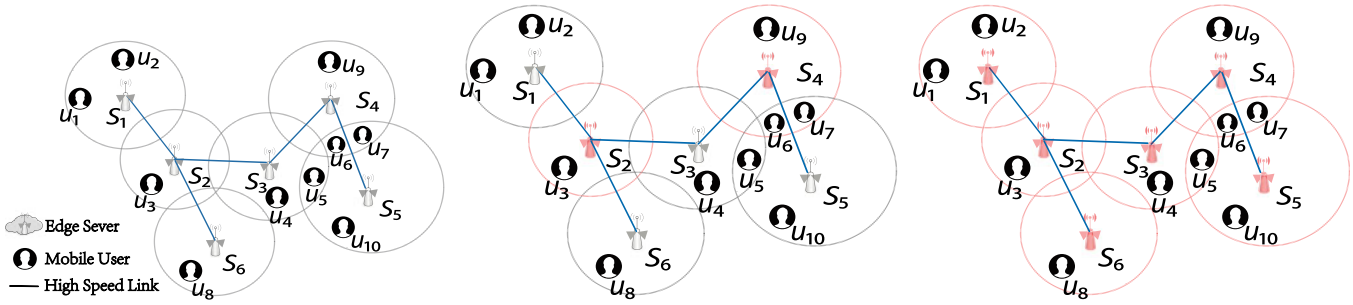
In this paper, we make the first attempt to study this fair edge data caching (FEDC) problem. Our aim is to help app vendors formulate fair edge caching strategies that maximize the fairness among users while fulfilling the proximity constraint, the capacity constraint, and the latency constraint at minimum caching costs. The main contributions of this paper are:

- Modeling and formulating the FEDC problem as a constrained optimization problem (COP), and proving its $\mathcal{NP}$-hardness.
- Proposing an optimal approach named FEDC-OPT to solve small-scale FEDC problem based on integer programming technique.
- Proposing a sub-optimal approach named FEDC-APX to find approximate solutions to large-scale FEDC problems efficiently, and proving its approximation rate theoretically.
- Evaluating FEDC-OPT and FEDC-APX against four representative approaches through experiments conducted on a widely-used real-world data set.

The rest sections of this paper are organized as follows. Section 2 presents a fundamental motivating example to motivate this research work. Section 3 presents the formulation of the FEDC problem and proves the $\mathcal{NP}$-hardness with theory. Section 4 presents our optimal approach and approximation approach in detail. Section 5 shows the experimental evaluations with our proposed approaches and the relevant approaches. Section 6 provides a review of the related work in existed studies. Section 7 is the conclusion of this paper, this section also points out the future work.

2 Motivation Example

Video streaming is a typical example that may benefit from edge caching. Fig. 1a is an example EDC scenario with six edge servers, i.e., $s_1, \ldots, s_6$, and 10 users, i.e., $u_1, \ldots, u_{10}$. Let us assume that a Facebook video goes viral in a certain area. Many users in the area would request this data. Facebook would like to cache video replicas on edge servers to cover all its users so that they can access the video under the latency constraint, i.e., within 1 hop over the edge server network. From Facebook's perspective, the benefit produced by caching the video on the edge servers is the reduction in users' video retrieval latency compared against the latency constraint [10], [13], [24]. Given the latency constraint of 1 hop, if a mobile user can retrieve the video data from a nearby edge server via 0 hop, the benefit produced for this mobile user is 2. If a mobile user needs to retrieve the data replica from an edge server via 1 hop, the data caching benefit for this user is 1. Otherwise, the benefit is 0. Similar to [10], [13], [24], caching cost is measured by the number of data replicas cached on edge servers. Modeling data retrieval latency and caching cost in this way will allow easy integration of specific latency and cost models [10], [13], [24]. In this paper, we study quasi-static EDC scenarios, where users' data demands do not vary during the formulation of the EDC strategy, similar to many studies of mobile edge computing [6], [25], [26], [27]. When new users join in



(a) EDC Scenario (b) EDC strategy R_1 (c) EDC strategy R_2

Fig. 1. An example EDC scenario.

the next time slot, a new EDC strategy can be formulated to accommodate their data demands.

As shown in Fig. 1a, none of the edge servers in this area can serve all the users in this area alone. Facebook needs to cache the video on at least two edge servers to cover all the users within 1 hop. Fig. 1b presents a caching strategy R_1, which caches the video on edge servers s_2 and s_4 to serve all the users within zero or 1 hop. The caching benefit produced by R_1 is 14. Specifically, at the cost of 2, i.e., caching two video replicas, the caching benefits produced for users u_3, u_6, u_7 and u_9 are 2 each, while the caching benefits produced for users u_1, u_2, u_4, u_5, u_8, and u_{10} are 1 each. Out of the 10 users, 4 can retrieve the video via zero hop and 4 via 1 hop. If we employ the widely-used Jain's fairness index [28] to evaluate the fairness degree among 10 users' data retrieval latency, the result is 0.8909. Another data caching strategy is presented in Fig. 1c. This strategy caches the video on all of the 6 edge servers. Compared with R_1 (Fig. 1b, the overall caching benefit produced by this strategy is higher, i.e., 20 versus 14. All the users achieve the same caching benefit. Thus, the highest possible fairness is achieved, with Jain's fairness index of 1.0. However, these come at a high caching cost of 6. It is not a cost-effective caching strategy. In fact, this strategy may not even be possible because Facebook may not be able to cache the video on every edge server due to individual edge servers' constrained storage resources. Similarly, if all the users achieve equally low benefits, the fairness among them is also high. However, such data caching strategies are not practical or preferable.

Both fulfilling the latency constraint, Strategies R_1 incurs the lowest caching cost while R_2 achieves the highest caching benefit and fairness. There are a lot of trade-off solutions in-between. From most app vendors' perspectives, a fair and cost-effective caching strategy should maximize the ratio of fairness over cost while fulfilling all the constraints. They need to be able to solve this fair edge data caching (FEDC) problem effectively and efficiently.

3 Problem Formulation

In this section, we formulate the proposed FEDC problem, and prove the $\mathcal{NP}$-hardness of FEDC problem based on the minimum dominating set problem (MDS) problem.

3.1 Problem Statement

Assume there is an undirected graph $G(S, E)$ that denotes an edge server network in a specific geographic area, where S denotes the set of the nodes in graph G and E denotes the set of the edges in G. Let each node $s_i \in S(1 \leq i \leq n)$ represent an edge server in the edge server network, and each edge $e_{i,j} \in E$ is the connecting link between s_i and s_j.

As discussed in Section 1, an EDC strategy needs to maximize the ratio of fairness over caching cost (referred to as the *fairness-cost ratio* hereafter) while fulfilling a set of constraints, in particular the latency constraint h. Thus, the FEDC problem can be modeled as a COP. We summarize the key notations in this paper in Table 1.

An EDC strategy can be represented with a vector $R = < r_1, \ldots, r_n >$, where $r_i = 1$ indicates that the data is cached on edge server $s_i \in S$, and $r_i = 0$ otherwise. Let $a_i = 1$ $(1 \leq i \leq n)$ represent that s_i has efficient capacities to cache the data. The capacity constraint discussed in Section 1 can be enforced with $r_i \leq a_i$. In this way, the app vendor can cache a data replica on an edge server s_i (r_i = 1), only when there are sufficient resources ($a_i = 1$) on s_i. If there are insufficient resources on s_i ($a_i = 0$), the data replica cannot be cached on s_i (r_i = 0)

$$r_i = \begin{cases} 1 & if \text{ data cached on } s_i \text{ with sufficient resources} \\ 0 & if \text{ data not cached on si} \end{cases} \tag{1}$$

TABLE 1
Summary of Notations

Notation	Description
a_i	if or not s_i have sufficient resources to cache data
$D_{i,j}$	distance between s_i and s_j
$E = \{e_1, \ldots, e_l\}$	set of edges in G
$FC(R)$	fairness-cost ratio achieved by R
G	edge server network
h	latency constraint
$J(R)$	Jain's fairness degree achieved by R
m	number of mobile users
n	number of edge servers
Q	caching benefit matrix
$q_{i,j}$	benefit of caching data on s_i for users covered by s_j
q_{u_k}	maximum caching benefit for user u_k
$R = \{r_1, \ldots, r_n\}$	EDC strategy
$S = \{s_1, \ldots, s_n\}$	set of edge servers
$U = \{u_1, \ldots, u_m\}$	set of mobile users
$U(s_i)$	set of mobile users covered by s_i
$U(R)$	set of users covered by R

Now, the data caching cost incurred by a caching strategy R, measured by the number of cached data replicas, can be calculated as

$$cost(R) = \sum_{i=1}^{n} r_i \tag{2}$$

Similar to [5], [10], the benefit produced by R for a user is generically measured by the reduction in its data replica retrieval latency against the latency constraint, calculated by the number of hops over the edge server network. As discussed in Section 2, edge servers can share cached data within a limited number of hops under the latency constraint. Let h denote this latency constraint, $q_{i,j}$ denote the benefit produced for any user located in server s_j's coverage area by caching data on edge server s_i. Now, $q_{i,j}$ can be calculated as follows:

$$q_{i,j} = \begin{cases} h - D_{i,j} + 1 & if\ D_{i,j} \le h \\ 0 & if\ D_{i,j} > h \end{cases} \tag{3}$$

where $D_{i,j}$ is the distance between edge servers s_i and s_j in hops.

To avoid blank coverage areas, the adjacent edge servers coverage areas often have intersections [6], [29]. A user $u_k \in U$ within in an intersection area that is covered by multiple edge servers can access data replica from any of these edge servers or the neighbour edge servers as long as the latency constraint is not violated. We assume that in such cases, u_k will retrieve data from the nearest edge server s_i, producing the highest caching benefit

$$q_{u_k} = \max\{q_{i,j} | r_i = 1, u_k \in U_j, \forall s_i, s_j \in S\} \tag{4}$$

where U_j is the set of mobile users that covered by edge server s_j.

In this study, we employ the widely-used Jain's fairness index [28] to quantify the fairness degrees of edge data caching strategies. Given a strategy R caching data for a set of users U, the fairness degree of R, denoted by $J(R)$, is calculated as follows:

$$J(R) = \frac{\left(\sum_{u_k \in U} q_{u_k}\right)^2}{|U| \sum_{u_k \in U} \left(q_{u_k}\right)^2}. \tag{5}$$

This index rates the fairness of the caching benefits produced for the users, ranging from $1/m$ (worst case) to 1 (best case). Its value is 1 when the same caching benefit is produced for all the users. For example, the EDC strategy R_2 presented in Fig. 1c achieves Jain's fairness index of 1.0 because it allows all the mobile users to retrieve the video from their nearby edge servers via 1 hop.

In an EDC scenario, all the mobile users must be able to access data from edge servers under the latency constraint. Calculated with Eq. (4), the caching benefit produced for any individual mobile user fulfills

$$0 < q_{u_k} \le h + 1,\ \forall u_k \in U, \tag{6}$$

where h is the latency constraint, i.e., the maximum number of hops over the edge server network via which mobile users can retrieve cached data.

As discussed in Section 2, there is a trade-off between caching cost and caching fairness. To measure this trade-off, we introduce the fairness-cost ratio, calculated as follows:

$$FC(R) = \frac{J(R)}{cost(R)} \tag{7}$$

Different EDC solutions produce different caching fairness degrees at different costs. The FEDC solution is the optimal EDC solution that maximizes the fairness-cost ratio

$$\text{objective: } \max\ FC(R). \tag{8}$$

3.2 Problem Hardness

In this section, we prove the $\mathcal{NP}$-hardness of the FEDC problem by the following theorems.

Theorem 1. *The COP of the FEDC problem is in $\mathcal{NP}$.*

Proof. Based on the above-mentioned constraint, any solution to the FEDC problem can be validated in polynomial time by checking whether the solution satisfies Constraint (6). Thus, the FEDC problem is in $\mathcal{NP}$. □

Theorem 2. *The FEDC problem is $\mathcal{NP}$-hard because it can be reduced from the $\mathcal{NP}$-hard MDS problem.*

Proof. To prove the problem hardness of FEDC, we introduce the minimum dominating set problem (MDS) problem [30], a well-known $\mathcal{NP}$-hard problem. The problem can be defined as: given an undirected graph $G = (V, W)$, where $V(|V| = n)$ is the set of vertices in graph G and $W(|W| = l)$ is the set of edges in graph G, let $c_{n,n}$ denote the matrix to describe the connections between the vertices in V such that $c_{i,j} = 1$ indicates that v_i and v_j are connected. The MDS problem aims to find the subset D of V with the minimum elements such that every vertex not in D is connected to V. It can be formulated as follows:

$$\text{objective:} \min \sum_{i=1}^{n} v_i \tag{9a}$$

$$s.t.: v_i \in \{0, 1\}, i = \{1, \ldots, n\} \tag{9b}$$

$$\sum_{j=1}^{n} c_{i,j} \ge 1, \forall i \in \{1, \ldots, n\} \tag{9c}$$

Now, let us first obtain a special case of the FEDC problem through the following steps: 1) setting the latency constraint $h = 0$, i.e., all the mobile users must be able to retrieve data from nearby edge servers covering them; 2) letting each mobile user be covered by at least two edge servers, i.e., all the mobile users are located in a coverage intersection. Since all the mobile users must be covered in the FEDC problem, the caching benefit produced for a mobile user is always 1. In this way, Eq. (4) can be ignored. Here, we prove that this special case of the FEDC problem is reducible from the MDS problem. Objective (7) of this special FEDC problem can be equally converted to $\min cost(R)$, because $q_{u_k} = 1, \forall u_k \in U$ and $J(R)$ is fixed at 1. Given an instance MDS(V, W, $c_{n,n}$), we can construct an instance FEDC(S, E, Q) in polynomial time where $|V| = |S|$, $|W| = |E|$ and Q is the data benefit

matrix converted from Eq. (3). Any solution that achieves (9a) while fulfilling (9b) also fulfills the constraint (1). Moreover, (9c) requires that every vertex not in D must be connected to at least one element in D. This is equal to Constraint (6) that all the mobile users must be covered. In conclusion, any strategy R will satisfy the special case of the FEDC problem if R satisfies the MDS problem. Thus, the FEDC problem can be reduced from the well-known MDS problem and it is a $\mathcal{NP}$-hard problem. □

4 EDC Strategy Formulation

To find optimal FEDC solutions, in this section, we propose an approach named FEDC-OPT that formulates the EDC strategy that achieves the highest fairness-cost ratio of all based on integer programming. As proved in Section 3.2, FEDC problem is $\mathcal{NP}$-hard. Thus, FEDC-OPT is only practical in small-scale EDC scenarios. To solve the FEDC problems in large-scale scenarios, we propose an approximation approach called FEDC-APX to approximate the optimal FEDC solutions effectively and efficiently.

4.1 Optimal Approach

To solve the FEDC problem optimally, we model it as a constrained optimization problem (COP) that includes of a finite set of decision variables $R = \{r_1, \ldots, r_n\}$, a set of domains $D = \{D_1, \ldots, D_n\}$ that list the values for r_i $(1 \leq i \leq n)$, a set of constraints $\mathcal{C}$ over R, and an optimization objective obj. We can a assign value to each $r_i \in R$ from its domain as a solution of the COP. The optimal solution is the assignment that achieves the optimization objective obj while fulfilling all the constraints in $\mathcal{C}$.

Denote graph $G = (S, E)$ represent the edge server network constituted by the edge servers in the EDC scenario, where $S = \{s_1, \ldots, s_n\}$ is the set of nodes in G and $E = \{e_1, \ldots, e_l\}$ is the set of edges in graph G. We employ n decision variables $R = \{r_1, \ldots, r_n\}$, $r_i = 1$ $(1 \leq i \leq n)$ to denote that the data is cached on edge server $s_i \in S$ and $r_i = 0$ otherwise. Thus, the COP model for the FEDC problem can be presented as

$$\max \frac{J(R)}{cost(R)} \tag{10}$$

$$r_i \in \{0, 1\}, \forall i \in \{1, \ldots, n\} \tag{11}$$

$$q_{u_k} = \max\{r_i * q_{i,j} | u_k \in U_j\},$$
$$\forall i, j \in \{1, \ldots, n\}, \forall k \in \{1, \ldots, m\} \tag{12}$$

$$1 \leq q_{u_k} \leq h + 1, k \in \{1, \ldots, m\} \tag{13}$$

Objective (10) maximizes the fairness-cost ratio $FC(R)$ achieved by caching strategy R. Constraint (11) defines the domains of r_i, indicating whether the data is cached on edge server s_i. Constraint (12) is converted from (4) to ensure that individual mobile users can access their nearest edge servers and retrieve cached data. Constraint (13) ensures the latency constraint - individual mobile users must not retrieve the cached data from edge servers more than h hops away over the edge server network.

The above-mentioned COP model can be solved with an integer programming package, e.g., IBM's CPLEX, Gurobi, etc. The solution is the EDC strategy that maximizes the fairness-cost ratio (10) while fulfilling all the constraints (11), (12), and (13). This optimal is referred to as FEDC-OPT.

4.2 Approximation Approach

As proved in Section 3.2, the FEDC problem is $\mathcal{NP}$-hard. Finding the optimal solutions in large-scale FEDC scenarios can be very difficult. Thus, FEDC-OPT is only suitable in small-scale EDC scenarios. To find solutions to large-scale FEDC problems effectively and efficiently, an approximation algorithm called FEDC-APX is proposed in this paper to find sub-optimal FEDC solutions efficiently.

Intuitively, an approximate approach to the FEDC problem should employ a heuristic to maximize the fairness-cost ratio. However, in certain scenarios, pursuing a maximum fairness-cost ratio solely may end up with an EDC strategy that achieves a high fairness-cost ratio produced by equally low caching benefits for individual users. Moreover, since all the mobile users must be covered, the EDC strategy pursuing a maximum fairness-cost may cause high caching costs to achieve the user coverage constraint. This will be experimentally validated in Section 5.2. To overcome this shortcoming, FEDC-APX employs heuristics to consider both fairness-cost ratio, benefit-cost ratio and cost itself. The pseudo code is presented in Algorithm 1.

FEDC-APX starts with the input and initialization (Lines 1-3). It takes an undirected graph $G(S, E)$, the set of mobile users U, and the caching benefit matrix Q calculated based on Eq. (3) as input. FEDC-APX will employ two candidate strategies, denoted by R_{FC} and R_{QC}, respectively. $J(R_{FC})$ and $J(R_{QC})$ are the fairness degrees achieved by strategy R_{FC} and R_{QC}, respectively, calculated with Eq. (5). $FC(R_{FC})$ and $FC(R_{QC})$ are the fairness-cost ratios achieved by R_{FC} and R_{QC}.

To formulate the first candidate strategy R_{FC}, Algorithm 1 calculates the fairness-cost ratio achieved by selecting each edge server $s_i \in S$ based on all mobile users' caching benefits with the equation $\frac{J(R_{FC} \cup s_i)}{cost(R_{FC} \cup s_i)}$ (Line 5). Then, it selects the edge server s_i among S that achieves the highest fairness-cost ratio, and includes it in the candidate strategy R_{FC} (Line 6). The fairness-cost ratio achieved by R_{FC} is saved into $FC(R_{FC})$ (line 7). Next, the algorithm removes the selected edge server s_i from S (Lines 8). The above progress iterates until all mobile users can access a data replica (Line 9). When the algorithm obtains the first candidate strategy R_{FC}, it resets the list of edge servers S to its original state (Line 10). Next, FEDC-APX formulates the second candidate strategy R_{QC} (Lines 11-16). It calculates and selects the edge server s_i among S that achieves the highest benefit-cost ratio (Line 12), and includes it in the candidate strategy R_{QC} (Line 13). The above progress iterates until all mobile users can access a data replica under the latency constraint (Line 16). After that, the third candidate strategy R_C is formulated (Lines 19-23). It calculates and selects the edge server s_i among S that covers the most uncovered users (Lines 19) and includes it in the candidate strategy R_C (Line 20), until all mobile users can access a data replica under the latency constraint (Line 23). In the end, among R_{FC}, R_{QC} and R_C, FEDC-APX returns the one that achieves the highest fairness-cost ratio as the final strategy R (Lines 24-25).

4.2.1 Complexity and Approximation Ratio Analysis

In Algorithm 1, the computational complexities of the iterations in Lines 4-9, Lines 11-16, and Lines 18-23 are $O(nm)$. Thus, the computational complexity of FEDC-APX is also $O(nm)$. This indicates that FEDC-APX can complete in polynomial time and is suitable for solving large-scale FEDC problems.

Algorithm 1. FEDC-APX

Input: Graph $G(S, E)$, U, Q
Output: caching strategy R, fairness-cost ratio $FC(R)$
1: Initialization:
2: $R_{FC}, R_{QC}, R_C \leftarrow \emptyset$
$J(R_{FC}), J(R_{QC}), J(R_C) = 0$
$FC(R_{FC}), FC(R_{QC}), FC(R_C) = 0$
3: End of initialization
4: **repeat**
5: Calculate fairness-cost ratio by $\frac{J(R_{FC} \cup s_i)}{cost(R_{FC} \cup s_i)}$ for each edge server $s_i \in S$
6: Select s_i with the highest value of the fairness-cost ratio $\max\{\frac{J(R_{FC} \cup s_i)}{cost(R_{FC} \cup s_i)} \mid s_i \in S\}$, then update $R_{FC} \leftarrow R_{FC} \bigcup s_i$
7: Update $FC(R_{FC}) = \frac{J(R_{FC})}{cost(R_{FC})}$
8: $S \leftarrow S - \{s_i\}$
9: **until** All users in U can access data replica
10: Reset S to original state
11: **repeat**
12: Calculate benefit-cost ratio by $\frac{Q(R_{QC} \cup s_i)}{cost(R_{QC} \cup s_i)}$ for each edge server $s_i \in S$
13: Select s_i that produces the highest value of the benefit-cost ratio, then update $R_{QC} \leftarrow R_{QC} \bigcup s_i$
14: Update $FC(R_{QC}) = \frac{J(R_{QC})}{cost(R_{QC})}$
15: $S \leftarrow S - \{s_i\}$
16: **until** All users in U can access data replica
17: Reset S to original state
18: **repeat**
19: Calculate the number of uncovered mobile users for each edge server $s_i \in S$
20: Select s_i that covers the most uncovered users, then update $R_C \leftarrow R_C \bigcup s_i$
21: Update $FC(R_C) = \frac{J(R_C)}{cost(R_C)}$
22: $S \leftarrow S - \{s_i\}$
23: **until** All users in U can access data replica
24: $R = \arg\max_{R \in \{R_{FC}, R_{QC}, R_C\}} FC(R)$
25: **return** R

Next, we prove FEDC-APX's approximation ratio, i.e., the ratio of the overall fairness achieved by FEDC-APX over that achieved by FEDC-OPT in the worst-case scenario.

Theorem 3. *FEDC-APX achieves an approximation ratio of* $\frac{Y_0^2}{lnX+1}$*, where* X *denotes the maximum number of mobile users that can be covered by any* $s_i \in S$ *within* h *hops, and* Y_0 *denotes the percentage of mobile users that can retrieve data from edge servers within zero hop according to* R*.*

Proof. As discussed above, Algorithm 1 consists of three main components, i.e., Lines 4-9, Lines 11-16 and Lines 18-23, producing three candidate strategies, i.e., R_{FC}, R_{QC} and R_C, respectively. As the algorithm outputs the best one among them, inspired by the study [31], we can analyze the approximation ratio of FEDC-APX by R_C. Now we employ an amortized strategy to analyze FEDC-APX's approximation ratio. We assign a unit caching cost to each new mobile user covered by caching a new data replica. According to constraint (6), the final solution must ensure that all mobile users are covered. Let R_{OPT} denote the optimal solution found by FEDC-OPT, R_{APX} denote the approximate solution found by FEDC-APX, and $newU(s_i)$ denote the set of new users covered by including s_i into R_{OPT}, i.e., caching a data replica on s_i. Due to the heuristic employed by FEDC-APX, this cost is at most $\frac{1}{|newU(s_i)|}$ per such new user. If it exceeds $\frac{1}{|newU(s_i)|}$, FEDC-APX will cache the data replica on s_i because it covers at least $|newU(s_i)|$ mobile users. In the worst case, no two users are covered by the same data replica. In such cases, the total cost of covering $newU(s_i)$ is at most $\frac{1}{|newU(s_i)|} + \frac{1}{|newU(s_i)|-1} + \ldots + \frac{1}{2} + \frac{1}{1}$. Since X is the maximum number of mobile users that can be covered by any $s_i \in S$ within h hops, there is $|newU(s_i)| \leq X$. Thus, $\forall s_i \in S$, the cost of covering $newU(s_i)$ by FEDC-APX is at most

$$\frac{1}{X} + \frac{1}{X-1} \ldots + \frac{1}{2} + \frac{1}{1} \leq \int_1^X \frac{dx}{x} + 1 \leq lnX + 1. \tag{14}$$

Let R^* denote the solution with the minimum cost to the FEDC problem, the total cost produced by FEDC-APX is at most $|R^*| \cdot (lnX + 1)$. Since the cost incurred by the optimal solution R_{OPT} to the FEDC problem is always lower than or equal to $|R^*|$, the caching cost incurred by R_{APX} is at most

$$\begin{aligned} cost(R_{APX}) &= |R_{APX}| \leq |R^*| * (lnX + 1) \\ &\leq |R_{OPT}| * (lnX + 1) \\ &= cost(R_{OPT}) * (lnX + 1) \end{aligned} \tag{15}$$

Let $Y_0, Y_1, \ldots, Y_h$ denote the percentages of users that can access data via $0, 1, \ldots h$ hops, respectively. The overall fairness degree achieved by R_{APX} is at least

$$\begin{aligned} J(R_{APX}) &\geq \frac{m^2[Y_0 * (h+1) + Y_1 * h + \ldots + Y_h * 1]^2}{m^2[Y_0 * (h+1)^2 + Y_1 * h^2 + \ldots + Y_h * 1^2]} \\ &\geq \frac{[Y_0 * (h+1) + (1 - Y_0) * 1]^2}{Y_0 * (h+1)^2 + (1 - Y_0) * h^2} \end{aligned} \tag{16}$$

Because of $0 \leq Y_0 \leq 1$, we can obtain $Y_0^3 \leq Y_0^2 \leq Y_0 \leq 1$. Thus, the lower bound of $J(R_{APX})$ is

$$\begin{aligned} J(R_{APX}) &\geq \frac{Y_0^2 * h^2 + 2Y_0 * h + 1}{h^2 + 2Y_0 * h + Y_0} \\ &\geq \frac{Y_0^2 * h^2 + 2Y_0^3 * h + Y_0^3}{h^2 + 2Y_0 * h + Y_0} \\ &= Y_0^2 * \frac{h^2 + 2Y_0 * h + Y_0}{h^2 + 2Y_0 * h + Y_0} = Y_0^2 \end{aligned} \tag{17}$$

Since the highest fairness degree achieved by any EDC strategies, including R_{OPT}, is at most 1, the approximation ratio of FEDC-APX is

TABLE 2
Experiment Settings

	N	$density$	M	h
Set #1	10, 15, 20, 25, 30	1	100	1
Set #2	30	1.0, 1.5, 2.0, 2.5, 3.0	100	1
Set #3	30	1	50, 100, 150, 200, 250	1
Set #4	30	1	100	1, 2, 3

$$ratio = \frac{\frac{J(R_{APX})}{cost(R_{APX})}}{\frac{J(R_{OPT})}{cost(R_{OPT})}} \geq \frac{J(R_{APX})}{J(R_{OPT})} * \frac{cost(R_{OPT})}{cost(R_{APX})} \geq \frac{Y_0^2}{lnX + 1} \quad (18)$$

□

5 Experimental Evaluation

This section evaluates the proposed approaches experimentally. All experiment sets are conducted on a Windows 10 machine equipped with an Intel Core i7-7300 processor (8 CPUs, 2.4 GHz) and 8 GB RAM. All the approaches are implemented in Java and FEDC-OPT solves the FEDC problem with IBM CPLEX Optimizer.

5.1 Experiment Setup

Data Set. Four sets of experiments, i.e., denoted as Sets #1 - #4, are conducted on the real-world EUA data set [32]. This data set has been used widely in the field of MEC and is available at https://github.com/swinedge/eua-dataset. It consists with the location information of real-world base stations and anonymous users in Melbourne CBD in Australia.

Competing Approaches. The performance of FEDC-OPT and FEDC-APX is compared against 4 representative approaches, including a state-of-the-art approach:

- *Random*: This approach caches data replicas on edge servers randomly until all mobile users are covered.
- *Greedy-Connection (GC)*: This greedy approach sorts edge servers by their degrees, i.e., the number of edge servers connected to them. Then, it always caches data replicas on the edge servers with the highest degrees, until the user coverage constraint is fulfilled.
- *Greedy-Covered-User (GCU)*: This approach first sorts edge servers by the number of mobile users covered by them. It then caches data replicas on the edge servers covering the most mobile users, until all mobile users are covered.
- *FEDC-APX-FC*: This approach is a variant of FEDC-APX. Instead of selecting the better one between R_{QC} and R_{FC}, it always returns R_{FC} as the final solution. Briefly speaking, it heuristically caches data replicas on edge servers to pursue the maximum fairness-cost ratio.
- *Benefit-Oriented-Optimal (BOA-IP)* [33]: This approach assumes that a mobile user can only access data from edge servers covering the user and pursues to maximize the overall data caching benefit calculated with Eq. (3). It models the optimization problem as an integer problem and solves it with IBM CPLEX Optimizer, similar to FEDC-OPT.
- *ST-MEDC* [24]: This state-of-the-art approach heuristically selects the edge server that provides the highest caching benefit or highest benefit-cost ratio, until the user coverage constraint is fulfilled.

Setting Parameters. In the experiments, the values of the following three parameters are varied to simulate a wide range of different EDC scenarios:

- Number of edge servers (n): This parameter determines the size of graph G that represents the edge server network constituted by the edge servers. It varies from 10 to 30 in steps of 5 in the experiments unless fixed.
- Network density (d): This parameter determines the density of graph G. In graph G, the number of nodes (n) and the number of edges (e) in G fulfills: $d = e/n$ and varies from 1 to 3 in steps of 0.5 in the experiments unless fixed.
- Number of mobile users (m): This parameter indicates the number of users to be covered and varies from 50 to 250 in steps of 50 in the experiments unless fixed.
- Latency Constraint (h): This parameter determines the maximum number of hops via which a mobile user can retrieve data from an edge server over the edge server network. It varies from 1 to 3 in steps of 1 in the experiments unless fixed.

Table 2 summarizes the parameter settings in the experiments. Each experiment is repeated for 100 times and the results are averaged. In all experiment sets, Considering the server capacity constraint mentioned in Section 1, we assume that each edge server has 10% chance have insufficient capacity to cache data replica.

Performance Metrics. We use three metrics to evaluate the performance of the approaches, two for effectiveness and one for efficiency.

For effectiveness evaluation, we compare the approaches in the following five metrics.

- Fairness degree, ranging from 0 to 1, calculated with Eq. (5), the higher the better;
- Fairness-cost ratio, ranging from 0 to 1, calculated with Eq. (7), the higher the better
- Caching benefit, ranging from 1 to h + 1, calculated with Eq. (4), the higher the better;
- Benefit-cost ratio, ranging from 0 to 1, the higher the better;
- Caching cost, ranging from 0 to N, calculated with Eq. (2), the lower the better.

For efficiency evaluation, we compare their computation time.

- Computation time, the time taken to find a solution, the lower the better.

TABLE 3
Effectiveness Comparison (Fairness-Cost Ratio, Fairness Degree, Benefit, Benefit-Cost Ratio, Caching Cost)

	Set #1					Set #2				
	fairness/cost	*fairness*	*benefit*	*benefit/cost*	*cost*	*fairness/cost*	*fairness*	*benefit*	*benefit/cost*	*cost*
FEDC-OPT	0.1202	0.9341	1.5700	0.2007	7.4800	0.1722	0.9142	1.3223	0.2440	5.4956
FEDC-APX	0.1071	0.9497	1.5000	0.1719	8.7000	0.1550	0.9048	1.3515	0.2280	6.0717
GC	0.0734	0.9529	1.6200	0.1196	13.8000	0.0788	0.9334	1.6040	0.1220	12.4012
GCU	0.0712	0.9797	1.9100	0.1232	13.8475	0.0733	0.9569	1.8035	0.1182	15.8500
ST-MEDC	0.0642	0.9833	1.9800	0.1256	15.3000	0.0704	0.9827	1.8604	0.1350	15.1719
FEDC-APX-FC	0.0438	0.9828	1.9400	0.0839	20.6000	0.0564	0.9455	1.6253	0.0951	17.9989
BOA-IP	0.0709	1.0000	2.0000	0.1178	16.9000	0.0551	1.0000	2.0000	0.1091	17.3696
Random	0.0616	0.9482	1.8000	0.0916	17.6000	0.0630	0.9429	1.6968	0.1070	16.4006

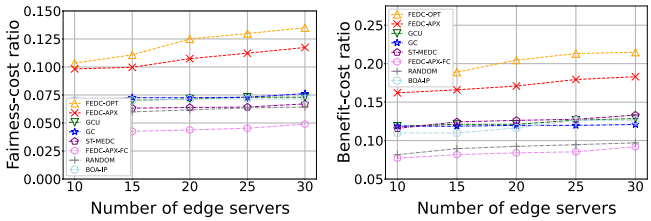
	Set #3					Set #4				
	fairness/cost	*fairness*	*benefit*	*benefit/cost*	*cost*	*fairness/cost*	*fairness*	*benefit*	*benefit/cost*	*cost*
FEDC-OPT	0.1167	0.9126	1.5601	0.1966	8.1292	0.2115	0.9174	2.4470	0.6003	5.2179
FEDC-APX	0.1080	0.9027	1.4960	0.1764	8.5763	0.1073	0.8122	2.2864	0.2736	8.3861
GC	0.0613	0.9370	1.7076	0.1076	16.0643	0.0576	0.8889	2.4148	0.1518	15.9556
GCU	0.0511	0.9783	1.9027	0.0968	19.9575	0.0500	0.9622	2.7921	0.1397	20.0296
ST-MEDC	0.0602	0.9925	1.9619	0.1208	16.8957	0.0616	0.9905	2.9637	0.1799	16.3299
FEDC-APX-FC	0.0470	0.9754	1.8854	0.0921	20.6962	0.0489	0.9974	2.9306	0.1439	20.7525
BOA-IP	0.0535	1.0000	2.0000	0.1075	19.0609	0.0554	1.0000	3.0000	0.1631	18.3923
Random	0.0468	0.9636	1.8356	0.0876	20.9698	0.0443	0.9445	2.7039	0.1218	22.2622

5.2 Experimental Results

Table 3 summarizes and compares the effectiveness of the six approaches. Figs. 2, 3, 4, and 5 illustrate the benefit-cost ratios and fairness-cost ratios achieved by the approaches, as well as their computation time.

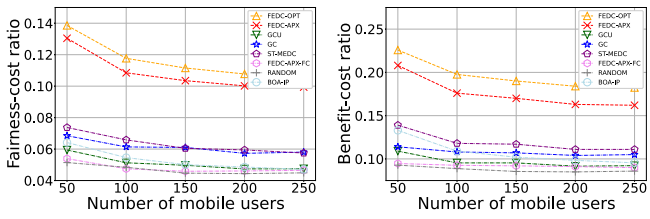
5.2.1 *Effectiveness*

Table 3 summarizes the experimental results in the four sets of experiments, where the winner is highlighted in grey and the runner up in light grey in each column. It shows that FEDC-OPT and FEDC-APX can always achieve the highest fairness-cost ratio, the highest benefit-cost ratio, and the lowest caching cost. Their advantages in these three metrics are remarkable. Take FEDC-OPT for example. It outperforms the GC, GCU, ST-MEDC, FEDC-APX-FC, BOA-IP, and Random in the fairness-cost ratio by an average of 126.13%, 152.68%, 142.04%, 216.47%, 166.50%, and 187.71% in Sets #1, #2, #3 and #4, respectively. FEDC-APX's performance is very close to FDEC-OPT's, with a 104.12% gap on average in fairness-cost ratio across six sets of experiments.



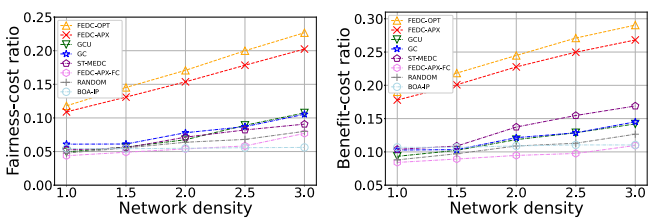
(a) fairness-cost ratio vs. n (b) benefit-cost ratio vs. n

Fig. 2. Effectiveness versus number of edge servers (n) in Set #1.



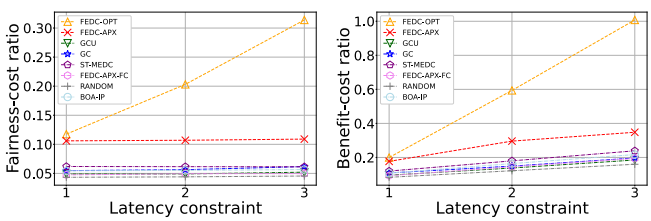
(a) fairness-cost ratio vs. m (b) benefit-cost ratio vs. m

Fig. 4. Effectiveness versus number of users (m) in Set #3.



(a) fairness-cost ratio vs. d (b) benefit-cost ratio vs. d

Fig. 3. Effectiveness versus network density (d) in Set #2.



(a) fairness-cost ratio vs. h (b) benefit-cost ratio vs. h

Fig. 5. Effectiveness versus latency constraint (h) in Set #4.

This indicates the high cost-effectiveness of the caching strategies formulated by FEDC-OPT and FEDC-APX. It is worth mentioning that FEDC-APX-FC's performance is much lower than FEDC-APX as shown in Table 3. The reason is that FEDC-APX-FC only focuses on selecting the edge server that can provide the highest fairness-cost ratio. In each of its iteration, it pursues a high fairness-cost ratio without considering the caching benefit. In the worst-case scenario, the caching benefits produced for individual mobile users are equally low. In the meantime, it needs to cache data replicas on edge servers continuously to fulfill the user coverage constraint. This drives FEDC-APX-FC to cache more data replicas than FEDC-APX, which increases its data caching cost and consequently reduces its fairness-cost ratio and the benefit-cost ratio. Its comparison against FEDC-APX validate the design of FEDC-APX introduced in Section 4.2.

In the *fairness* columns, we can see that BOA-IP and ST-MEDC always achieve the highest and the second highest fairness degrees. In particular, BOA-IP can always achieve the maximum fairness degree of 1.0. As discussed in Section 2, high fairness may be caused by mobile users' equally low caching benefits. As discussed above in Section 5.1, BOA-IP pursues to maximize the overall caching benefit at the minimum caching cost, assuming that mobile users can only access data from edge servers covering them. Thus, BOA-IP will ensure that all the mobile users can access data via 0 hop over the edge server network. This will yield the same and highest caching benefit for every individual mobile user, as can be confirmed by BOA-IP's performance demonstrated in the *benefit* columns. Thus, BOA-IP achieves high caching fairness by achieving equally high caching benefits for mobile users. Compared with BOA-IP or any other three approaches, FEDC-OPT and FEDC-APX do not have an advantage in maximizing fairness or caching benefit because they are designed to pursue cost-effectiveness, considering not just caching fairness but also caching costs. As discussed in the last paragraph, the *fairness/cost* columns indicate FEDC-OPT and FEDC-APX's performance in maximizing the fairness-cost ratio. Now let us take a look at the *benefit/cost* columns in Table 3. FEDC-OPT and FEDC-APX are the clear winners in the competition for the maximum benefit-cost ratio. Across all four sets of experiments, they outperform GC, GCU, ST-MEDC, FEDC-APX-FC, Random, and BOA-IP by an average of 163.65% and 80.46%, respectively. The key to FEDC-OPT and FEDC-APX's profound advantages in maximizing the fairness-cost ratio and the benefit-cost ratio is their ability to achieve high caching fairness and benefits at low caching costs. This is evidenced by the *cost* columns. We can see that the costs incurred by the caching strategies formulated by FEDC-OPT and FEDC-APX are significantly lower than those formulated by the other four approaches. Specifically, FEDC-OPT's caching cost is 49.29%, 57.63%, 56.66%, 62.92%, 58.84% and 61.77% lower than GC, GCU, ST-MEDC, FEDC-APX-FC, BOA-IP, and Random, respectively. FEDC-APX incurs slightly higher caching costs than FEDC-OPT, but still outperforms GC, GCU, ST-MEDC, FEDC-APX-FC, BOA-IP and Random considerably by 45.15%, 54.17%, 53.12%, 59.89%, 55.48% and 58.65%, respectively. Overall, the results indicate the importance of trading off between caching fairness and caching costs.

Fig. 2 compares the performance of the six approaches in Set #1 and demonstrates the impact of the number of edge servers (n) which increases from 10 to 30 in steps of 5. Unsurprisingly, FEDC-OPT and FEDC-APX achieve the highest fairness-cost ratios, as shown in Fig. 2a, and benefit-cost ratios, as shown in Fig. 2b. The reasons have been discussed above based on Table 3 and thus are not repeated here. Fig. 2a shows that there is an increase in the fairness-cost ratios achieved by all the six approaches when n increases, some more significant than the others. FEDC-OPT achieves a 34.46% increase in fairness-cost ratio from 0.1004 to 0.1350, and a 15.75% increase in benefit-cost ratio from 0.1848 to 0.2139. FEDC-APX's performance increase is less significant than FEDC-OPT's, i.e., by 19.55% in fairness-cost ratio and 12.89% in benefit-cost ratio. With more edge servers available in the same area, individual users are covered by more edge servers. This expands the solution space where FEDC-OPT and FEDC-APX search for caching strategies. Thus, they are more likely to be able to find caching strategies that yield high fairness-cost ratios. In the meantime, a larger solution space makes it easier for FEDC-OPT and FEDC-APX to find edge servers that cover a lot of edge servers. Caching data replicas on these edge servers increases the overall caching benefit and reduces the overall caching cost. Consequently, their benefit-cost ratios increase with n, as shown in Fig. 2b.

Fig. 3 demonstrates the impact of the network density d in Set # 2. Figs. 3 ab and 3b show d impacts the fairness-cost ratio and benefit-cost ratio in a similar manner, and its impacts on FEDC-OPT and FEDC-APX are more significantly than n. As shown in Fig. 3a, when d increases from 1.0 to 3.0, the fairness-cost ratios achieved by FEDC-OPT and FEDC-APX increase by 92.12% from 0.1180 to 0.2267 and by 87.61% from 0.1090 to 0.2045, respectively. Similar phenomena can be observed in Fig. 3b about the benefit-cost ratio. A higher network density means that the edge servers are connected to more other edge servers on average, i.e., individual edge servers have more neighbour edge servers. This allows FEDC-OPT and FEDC-APX to serve more mobile users with low data retrieval latency without having to cache more data replicas. Accordingly, their fairness-cost ratios and benefit-cost rations both increase with d, as shown in Figs. 3a and 3b, respectively. The results observed in this set of experiments demonstrate that network density also impacts caching performance over the edge server network profoundly, in addition to its capacity, i.e., the number of edge servers.

Fig. 4 shows the performance of FEDC-OPT and FEDC-APX when the EDC scenario scales up in the number of mobile users (m) in Set #3. Similar to n in Set #1 and d in Set #2, the increase in m in the fairness-cost ratio and the benefit-cost ratio both in the same way. From Fig. 4a we can see that when m increasing from 50 to 250, FEDC-OPT and FEDC-APX both suffer a performance drop in maximizing the fairness-cost ratio, by 22.29% from 0.1386 to 0.1077 for the former and 23.16% from 0.1304 to 0.1002 for the latter. The reason is that, given a fixed number of 30 edge servers, when m increases, FEDC-OPT and FEDC-APX need to cache more data replicas to serve all the mobile users. This increases the caching cost immediately but does not necessarily increase the caching benefit as much. This is

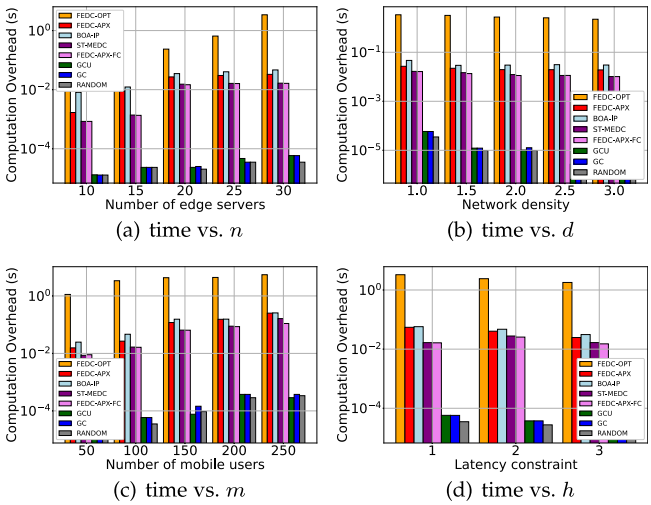


Fig. 6. Efficiency versus parameters.

evidenced by FEDC-OPT's 17.89% drop and FED-APX's 20.67% drop in benefit-cost ratio, as shown in Fig. 4b. The increase in caching cost also reduces their fairness-cost ratios, as shown in Fig. 4a. An interesting observation in Set #3 is that when m exceeds 200, FEDC-OPT and FEDC-APX's performance drops stop. This is because an m larger than 200 does not further spread the mobile users across the edge servers' coverage areas. As a result, FEDC-OPT and FEDC-APX do not need to cache more data replicas to serve the extra mobile users.

Fig. 5 shows the performance of the six approaches under different latency constraints (h) in Set #4. FEDC-OPT is the clear winner in this set of experiments. FEDC-APX seconds to FEDC-OPT and outperforms the other approaches significantly. When the latency constraint is relaxed with a larger h, mobile users can retrieve data replicas from edge servers further away over the edge server network. Also, an edge server can serve more users via h-hops. In this way, all the mobile users can be served by edge servers with fewer data replicas, and the data hit ratio can be guaranteed as well. In the meantime, according to Eq. (3), a larger h will lead to higher caching benefits when users are served. Taking both advantages, FEDC-OPT manages to achieve much higher fairness-cost ratios and benefit-cost ratios when h increases, as shown in Figs. 5a and 5b, respectively. FEDC-APX also achieves a higher benefit-cost ratio when the latency constraint is relaxed. However, a larger h increases the differences in the caching benefits produced by serving mobile users with a low data retrieval latency and those with a high data retrieval latency. Thus, according to Algorithm 1, FDEC-APX tends to cache more data replicas to increase the fairness-cost ratio. Thus, it cannot match the ability of FEDC-OPT to maximize the fairness-cost ratio when h increases.

5.2.2 Efficiency

Figs. 6a, 6b, 6c, and 6d demonstrate the computation times taken by the approaches to find a solution in Set #1 - Set #4. We can see that FEDC-OPT pays a high price for its high effectiveness, taking much more time than the other approaches to find a solution. This confirms the $\mathcal{NP}$-hardness of the FEDC problem as analyzed in Section 3.2. Trying to find optimal FEDC solutions, FEDC-OPT suffers from high computational overheads and low scalability, especially in EDC scenarios with a large number of edge servers (n), as shown in Fig. 6a. When n increases from 10 to 30, the computation overhead of FEDC-OPT increases dramatically from 0.0374 seconds to 3.4204 seconds in Set #1, while the other seven approaches' computation times increase only slightly, in particular, from 0.0016 seconds to 0.0329 seconds for FEDC-APX. In Fig. 6b, with the network density increases from 1 to 3, the computation time of FEDC-OPT decreases from 3.3844 seconds to 2.2265 seconds. The computation time of FEDC-APX also decreases, as well as the other six approaches. This confirms our finding above from Fig. 3b - a higher network density makes it easier to find a solution to the FEDC problem. When the number of mobile users n increases from 50 to 250 in Set #3, the computation time of FEDC-OPT increases from 1.1283 seconds to 4.4733 seconds, as shown in Fig. 6c. Such low efficiency makes FEDC-OPT unsuitable in most EDC scenarios because mobile users cannot tolerate such a long delay before they can retrieve the requested data. In experiment Set #4, with the increase in the latency constraint h, the computation time of FEDC-OPT decreases from 3.2844 seconds to 1.7987 seconds, as shown in Fig. 6d. However, this is still not acceptable in most EDC scenarios. Fortunately, FEDC-APX offers a promising solution in the EDC scenarios simulated in Set #3 and Set #4, always taking only 0.01 - 0.3 seconds to formulate an EDC strategy.

Overall, FEDC-OPT is practical only in small-scale EDC scenarios due to its high computational overheads. Its low efficiency outweighs its ability to maximize the fairness-cost ratio and the cost-benefit ratio. In large-scale scenarios, FEDC-APX is a more practical option because it strikes a proper trade-off between effectiveness and efficiency. FEDC-APX's high efficiency also makes it a promising FEDC solution under online settings where data caching demands may vary dynamically over time.

6 Related Work

Mobile edge computing (MEC) is a new paradigm enabled by 5 G/6 G. It is the extension of the mobile cloud computing paradigm by enabling computing and storage resources at the edge of the network [34], [35]. App vendors can cache data on edge servers to enable low-latency data retrieval for their users [36], [37]. The problem of edge data caching has attracted a lot of attention. Due to the proximity constraint, a large body of studies has attempted to improve the cost-effectiveness of edge data caching by maximizing the cache hit rate [38], [39], [40]. Collaborative data caching aims to tackle the challenge of cost-effectiveness in a different way [10], [23]. Specifically, an edge server can retrieve data from its nearby edge servers to serve its users. Gharaibeh et al. [41] leverage the collaboration among edge servers to minimize the total caching cost. They proposed an algorithm that determines how edge servers retrieve and cache data collaboratively to fulfill users' data requests. Tran et al. [42] proposed two approaches for edge video caching specifically, one for allocating data and the other for scheduling user requests. The main objective of their study is to optimize users' QoE by caching different bitrate versions of a video data on edge servers. Most existing studies of edge

data caching aim to achieve edge infrastructure providers' various optimization objectives without considering caching costs. In recent years, researchers are starting to study the edge data caching problems from the app vendor's perspective, taking caching costs into account [17], [24], [33], [37]. However, these approaches solely pursue the optimization of caching performance and do not consider the fairness among users.

Fairness is critical to many app vendors' long-term business goals [20]. It has been considered in studies of resource allocation in various domains, e.g., edge device caching [43], fog device cooperation [44], and wireless network management [19]. Cao et al. [45] try to solve the problem with an auction-based approach that allows an edge infrastructure provider to allocate cache spaces based on users' data demands. In [43], the authors propose an algorithm that combines the fairness caching cost and contention cost and pursues fairness-oriented collaborative data caching across mobile and IoT devices. The authors of [46] propose an approach named QKnober to balance fairness and system efficiency elastically and flexibly during the resource allocation process in the cloud computing environment. The authors of [44] propose an algorithm named the fairness cooperative algorithm (FCA) to solve the joint optimization problem of data peer offloading in the fog computing environment, taking both users' QoE and fog nodes' energy into account. The authors of [47] propose a blockchain-based system and a resource allocation model, considering the fairness degree cost (FDC) for quick and fair resource allocation on mobile and IoT devices. They formulate the problem by combining the fairness factor and the storage efficient factor. In [48], the researchers study the fair resource allocation problem in a system that contains different resource types required by users. The concept of dominant resource fairness proposed in [48] has been widely employed in a large body of research to incentivize users to share resources, such as [22]. In [49], the authors present a cloud-radio access infrastructure during the up-link wireless resource allocation process, which can balance network throughput and allocation fairness. The authors of [50] propose a fairness-based protocol, considering the fair data link allocation among secondary users in 5 G cognitive Radio Ad Hoc Networks. The authors of [51] propose a power allocation scheme that can achieve the optimal Non-Orthogonal Multiple Access (NOMA)-based power allocation while guaranteeing a high level of fairness. The authors of [52] investigate the caching problem in cache-assisted wireless networks. They divide the caching process into a caching phase and a delivery phase, where proportional fairness and min-max fairness are considered, respectively. They formulate an optimization problem for each phase with a corresponding optimal objective, i.e., minimizing users' total weighted latency and minimizing their maximum latency.

Fair edge data caching is fundamentally different from these fairness-aware problems for the unique characteristics of the MEC environment, e.g., proximity constraint, capacity constraint, multi-hop data transmissions, etc. Edge data caching without the consideration of fairness leads to significant differences in users' QoE. For example, as demonstrated in Section 5, the approaches proposed in [33] (implemented as BOA-IP in our experiments) and [24] (implemented as ST-MEDC in our experiments) suffer from low benefit-cost ratios. This will impact users' QoE and jeopardize app vendors' long-term business goals [20]. Thus, app vendors must consider fairness while pursuing high caching benefits with their edge data caching strategies. This new problem is referred to as the fair edge data caching (FEDC) problem, which aims to strike a trade-off between caching fairness, caching benefit and caching cost. To our best knowledge, this study is the first to consider fairness in edge data caching from the app vendor's perspective.

7 Conclusion and Future Work

In this paper, we made the first attempt to study the fair edge data caching (FEDC) problem in the MEC environment from the app vendor's perspective. We modelled the problem and proved its $\mathcal{NP}$-hardness theoretically. To solve this problem, we proposed an optimal approach named FEDC-OPT based on integer programming. Considering the fact that finding optimal solutions to large-scale FEDC problems is intractable, we proposed an approximation approach called FEDC-APX to accommodate large-scale FEDC scenarios with a theoretically-proved approximation ratio. The results of extensive experiments demonstrate that our approaches outperform the baseline approaches and the state-of-the-art approaches significantly. In addition, it also shows that while FEDC-OPT achieves the highest fairness-cost ratio and benefit-cost ratio, FEDC-APX is the better option in large-scale EDC scenarios for its high efficiency.

As demonstrated in Section 5, FEDC-APX is not capable of fully utilizing a relaxed latency constraint like FEDC-OPT. In the future, we will investigate this issue and enhance FEDC-APX to overcome this limitation.

References

[1] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.

[2] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of things (IoT): A vision, architectural elements, and future directions," *Future Gener. Comput. Syst.*, vol. 29, no. 7, pp. 1645–1660, 2013.

[3] N. Wang, B. Varghese, M. Matthaiou, and D. S. Nikolopoulos, "ENORM: A framework for edge node resource management," *IEEE Trans. Serv. Comput.*, vol. 13, no. 6, pp. 1086–1099, Nov./Dec. 2017.

[4] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016.

[5] X. Xia, F. Chen, J. Grundy, M. Abdelrazek, H. Jin, and Q. He, "Constrained app data caching over edge server graphs in edge computing environment," *IEEE Trans. Serv. Comput.*, to be published, doi: 10.1109/TSC.2021.3062017.

[6] Q. He et al., "A game-theoretical approach for user allocation in edge computing environment," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 3, pp. 515–529, Mar. 2019.

[7] S. Ma, S. Guo, K. Wang, W. Jia, and M. Guo, "A cyclic game for service-oriented resource allocation in edge computing," *IEEE Trans. Serv. Comput.*, vol. 13, no. 4, pp. 723–734, Jul./Aug. 2020.

[8] W. Zhang, Y. Hu, Y. Zhang, and D. Raychaudhuri, "SEGUE: Quality of service aware edge cloud service migration," in *Proc. IEEE Int. Conf. Cloud Comput. Technol. Sci.*, 2016, pp. 344–351.

[9] Q. He et al., "A game-theoretical approach for mitigating edge DDoS attack," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 4, pp. 2333–2348, Jul./Aug. 2022.

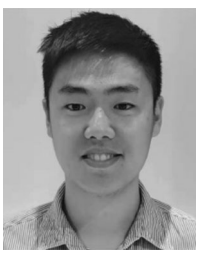
[10] X. Xia, F. Chen, Q. He, J. Grundy, M. Abdelrazek, and H. Jin, "Online collaborative data caching in edge computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 2, pp. 281–294, Feb. 2021.

[11] B. Li, Q. He, L. Yuan, F. Chen, L. Lyu, and Y. Yang, "EdgeWatch: Collaborative investigation of data integrity at the edge based on blockchain," in *Proc. 28th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2022, pp. 3208–3218.
[12] L. Wang, K.-K. Wong, S. Lambotharan, A. Nallanathan, and M. Elkashlan, "Edge caching in dense heterogeneous cellular networks with massive mimo-aided self-backhaul," *IEEE Trans. Wireless Commun.*, vol. 17, no. 9, pp. 6360–6372, 2018.
[13] X. Xia, F. Chen, Q. He, J. C. Grundy, M. Abdelrazek, and H. Jin, "Cost-effective app data distribution in edge computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 1, pp. 31–44, Jan. 2021.
[14] B. Li, Q. He, F. Chen, H. Jin, Y. Xiang, and Y. Yang, "Auditing cache data integrity in the edge computing environment," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 5, pp. 1210–1223, May 2021.
[15] X. Zhang and Q. Zhu, "Hierarchical caching for statistical QoS guaranteed multimedia transmissions over 5G edge computing mobile wireless networks," *IEEE Wireless Commun.*, vol. 25, no. 3, pp. 12–20, Jun. 2018.
[16] Y. Ye, Z. Zhang, G. Yang, and M. Xiao, "Minimum cost based clustering scheme for cooperative wireless caching network with heterogeneous file preference," in *Proc. IEEE Int. Conf. Commun.*, 2017, pp. 1–6.
[17] X. Xia et al., "Graph-based optimal data caching in edge computing," in *Proc. Int. Conf. Serv.-Oriented Comput.*, 2019, pp. 477–493.
[18] X. Hou and S. Dey, "Motion prediction and pre-rendering at the edge to enable ultra-low latency mobile 6DoF experiences," *IEEE Open J. Commun. Soc.*, vol. 1, pp. 1674–1690, 2020.
[19] S. Huaizhou, R. V. Prasad, E. Onur, and I. G. M. M. Niemegeers, "Fairness in wireless networks: Issues, measures and challenges," *IEEE Commun. Surv. Tut.*, vol. 16, no. 1, pp. 5–24, Jan.–Mar. 2013.
[20] T.-H. Ho and X. Su, "Peer-induced fairness in games," *Amer. Econ. Rev.*, vol. 99, no. 5, pp. 2022–2049, 2009.
[21] A. A. Barakabitze et al., "QoE management of multimedia streaming services in future networks: A tutorial and survey," *IEEE Commun. Surv. Tut.*, vol. 22, no. 1, pp. 526–565, Jan.–Mar. 2019.
[22] C. Joe-Wong, S. Sen, T. Lan, and M. Chiang, "Multiresource allocation: Fairness–efficiency tradeoffs in a unifying framework," *IEEE/ACM Trans. Netw.*, vol. 21, no. 6, pp. 1785–1798, Dec. 2013.
[23] Z. Xu, L. Zhou, S. C.-K. Chau, W. Liang, Q. Xia, and P. Zhou, "Collaborate or separate? distributed service caching in mobile edge clouds," in *Proc. IEEE Conf. Comput. Commun.*, 2020, pp. 2066–2075.
[24] X. Xia et al., "OL-MEDC: An online approach for cost-effective data caching in mobile edge computing systems," *IEEE Trans. Mobile Comput.*, to be published, doi: 10.1109/TMC.2021.3107918.
[25] J. L. D. Neto, S.-Y. Yu, D. F. Macedo, J. M. S. Nogueira, R. Langar, and S. Secci, "ULOOF: A user level online offloading framework for mobile edge computing," *IEEE Trans. Mobile Comput.*, vol. 17, no. 11, pp. 2660–2674, Nov. 2018.
[26] L. Chen, S. Zhou, and J. Xu, "Computation peer offloading for energy-constrained mobile edge computing in small-cell networks," *IEEE/ACM Trans. Netw.*, vol. 26, no. 4, pp. 1619–1632, Aug. 2018.
[27] B. Li et al., "READ: Robustness-oriented edge application deployment in edge computing environment," *IEEE Trans. Serv. Comput.*, vol. 15, no. 3, pp. 1746–1759, May/Jun. 2022.
[28] R. K. Jain et al., *A Quantitative Measure of Fairness and Discrimination*, Hudson, MA, USA: Eastern Research Laboratory, Digital Equipment Corporation, 1984.
[29] P. Lai et al., "Edge user allocation with dynamic quality of service," in *Proc. Int. Conf. Serv.-Oriented Comput.*, 2019, pp. 86–101.
[30] D. L. Grinstead and P. J. Slater, "On minimum dominating sets with minimum intersection," *Discrete Math.*, vol. 86, no. 1/3, pp. 239–254, 1990.
[31] F. Chen, J. Zhou, X. Xia, H. Jin, and Q. He, "Optimal application deployment in mobile edge computing environment," in *Proc. IEEE 13th Int. Conf. Cloud Comput.*, 2020, pp. 184–192.
[32] P. Lai et al., "Optimal edge user allocation in edge computing with variable sized vector bin packing," in *Proc. Int. Conf. Serv.-Oriented Comput.*, 2018, pp. 230–245.
[33] Y. Liu, Q. He, D. Zheng, X. Xia, F. Chen, and B. Zhang, "Data caching optimization in the edge computing environment," *IEEE Trans. Serv. Comput.*, vol. 15, no. 4, pp. 2074–2085, Jul./Aug. 2022.
[34] Q. He, Z. Dong, F. Chen, S. Deng, W. Liang, and Y. Yang, "Pyramid: Enabling hierarchical neural networks with edge computing," in *Proc. Web Conf.*, 2022, pp. 1860–1870.
[35] L. Yuan et al., "CoopEdge: A decentralized blockchain-based platform for cooperative edge computing," in *Proc. Web Conf.*, 2021, pp. 2245–2257.
[36] X. Xia et al., "Data, user and power allocations for caching in multi-access edge computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 5, pp. 1144–1155, May 2022.
[37] X. Xia et al., "Budgeted data caching based on k-median in mobile edge computing," in *Proc. 27th IEEE Int. Conf. Web Serv.*, 2020, pp. 197–206.
[38] T. Peng et al., "Value-aware cache replacement in edge networks for Internet of Things," *Trans. Emerg. Telecommun. Technol.*, vol. 32, no. 9, 2021, Art. no. e4261.
[39] Y. Fu, Q. Yu, A. K. Y. Wong, Z. Shi, H. Wang, and T. Q. S. Quek, "Exploiting coding and recommendation to improve cache efficiency of reliability-aware wireless edge caching networks," *IEEE Trans. Wireless Commun.*, vol. 20, no. 11, pp. 7243–7256, Nov. 2021.
[40] Q. Li, Y. Zhang, A. Pandharipande, Y. Xiao, and X. Ge, "Edge caching in wireless infostation networks: Deployment and cache content placement," in *Proc. IEEE Conf. Comput. Commun. Workshops*, 2019, pp. 1–6.
[41] A. Gharaibeh, A. Khreishah, B. Ji, and M. Ayyash, "A provably efficient online collaborative caching algorithm for multicell-coordinated systems," *IEEE Trans. Mobile Comput.*, vol. 15, no. 8, pp. 1863–1876, Aug. 2015.
[42] T. X. Tran and D. Pompili, "Adaptive bitrate video caching and processing in mobile-edge computing networks," *IEEE Trans. Mobile Comput.*, vol. 18, no. 9, pp. 1965–1978, Sep. 2019.
[43] Y. Huang, X. Song, F. Ye, Y. Yang, and X. Li, "Fair and efficient caching algorithms and strategies for peer data sharing in pervasive edge computing environments," *IEEE Trans. Mobile Comput.*, vol. 19, no. 4, pp. 852–864, Apr. 2020.
[44] Y. Dong, S. Guo, J. Liu, and Y. Yang, "Energy-efficient fair cooperation fog computing in mobile edge networks for smart city," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 7543–7554, Oct. 2019.
[45] X. Cao, J. Zhang, and H. V. Poor, "An optimal auction mechanism for mobile edge caching," in *Proc. IEEE 38th Int. Conf. Distrib. Comput. Syst.*, 2018, pp. 388–399.
[46] S. Tang, C. Yu, and Y. Li, "Fairness-efficiency scheduling for cloud computing with soft fairness guarantees," *IEEE Trans. Cloud Comput.*, to be published, doi: 10.1109/TCC.2020.3021084.
[47] Y. Huang, J. Zhang, J. Duan, B. Xiao, F. Ye, and Y. Yang, "Resource allocation and consensus on edge blockchain in pervasive edge computing environments," in *Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst.*, 2019, pp. 1476–1486.
[48] A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker, and I. Stoica, "Dominant resource fairness: Fair allocation of multiple resource types," in *Proc. 8th USENIX Conf. Netw. Syst. Des. Implementation*, 2011, pp. 323–336.
[49] X. Cheng et al., "Uplink resource allocation for trade-off between throughput and fairness in C-RAN-based neighborhood area network," in *Proc. IEEE/IFIP Netw. Operations Manage. Symp.*, 2018, pp. 1–6.
[50] A. Li and G. Han, "A fairness-based MAC protocol for 5G cognitive radio ad hoc networks," *J. Netw. Comput. Appl.*, vol. 111, pp. 28–34, 2018.
[51] H. Xing, Y. Liu, A. Nallanathan, Z. Ding, and H. V. Poor, "Optimal throughput fairness tradeoffs for downlink non-orthogonal multiple access over fading channels," *IEEE Trans. Wireless Commun.*, vol. 17, no. 6, pp. 3556–3571, Jun. 2018.
[52] S. Rezvani, N. Mokari, M. R. Javan, and E. A. Jorswieck, "Fairness and transmission-aware caching and delivery policies in OFDMA-based hetnets," *IEEE Trans. Mobile Comput.*, vol. 19, no. 2, pp. 331–346, Feb. 2020.

Jingwen Zhou received the first bachelor's degree from South West University, China, in 2019, and the second bachelor's degree (Honours) from Deakin University, Australia, in 2020. She is currently working toward the PhD degree with Deakin University. Her research interests include edge computing, service computing, cloud computing and software engineering.

Feifei Chen (Member, IEEE) received the PhD degree from the Swinburne University of Technology, Australia, in 2015. She is a senior lecturer with Deakin University. Her research interests include software engineering, edge computing, cloud computing and green computing.

Qiang He (Senior Member, IEEE) received the first PhD degree from the Swinburne University of Technology, Australia, in 2009, and the second PhD degree in computer science and engineering from the Huazhong University of Science and Technology, China, in 2010. He is currently an associate professor with Swinburne. His research interests include edge computing, service computing, software engineering and cloud computing.



Xiaoyu Xia (Member, IEEE) received the master's degree from the University of Melbourne, Australia, and the PhD degree from Deakin University, Australia. He is a lecturer with the University of Southern Queensland, Australia. His research interests include edge computing, service computing, software engineering and cloud computing.

Rui Wang received the PhD degree from the University of New South Wales, Australia, in 2015. She is a senior research scientist with Data61, the Commonwealth Scientific and Industrial Research Organisation Australia. Her research areas include Digital Twin for Industry 4.0, Human-Computer Interaction, Virtual Reality, Augmented Reality and Mixed Reality.

Yong Xiang (Senior Member, IEEE) received the PhD degree in electrical and electronic engineering from the University of Melbourne, Australia. He is a professor with the School of Information Technology, Deakin University, Australia. His research interests include information security and privacy, signal and image processing, data analytics and machine learning, Internet of Things, and blockchain. He has published 6 monographs, more than 180 refereed journal articles, and numerous conference papers in these areas. He is the senior area editor of *IEEE Signal Processing Letters* and the associate editor of *IEEE Communications Surveys and Tutorials*. He was the associate editor of *IEEE Signal Processing Letters* and *IEEE Access*, and the guest editor of *IEEE Transactions on Industrial Informatics* and *IEEE Multimedia*. He has served as honorary chair, general chair, program chair, TPC chair, symposium chair and track chair for many conferences, and was invited to give keynotes with a number of international conferences.