Received September 20, 2020, accepted October 12, 2020, date of publication October 19, 2020, date of current version October 28, 2020.

*Digital Object Identifier 10.1109/ACCESS.2020.3031964*

# A New Framework for Mobile Edge Caching by Proposing Flexible User in Heterogeneous Cellular Networks

**PARISA ESLAMI[1], (Member, IEEE), MOHAMMAD HOSSEIN AMERIMEHR[2], AND SEYED POOYA SHARIATPANAHI[3]**
[1]Department of Electrical and Computer Engineering, Islamic Azad University, Tehran 1477893855, Iran
[2]ICT Research Institute (ITRC), Tehran 1439955471, Iran
[3]School of Electrical and Computer Engineering, College of Engineering, University of Tehran, Tehran 1439957131, Iran

Corresponding author: Mohammad Hossein Amerimehr (mh.amerimehr@itrc.ac.ir)

**ABSTRACT** The bursting increase in requesting wireless data has caused several issues in network peak-traffic duration. This negatively results in significant data delivery delay imposed on users that can eventually impact the network's quality of service and users' quality of experience. In this research, regarding mobile edge caching as a potential solution to decrease such delay, we propose a new framework in which we introduce the concept of the *flexible user* where he requests for a set of multiple files from the library with a unique feature, e.g., 5 movies within comedy genre from the library in the peak-traffic duration. The satisfactory criterion for the flexible user is to receive any of the files within the requested set. This definition of the flexible user indicates a new concept which captures interesting scenarios. In order to model this concept, we generalize the conventional Zipf distribution to a multivariate one as the modeling method for popular data. We formulate the problem of finding the optimal cache data placement, which minimizes the average total delivery delay in the network while satisfying the helpers' cache size constraints. To this end, we derive the average delivery delay per user as well as the average total delivery delay in the network, according to the new generalized Zipf distribution. Finding the optimal solution is proved to be NP-Hard. We leverage on the problem property to propose an efficient approximation method, called greedy algorithm, which performs within a constant factor as good as the optimal solution. Afterwards, we propose an algorithm called *speedy-greedy* to significantly reduce the computational complexity of the greedy algorithm while achieving the same performance. Simulation results indicate that our proposed framework significantly decreases the average total delivery delay of the system model that can help the network maintain its quality of service in network peak-traffic duration.

**INDEX TERMS** Mobile edge caching, data delivery and management, delivery delay, femto-caching.

## I. INTRODUCTION

Advances in telecommunications engineering - from old generations, 2G and 3G, to current generations, 4G, 5G and beyond - allow wireless cellular networks to provide higher security, speed, reliability and capacity in accessing data pool [1]. These advances have resulted in a tremendous increase in wireless data requisition by users [2]. It is estimated that users request for almost one terabyte of wireless data by 2022 [3]. This burst in wireless data demand imposes extremely high overhead to the network to process and deliver the requested data to users in network peak-traffic duration. In order for the network to be able to support such a high demand for wireless data, it is mandatory to merge some new technologies in communications such as internet of things (IoT), massive MIMO, mobile edge caching (MEC), mobile cloud computing (MCC), etc. [4]–[6].

As a working principle for such a data call by users, we have: when a user places a request for data, the network first locates where the source of the requested data is in the internet cloud or data centers [7]. Then, the requested data is fetched from its source, and is transferred throughout the forming elements of network such as data centers, backbone, backhaul cables, routers, switches, base stations, etc., to be

The associate editor coordinating the review of this manuscript and approving it for publication was Huaqing Li.

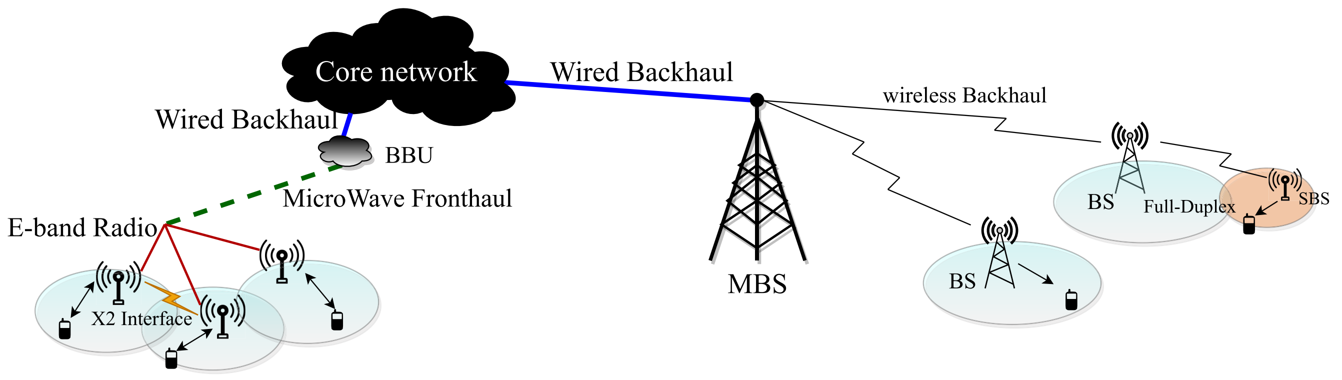


**FIGURE 1.** Simplified illustration of working principal of a network from receiving data request up to delivering data.

delivered to the user [8]. The general view of this process can be seen in Fig. 1. Although, the network elements have high operating capacity, they are not capable to handle the aforementioned explosive increase in data that ultimately results in traffic congestion [9], [10]. Network's backbone and backhauls are the elements that are most likely to significantly intensify traffic congestion due to their physical limitations [11]. Accordingly, they are considered as network bottlenecks [12]. Traffic congestion caused by backbone and backhauls increases network latency and results in a notable delay in transferring data from one point to another throughout the network. This eventually degrades not only the network's performance (Quality of Service -QoS-), but also users' Quality of Experience (QoE) [13], [14].

One solution to maintain the network's high performance in peak-traffic duration is to deploy a small-scale format of a macro base station (MBS) under the supervision of main MBS, as shown in Fig. 2. Small-scale MBSs, according to their size of coverage area and applications, can be named small-cell, pico-cell and femto cell [15]. From among these small-scale MBSs, femto cells are known for large storage capacity, flexibility and cost efficiency. That is the reason why they are the most studied small-scale MBSs in the literature [16]. Additionally, the main idea behind proposing femto cells was to bring users and terminal devices in the network closer to one another, which makes femto cells be generally beneficial with compensating poor cellular coverage, creating network capacity wherever needed, and improving network QoS by offloading traffic and overhead from the macro base station [17], [18]. This makes cellular regions to be heterogeneous networks (HetNets) nowadays. One approach to offload notable traffic and overhead from the network is to store popular data closer to users. This technique is called *caching* [19]. One advantage of using the caching technique is to reduce the need for large bandwidth in peak-traffic duration.

One particular application for which femto cells can be utilized is to function as distributed storages throughout the network. Thus, as of this functionality, these storages cache the most requested or popular data within the macro base station's coverage region [19]. This technique is called *femto-caching* [20]. In other words, what femto-caching

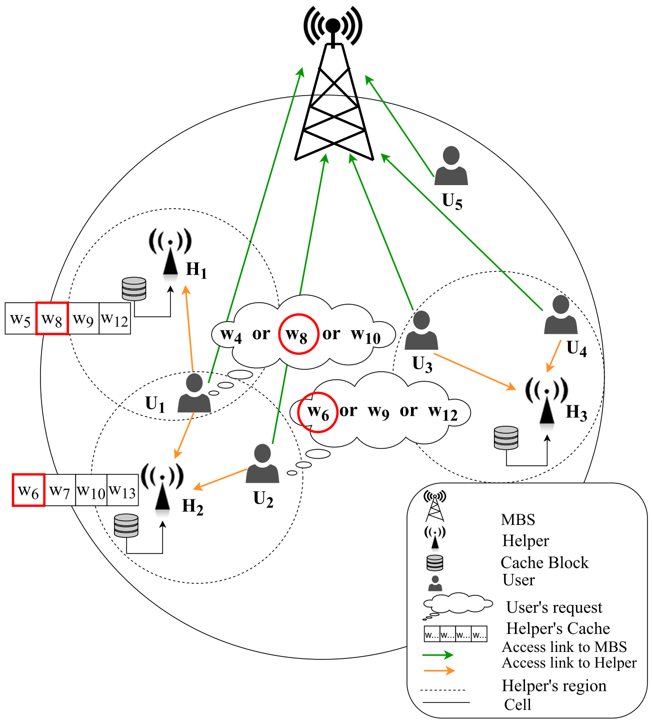


**FIGURE 2.** The considered system model in a heterogeneous wireless cellular network.

contributes to is bringing popular data closer to users so that the popular data will not be fetched and transferred throughout the network every time a user places a request for it.

On the other hand, femto-caching strongly assists with decreasing data delivery delay, overhead from the network, packet collisions and traffic congestion [21]. Other locations where popular data can be cached in networks elements are in the user equipment (UE) mostly in device-to-device (D2D) and internet of thing (IoT) applications [22]–[24], macro base stations (MBSs), small-cells and pico-cells, relays and C-RANs (a novel proposed architecture for 5G cellular networks) [25]. In the literature, femto cells that are used for femto-caching technique are usually referred to as *helpers* [26]. Each femto cell, in this manner, consists of a helper and cache storage as illustrated in Fig. 2.

In this article, we generalize data requisition format in MEC by considering that the user requests a set of data (multiple data) from data pool or library. Due to the fact that this generalization can provide significant flexibility to the users, the user in this research is called flexible user. Our contributions can be summarized as follows.

- We introduce the concept of the flexible user where the user requests for a set of files with a unique feature without having a preference on any of the files within this set. This concept captures interesting scenarios such as film genre and database features. A user may be interested in watching a movie from a specific genre (e.g., a user in the mood of watching a new comedy movie). Another scenario includes requesting any data from a database which has a specific feature.
- Considering the flexible user requests, the conventional Zipf can no longer be used to model the users' request probability mass function. Hence, We generalize the conventional Zipf distribution to a multivariate one to devise a model for flexible user requests.
- We formulate the problem of optimal cache data placement with helpers' cache size constraints. In this regard, we derive the average delivery delay per user as well as the average total delivery delay in the network, according to the new generalized Zipf distribution.
- Proving that the problem is NP-hard, we show that the problem exhibits a property which enables us to provide an efficient approximate algorithm. More precisely, we will prove that the problem is in the form of maximizing a monotone submodular function subject to matroid constraints.
- We invoke the problem structure to provide the greedy algorithm which is guaranteed to performs within a constant of $\frac{1}{2}$ as good as the optimal solution.
- Furthermore, we leverage on the problem structure to provide an algorithm called *speedy greedy* which achieves the same performance as the greedy algorithm in terms of delivery delay, while significantly reduces the computational complexity.

The rest of the paper is organized as follows. A summary of related works is presented in Section II. In Section III, the system model of the proposed new framework is presented. In Section IV, we formulate the problem of obtaining optimal popular data placement in helpers and investigate its solutions. In Section V, we propose greedy and speedy-greedy algorithms to solve the formulated problem in Section IV. In Section VI, we present the results from algorithms in Section V. Eventually, we conclude the paper in Section VII.

## II. RELATED WORKS

In [20], Shanmugam and Golrezaei mostly focused their work on video streaming delivery delays. They developed the idea of data placement in small base stations to minimize the total average delivery delay of their system model. In their work, the users request for only one data in the network peak-traffic duration. In their system model, the network attempts to deliver the only requested data to the user. Users' requests are assumed to be asynchronous meaning that users request one data at different times. Additionally, users receive the data from the helper offering the least delivery delay. This results in users to experience the least delay in the network to receive their requested data.

The work most close to our problem is [20]. We rely and expand upon this work. Reference [20] assumed that each user requests for a specific file, according to a single-variable Zipf distribution. However, in our problem we assume that each user has a preferred subset of files from which one the files should be delivered. Thus, we are working with distributions on subsets rather than single files. Hence, developing new problem formulation and solution is more challenging than [20].

A joint caching and processing framework for video on-demand in MEC networks has been proposed in [27], [28]. MEC servers collaborate to cache and transcode different chunks of one video content, in order to use the backhaul and storage resources more efficiently and reduce the content delivery delay.

Authors in [29] addressed collaborative service placement for edge computing in small cell networks. They proposed an efficient decentralized service placement algorithm based on graph coloring on the small cell network. Li *et al.* [30] studied optimal content allocation over a resilient caching network, where each cache may fail under some situations. They proposed a centralized algorithm which achieves a $(1-\frac{1}{e})-$approximation guarantee. Moreover, the authors devised a distributed algorithm based on concave relaxation of the objective function.

The goal of [31] is scheduling of downlink file transmission with the assistance of wireless cache nodes. Two scenarios are considered in this article: 1) the BS reactively casts the data to the requesting user with associated cache node, and 2) the BS proactively casts some data to the selected cache node without request. Downlink transmission resource minimization is formulated as a dynamic programming problem and approximated by a finite-horizon Markov decision process (MDP) to reduce the computational complexity.

Ioannidis and Yeh in [32] studied optimal data placement over a cache network where its topology and the demand rates are unknown. Their objective is to determine the data placement that maximizes the reduction of the routing cost due to intermediate caching. To obtain their goal, they proposed a distributed, adaptive algorithm that performs stochastic gradient ascent on a concave relaxation of the expected caching gain, and constructs a probabilistic content placement within a factor from the optimal, in expectation.

Sermpezis *et al.* have built their research [33] on the assumption that the requested data may not be cached in the helper that the user is connected to. Then, the network recommends other cached data to the user that have high similarity with the requested data. If the user accepts one of the recommendations, soft cache-hit occurs. They formulated the optimal femtocaching problem with soft cache hit and

**TABLE 1.** List of symbols.

| Symbol | Description |
|---|---|
| $F$ | Number of files |
| $U$ | Number of users |
| $M$ | Helper cache size |
| $N$ | Number of helpers |
| $K$ | Flexible index |
| $\mathcal{L}$ | Library of the requested files |
| $\mathcal{S}_l$ | A user's flexible request (preferred subset) |
| $\mathcal{R}(K)$ | Set of flexible requests containing $K$ files |
| $t_j$ | Number of connected users to helper ($h_j$) |
| $r_j$ | helper ($h_j$) coverage range |
| $r_0$ | MBS coverage range |
| $z_{h_j,u_i}$ | Delay between helper ($h_j$) and user ($u_i$) |
| $q_{w_f,h_j}$ | Indicator representing the caching of file ($w_f$) by helper ($h_j$) |
| d($h_j$,$u_i$) | Distance between helper ($h_j$) and user ($u_i$) |
| $P_r(\mathcal{S}_l)$ | Probability mass function of a flexible request $\mathcal{S}_l$ |

proposed an efficient algorithm with provable performance. It is worth noting that although other notations of flexibility has been considered in [33], it is completely different from our framework.

In cache networks, machine learning techniques can be applied to predict content popularity and design accurate content request model [34]–[37]. This can help to identify suitable cache contents which results in higher cache hit ratio and lower delivery delay. Unlike the works which exploit machine learning techniques to evaluate the distribution statistics of users' requests, in our paper the statistics of users' requests are assumed to given by a known distribution (i.e., generalized Zipf function).

## LIST OF SYMBOLS

For more readability of paper, below we provide the list of symbols and some descriptions exclusively for our system model in Table 1.

## III. SYSTEM MODEL AND NETWORK TOPOLOGY

In this research, we study a heterogeneous cellular region which includes an MBS capable of access to a library of $F$ files, i.e., $\mathcal{L} = \{w_1, w_2, \ldots, w_F\}$. This region also contains $N$ femto cell stations or helpers. Each of these helpers is capable of caching only $M$ files, in network off-peak hours, due to its storage limitations. This region, additionally, comprises $U$ users as shown in Fig. 2. All users, in this region, have connection to the MBS. On the other hand, only those users who are within the coverage range of a helper are able to have a connection with that helper. This implies that a user can have either zero, one or even more connections to helpers. We focus on one-directional connection where users can only receive files from MBS or helpers. The MBS is able to provide requested files to the users with the largest delay compared to the delays that helpers can provide the same files if they have them already cached. If the helpers, connected to the users, do not have the requested files cached, then the users will have to experience the largest delay to receive the files from MBS. MBS's long delay is considered to be the same for all users. On the other hand, helpers have different delays to provide the requested files to users. Therefore, we need to determine if $i^{th}$ user ($u_i$) has a connection with $j^{th}$ helper ($h_j$), and how much delay $u_i$ will experience by receiving the requested file from $h_j$. For the sake of simplicity, we ignore all physical channels non-idealities such as interference and fading to avoid additional complexity. Hence, a user can receive a file from a helper without error if there is a connection between them. Due to the inherent complexity of our problem, this assumption avoids additional complexity. We define a matrix $\mathbf{Z}$ as the delay matrix, which its element ($z_{h_j,u_i}$) represent the delay between $u_i$ and $h_j$.

$$\mathbf{Z} = \begin{bmatrix} z_{h_0,u_1} & z_{h_0,u_2} & \cdots & z_{h_0,u_U} \\ z_{h_1,u_1} & z_{h_1,u_2} & \cdots & z_{h_1,u_U} \\ z_{h_2,u_1} & z_{h_2,u_2} & \cdots & z_{h_2,u_U} \\ \vdots & \vdots & \ddots & \vdots \\ z_{h_N,u_1} & z_{h_N,u_2} & \cdots & z_{h_N,u_U} \end{bmatrix}_{((N+1)\times U)}$$

Some remarks should be noted in this matrix as follow:

- We have: $z_{h_j,u_i} \geqslant 0, \forall\ 1 \leqslant j \leqslant N$ and $1 \leqslant i \leqslant U$.
- If $z_{h_j,u_i} = \infty$, it means that there is no connection between $h_j$ and $u_i$.
- $z_{h_0,u_i} \geqslant 0, \forall\ 1 \leqslant i \leqslant U$, where it represents the connection delay between $u_i$ and the MBS.

Entries of matrix $\mathbf{Z}$ are dependent on users', helpers' and MBS' locations (Cartesian coordinates $\{x, y\}$) in the cellular region. More precisely, we have:

$$z_{h_j,u_i} = \begin{cases} Ct_j(d^{\alpha}(h_j, u_i)) & \text{, if} \quad d(h_j, u_i) \leqslant r_j \\ \infty & \text{, if} \quad d(h_j, u_i) > r_j \end{cases} \tag{1}$$

where $C$ and $\alpha$ are two constants. Distance between $h_j$ and $u_i$ is defined as:

$$d(h_j, u_i) \triangleq \sqrt{(x(h_j) - x(u_i))^2 + (y(h_j) - y(u_i))^2} \tag{2}$$

In (1), we drive $t_j$ as below where it shows the number of connected users to each helper:

$$t_j = \sum_{i=1}^{U} \mathbb{1}(d(h_j, u_i) \leqslant r_j) \tag{3}$$

where $\mathbb{1}(.)$ is the indicator function, $t_j$ is introduced because the bandwidth of each helper is shared among the users connected to it. Now that connections and delays between users and helpers in the cellular region are defined, we need to investigate the cached files in helpers. Consequently, we define a matrix $\mathbf{Q}$ as the cached content placement matrix, such that its element $q_{w_f,h_j}$ indicates if $f^{th}$ file ($w_f$) is cached in $h_j$.

$$\mathbf{Q} = \begin{bmatrix} q_{w_1,h_1} & q_{w_1,h_2} & \cdots & q_{w_1,h_N} \\ q_{w_2,h_1} & q_{w_2,h_2} & \cdots & q_{w_2,h_N} \\ \vdots & \vdots & \ddots & \vdots \\ q_{w_F,h_1} & q_{w_F,h_2} & \cdots & q_{w_F,h_N} \end{bmatrix}_{(F\times N)}$$

If $w_f$ is cached in $h_j$, then: $q_{w_f,h_j} = 1$; otherwise, $q_{w_f,h_j} = 0$. As it is usually considered in the literature, when the user requests for only one file $w_f$, file's popularity distribution follows the univariate Zipf distribution [16], [38]. This distribution is defined as follows:

$$\mathrm{P_r}(w_f) = \frac{(\mathrm{rank}(w_f))^{-\lambda}}{\sum_{f=1}^{F}(\mathrm{rank}(w_f))^{-\lambda}}, \quad w_f \in \mathcal{L} \tag{4}$$

where the rank$(w_f)$ shows each file's popularity rank in the library. Next, we add flexibility to users' requesting format to propose the ides of flexible users.

### A. FLEXIBLE USER: IDEA AND IMPLICATIONS

We herein present the idea of *flexible user* in addition to one of its characteristics. Moreover, we relate this concept to QoS and QoE.

*Definition:* A user in a wireless network is defined as a *flexible user* where it requests multiple data with a common feature, from which the user is interested in receiving anyone. This creates a subset of multiple data from the library. We refer to this subset as a *preferred subset*. In this manner, a flexible user would be satisfied even if it receives only one of the requested data from his preferred subset. As in a higher level of analysis, this simply increases the probability of one of the requested data to be delivered to a flexible user, which increases the QoE. Consequently, this constructively impacts QoS. Associated with this idea, we define the *flexibility index* where it shows the number of requested data by a flexible user i.e. preferred subset size, Let $\mathcal{R}(K)$ denotes the set of flexible requests (preferred subsets) that contain exactly $K$ files. In other words, $\mathcal{R}(K) \triangleq \{\mathcal{S}_l | \mathcal{S}_l \subseteq \mathcal{L}, |\mathcal{S}_l| = K\}$. The following example will more clarify this concept.

*Example:* We illustrate the application of flexible user concept with two important scenarios. The first scenario happens where a flexible user requests for some movies within the comedy genre. Meanwhile, other movies' factors are not important, e.g., the director, stars etc. This creates a preferred subset with multiple movies with a common feature, the genre of being comedy. Through an example, we later on statistically show that requesting for multiple data will increase the probability of one of the requested data to be cached in helper. The second scenario exemplifies a database where any data including a specific feature is of interest. For instance, a group of researchers who are conducting a research on the human genome. They are intending to use samples of people's DNA stream with a specific lung disease. For this research, the record of these people's age (people between the ages of 60 and 65) is important for their analysis. On the other hand, other factors such as the city of residence, place of birth, geographical area, and type of treatment do not play a notable role. This is where the definition of flexible user can help categorize the data. A clear intuition can be seen in Fig. 2 where $u_1$ and $u_2$ request for sets of $\{w_4, w_8, w_{10}\}$ and $\{w_6, w_9, w_{12}\}$, respectively. $u_1$ is connected to both $h_1$ and $h_2$, which have $w_8$ and $w_{10}$ cached, respectively. $u_1$ receives $w_8$ from $h_1$ since $h_1$ offers a smaller delay compared to what $h_2$ offers. This way, $u_1$ is satisfied since it has received one of the requested data with a smaller delay. The same procedure applies to $u_2$ with one difference; that is, $u_2$ is only connected to $h_2$. The delay offered by $h_2$ is much smaller than MBS. Thus, $u_2$ receives $w_6$ from $h_2$.

Our motivation to provide multi-variable Zipf distribution is that the flexible user requests a subset of files rather than a single file. Thus, we are working with distributions on subsets rather than single files. We are inspired by [39] to extend single-variable Zipf distribution (4) to generalized Zipf distribution (5). We generalize the distribution in (5) in order to model the popular data with a flexible user's format of the request. In order to accomplish this, we propose the following probability mass function (PMF) of flexible requests. It is worth noting that this generalized distribution is also a power-law distribution based on the sum of the ranks of files in each subset.

$$P_r(\mathcal{S}_l) = \frac{\left(\sum_{w_f \in \mathcal{S}_l} \mathrm{rank}(w_f)\right)^{-\lambda}}{\sum_{\mathcal{S}_l \in \mathcal{R}(K)} \left(\sum_{w_f \in \mathcal{S}_l} \mathrm{rank}(w_f)\right)^{-\lambda}}, \quad \mathcal{S}_l \in \mathcal{R}(K) \tag{5}$$

where rank$(w_f) = f$. The above equation represents a PMF, since every entity is positive and sums up to one. It should be noted that if flexibility index is $K = 1$, the distribution in (5) reduces to the normal Zipf distribution in (4). In other words, the univariate distribution as defined in (4) corresponds to the special case where the subset size is equal to 1.

*Example:* Let $\mathcal{L} = \{w_1, w_2, w_3, w_4\}$ and $K = 3$. $\mathcal{R}(3)$ has four elements including: $\mathcal{S}_1 = \{w_1, w_2, w_3\}$, $\mathcal{S}_2 = \{w_2, w_3, w_4\}$, $\mathcal{S}_3 = \{w_1, w_3, w_4\}$ and $\mathcal{S}_4 = \{w_1, w_2, w_4\}$. For instance, the probability that the flexible user requests $\mathcal{S}_2$ is obtained by $\mathrm{P_r}(\mathcal{S}_2) = \frac{(2+3+4)^{-\lambda}}{B}$, where $B = (1+2+3)^{-\lambda} + (1+2+4)^{-\lambda} + (1+3+4)^{-\lambda} + (2+3+4)^{-\lambda}$.

## IV. PROBLEM SETUP

In this section, we formulate the problem of finding the optimal data placement in helpers, minimizing the average total delivery delay. Then, we analyze the complexity of the optimization problem.

### A. DERIVATION OF AVERAGE DELIVERY DELAY PER USER

In order to derive the average total delivery delay per user, we consider a system model where all the parameters defined in the previous section such as placement matrix $\mathbf{Q}$, network topology and files' popularity distribution are known. Then, we derive the average delivery delay per user. Equation (6), as shown at the bottom of the page shows the average delay of the network for delivering all the requests. Next we discuss its derivation.

As mentioned in the system model, a user might have a connection to more than one helper. Therefore in order to find the helper with the least delay, the user sorts out the helpers based on their delivery delay in ascending order (Fig. 3) and downloads from the helper with the least delay which meets its request. The ascending order results in $\mathcal{H}(u_i) = \{h_I(1,u_i), h_I(2,u_i), \ldots, h_I(\beta_{u_i}, u_i)\}$, where $\beta_{u_i}$ is the number of helpers plus the base station. By default, all users in the cellular network have a connection to MBS, which is characterized by the largest delivery delay. Therefore, MBS holds the last position in the sorting order $\mathcal{H}(u_i)$. Let assume $h_I(m,u_i)$ is the helper with the minimum delay which contains at least one of the files within its requested set of files $\mathcal{S}_l$. The term (7) shows the indicator function for the case that the helper $h_I(m,u_i)$ contains at least one of the files within the request $\mathcal{S}_l$. Also, the term (8) indicates that none of the files requested by the user is cached in a helper that has less delivery delay compared to $h_I(m,u_i)$.

$$[1 - \prod_{w_f \in \mathcal{S}_l} (1 - q_{w_f, h_{I(m,u_i)}})] \tag{7}$$

$$[\prod_{j=1}^{m-1} \prod_{w_f \in \mathcal{S}_l} (1 - q_{w_f, h_{I(j,u_i)}})] \tag{8}$$

$$[\prod_{j=1}^{\beta_{u_i}-1} \prod_{w_f \in \mathcal{S}_l} (1 - q_{w_f, h_{I(j,u_i)}})] \tag{9}$$

Additionally, the term (9) is also an indicator where points to those requested files that are not cached in which any helpers connected to the user. As a result, the user needs to receive the file form MBS. Finally, $\overline{D}_{u_i}$ is the average delay for $i^{th}$ user.

### B. PROBLEM FORMULATION

The minimization of the total average data delivery delay by finding the optimal data placement in helpers can be expressed as Problem 1.

*Problem 1:*

$$\underset{\mathbf{Q}}{maximize} \quad \sum_{i=1}^{U} ( z_{h_0,u_i} - \overline{D}_{u_i}) \tag{10}$$

$$subject\ to \quad \sum_{f=1}^{F} q_{w_f,h_j} \leqslant M, \quad \forall h_j \tag{10-a}$$

$$\mathbf{Q} \in \{0,1\}^{F \times N} \tag{10-b}$$

Constraint (10-a) expresses the helpers' cache sizes limitations. The other constraint (10-b) points to the binary value of matrix $\mathbf{Q}$, which indicates if a file is cached in a helper. As discussed later, the problem structure enables us to provide an efficient solution, despite finding the optimal solution is computationally intractable.

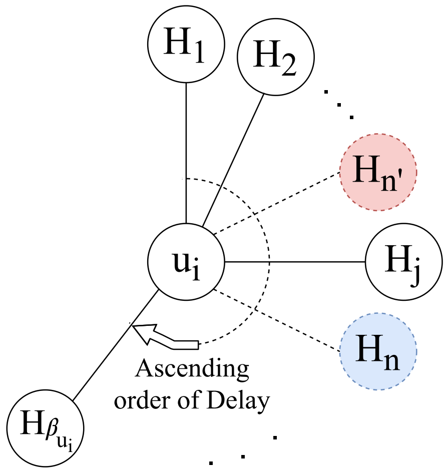


**FIGURE 3.** Sorted helpers $\{H_1 \to H_j \to H_{\beta_{u_i}}\}$ based on their delivery delay to user $u_i$, where $H_j$ is short form of $h_{I(j,u_i)}$.

### C. COMPLEXITY ANALYSIS

The optimal solution for Problem 1 provides the data placement in helpers which minimizes the total average data delivery delay. The optimization problem is an integer programming as all the variables are binary integers. This problem is NP-complete as stated below.

*Lemma 1:* Problem 1 is proved to lie within NP-complete problems.

*Proof:* In the case of $K = 1$, Problem 1 reduces to the one considered in [20]. Since the problem in [20] is proven to be NP-complete, thus, Problem 1 is also NP-complete. ■

## V. APPROXIMATION SOLUTIONS

As we proved that Problem 1 is an NP-complete problem, the exhaustive search algorithm as the optimal solution for problem 1 can be applied only to small-scale networks. On the other hand for practical scales, we need to utilize approximation solutions. One of the approximation solutions is called *greedy algorithm*. The Greedy algorithm is an algorithm paradigm that builds up a solution by making the local optimum choice at each iteration with the intention of finding the global optimum. There is no convergence guarantee in

$$\overline{D}_{u_i} = \sum_{m=1}^{\beta_{u_i}-1} z_{h_{I(m,u_i)},u_i} \sum_{\mathcal{S}_l \in \mathcal{R}(K)} \{[1 - \prod_{w_f \in \mathcal{S}_l} (1 - q_{w_f,h_{I(m,u_i)}})][\prod_{j=1}^{m-1} \prod_{w_f \in \mathcal{S}_l} (1 - q_{w_f,h_{I(j,u_i)}})] \cdot P_r(\mathcal{S}_l)\}$$
$$+ z_{0,u_i} \sum_{\mathcal{S}_l \in \mathcal{R}(K)} [\prod_{j=1}^{\beta_{u_i}-1} \prod_{w_f \in \mathcal{S}_l} (1 - q_{w_f,h_{I(j,u_i)}})] \cdot P_r(\mathcal{S}_l) \tag{6}$$

general. However, if the problem has a submodular structure with matroid constraints, the greedy algorithm performs within a constant factor as good as the optimal solution [40]. In the following, we will prove that Problem 1 has this property and the greedy algorithm guarantees an approximately optimal solution. Moreover, as discussed later, we invoke the submodular structure of the problem to provide an algorithm called *speedy greedy* which performs exactly the same as the greedy algorithm but significantly reduces the computational complexity. There are three steps in proving this [41]. Each following lemma and theorem will explore the steps in our work, and their corresponding proofs are provided right after them. First, we provide the definition of matroid:

*Definition:* A *matroid* is a structure that generalizes the notion of independence in linear algebra [42]. The most significant matroid's concept in vector space is the expression of independent sets so that the matroid is defined as a pair $(\mathcal{V}, \mathcal{I})$ consisting of a finite ground set $\mathcal{V}$ and $\mathcal{I}$. $\mathcal{I}$ is a collection of independent subsets of $\mathcal{V}$ satisfying the following properties [40]:

- $A \subset B \subset \mathcal{V}$ and $B \in \mathcal{I}$ implies $A \in \mathcal{I}$.
- $A, B \in \mathcal{I}$ and $|B| > |A|$ implies $\exists\ e \in B \setminus A$ such that $A \cup \{e\} \in \mathcal{I}$.

*Lemma 2:* Problem 1 constraints, (10-a) and (10-b), are in the form of matroid constraints.

*Proof:* Since Problem 1 constraints are identical to constraints of the problem considered in [20], it can be thus concluded that the provided proof in [20] can also prove that Problem 1 constraints, (10-a) and (10-b), are in the form of the matroid. ■

*Lemma 3:* The objective function of Problem 1 has monotonicity property.

*Proof:* Since by adding a new file to cache placement, it is more likely to download the request with less delay, the value of objective function cannot decrease. As a result, the objective function is monotone. ■

*Theorem 1:* The Problem 1 objective function is a submodular one.

*Proof:* The proof procedure is similar to [20] but details are different due to the flexibility of user request. If $\mathcal{V}$ is a finite ground set, a submodular function is a set function $f : 2^{\mathcal{V}} \to \mathbb{R}$, if for all sets $A, B \subset \mathcal{V}$ with $A \subset B$ and every $x \in \mathcal{V} \backslash B$, we have $f(A \cup \{x\}) - f(A) \geq f(B \cup \{x\}) - f(B)$. First we define a ground set $\mathcal{V}$ comprised of elements $x_{w_f,h_j}$ denoting the placement of file $w_i$ into the cache of helper $h_j$, i.e., $\mathcal{V} = \{x_{w_1,h_1}, x_{w_2,h_1}, \ldots, x_{w_F,h_N}\}$. Let $\mathcal{Q} \subset \mathcal{V}$ denotes the placement set corresponding to matrix $\mathbf{Q}$ such that $x_{w_f,h_j} \in \mathcal{Q}$ if and only if $q_{w_f,h_j} = 1$. We consider the set function $G_{u_i}(\mathcal{Q}) \triangleq z_{h_0,u_i} - \overline{D}_{u_i}$ and prove the submodularity of $G_{u_i}$. To this end, we will show that if the placement set $\mathcal{Q}$ becomes larger, marginal gain value (the amount of increase in $G_{u_i}$) of adding a new file to a helper decreases. We consider two placement sets $\mathcal{Q}$ and $\mathcal{Q}'$ where $\mathcal{Q} \subset \mathcal{Q}' \subset \mathcal{V}$. Consider adding a file $w_f$ to both placement sets in helper $h_j$ and focus on user request $\mathcal{S}_l$ which contains $w_f$. Let $\Delta_{\mathcal{Q}'} \triangleq G_{u_i}(\mathcal{Q}' \cup \{x_{w_f,h_j}\}) - G_{u_i}(\mathcal{Q}')$ and $\Delta_{\mathcal{Q}} \triangleq G_{u_i}(\mathcal{Q} \cup \{x_{w_f,h_j}\}) - G_{u_i}(\mathcal{Q})$. Also, $h_{I(n,u_i)}$ and $h_{I(n',u_i)}$ denote the helper with the least delay that user $u_i$ is able to retrieve at least one file from request $\mathcal{S}_l$, for the file placement $\mathcal{Q}$ and $\mathcal{Q}'$, respectively. Since $\mathcal{Q} \subset \mathcal{Q}'$, we infer $z_{h_{I(n,u_i)}} \geq z_{h_{I(n',u_i)}}$ (i.e., due to the larger set of files placed in helpers for the file placement $\mathcal{Q}'$, user $u_i$ can retrieve the request with less delay).

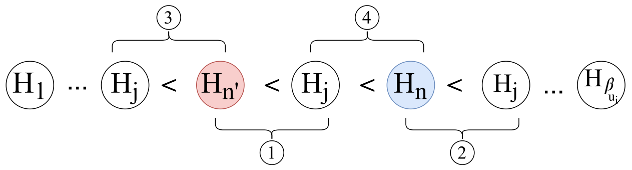


**FIGURE 4.** Possible states of $j^{th}$ helper placement in two sets of different combinations of how files can be cached in helpers.

By adding $w_f$ to both placement sets, two cases are considerable:

1) $z_{h_{I(n',u_i)}} < z_{h_{I(j,u_i)}}$: as illustrated in state 1 in Fig. 4, we deduce that according to placement $\mathcal{Q}'$, user $u_i$ can download the request from a helper with lower delay compared to $h_j^{u_i}$, i.e., $\Delta_{\mathcal{Q}'} = 0$. if $z_{h_{I(n,u_i)}} < z_{h_{I(j,u_i)}}$, as illustrated in state 2 in Fig. 4, we also infer $\Delta_{\mathcal{Q}} = 0$. Otherwise, by adding $w_f$ to file placement $\mathcal{Q}$, user $u_i$ is able to decrease the download delay by downloading the file from the helper $h_j$. In this case, $\Delta_{\mathcal{Q}} = Pr(\mathcal{S}_l)\{z_{h_{I(n,u_i)}} - z_{h_{I(j,u_i)}}\}$. Obviously, $\Delta_{\mathcal{Q}} \geq \Delta_{\mathcal{Q}'}$.

2) $z_{h_{I(n',u_i)}} > z_{h_{I(j,u_i)}}$: as illustrated in state 3 in Fig. 4, Hence, $\Delta_{\mathcal{Q}'} = Pr(\mathcal{S}_l)\{z_{h_{I(n',u_i)}} - z_{h_{I(j,u_i)}}\}$, similarly, as illustrated in state 4 in Fig. 4, we have $\Delta_{\mathcal{Q}} = Pr(\mathcal{S}_l)\{z_{h_{I(n,u_i)}} - z_{h_{I(j,u_i)}}\}$. Noting that $z_{h_{I(n',u_i)}} \leq z_{h_{I(n,u_i)}}$, we infer that $\Delta_{\mathcal{Q}} \geq \Delta_{\mathcal{Q}'}$.

Hence, we have proved the submodularity of $G_{u_i}$. Since the objective function of Problem 1 is summation of $G_{u_i}$, and the summation of submodular functions is also submodular, the proof is complete. ■

Now we have proved that Problem 1 is in the form of maximizing a monotone submodular function subject to matroid constraints. In this case, the greedy algorithm performs within a constant factor of $\frac{1}{2}$ as good as the optimal solution [41], [43]. Hence, we have the following lemma:

*Lemma 4:* Greedy algorithm 1 achieves a $\frac{1}{2}-$approximation solution for Problem 1.

## *A. GREEDY ALGORITHM*

As we proved in the previous section the problem has a nice property which allows us to efficiently exploit the greedy algorithm. This algorithm initially starts off working over an empty *placement set* ($\mathcal{A} = \emptyset$). In each step, a file with maximum marginal value is added to the placement set while considering the feasible solutions for the Problem 1. Let $\mathcal{A}_h$ denote the subset of $\mathcal{A}$ which is placed in the cache of helper $h$. $\mathcal{V}_h$ denotes the set of all files that might be cached in helper $h$. Also, $\mathcal{F}(\mathcal{A})$ denotes the feasible set of elements (files) which can be added to placement set $\mathcal{A}$, while satisfying the helper cache size constraints. i.e., $\mathcal{F}(\mathcal{A}) = \{x \in \mathcal{V} \backslash \mathcal{A} \mid |(\{x\} \cup \mathcal{A}_h) \cap \mathcal{V}_h| \leq M, \forall h\}$. This algorithm starts off from

**Algorithm 1** Greedy Algorithm for Proposed System Model

```
    Input: $\mathcal{V}$
    Output: $\mathcal{A}$
 1: Initialize
 2: $\mathcal{A} \leftarrow \emptyset$
 3: while CacheContent $(\mathcal{A}, h_j) \leq M$ for some j do
 4:     $x^* \leftarrow \underset{x \in \mathcal{F}(\mathcal{A})}{\text{argmax}} \sum_{u=1}^{U}(G_u(\mathcal{A} \cup \{x\}) - G_u(\mathcal{A}))$
 5:     $\mathcal{A} \leftarrow \mathcal{A} \cup \{x^*\}$
 6: end while
 7: function CacheContent$(\mathcal{A}, h_j)$
 8:     $t \leftarrow 0$
 9:     for all $x_{i,w_f} \in \mathcal{A}$ do
10:         if $i = h_j$ then
11:             $t \leftarrow t + 1$
12:         end if
13:     end for
14:     return $t$
15: end function
```

where there is no file cached in helpers. Thus, users will need to request their *preferred set* of files from MBS. This results in the initial delivery delay to be the largest one. In each iteration, the greedy algorithm adds one element with the maximum marginal value to the placement set while maintaining the feasibility of the solution. In other words, at each iteration a feasible file placement that maximizes the reduction in total average delivery delay (compared to the previous iteration) is selected and added to the placement set. This algorithm continues until each helper cache size is filled. Due to the submodularity property of the problem, the marginal value of each file in the placement set decreases while more files are added to $\mathcal{A}$. Algorithm 1 summarizes the greedy method.

### B. SPEEDY-GREEDY ALGORITHM

In this subsection, we propose a faster greedy algorithm, called *speedy-greedy*, which achieves the same performance as the standard greedy algorithm (Algorithm 1). Our proposed algorithm is based on the accelerated variant of the greedy algorithm which was originally proposed by Minoux et. al in [44]. This method leverages on the monotonicity and submodularity of the objective function in order to avoid unnecessary calculation in the selection process of the cache placement. Since the speedy greedy algorithm only eliminates the unnecessary calculations, the final solution of the speedy greedy exactly coincides with the standard greedy algorithm.

Let $\Delta_x(\mathcal{A})$ denote the marginal value of adding element $x$ to the placement set $\mathcal{A}$, i.e., $\Delta_x(\mathcal{A}) = \sum_{u=1}^{U}(G_u(\mathcal{A} \cup \{x\}) - G_u(\mathcal{A}))$. The key idea is that as the placement set grows, the marginal value $\Delta_x$ can never increase. In other words, if $\mathcal{A}^{(k)}$ and $\mathcal{A}^{(k')}$ denote the placement set at iteration $k$ and $k'$, respectively ($k < k'$) then, since $\mathcal{A}^{(k)} \subset \mathcal{A}^{(k')}$

**Algorithm 2** Speedy Greedy Algorithm for Proposed System Model

```
    Input: $\mathcal{V}$
    Output: $\mathcal{A}$
 1: Initialize
 2: $\mathcal{A} \leftarrow \emptyset$
 3: $k \leftarrow 0$
 4: $\rho(x) \leftarrow \infty$        ; $\forall x \in \mathcal{V}$
 5: while CacheContent $(\mathcal{A}, h_j) \leq M$ for some j do
 6:     $k \leftarrow k + 1$
 7:     if $k = 1$ then
 8:         $\rho(x) \leftarrow \Delta_x(\mathcal{A})$        ; $\forall x \in \mathcal{V}$
 9:         $x^* \leftarrow \underset{x \in \mathcal{V}}{\text{argmax}}\ \rho(x)$
10:         $\mathcal{A} \leftarrow \mathcal{A} \cup \{x^*\}$
11:     else
12:         $\tilde{x} \leftarrow \underset{x \in \mathcal{F}(\mathcal{A})}{\text{argmax}}\ \rho(x)$
13:         $\rho(\tilde{x}) \leftarrow \Delta_{\tilde{x}}(\mathcal{A})$
14:         $\mathcal{X} \leftarrow \{x \mid x \in \mathcal{F}(\mathcal{A}), \rho(x) \geq \rho(\tilde{x})\}$
15:         $\rho(x) \leftarrow \Delta_x(\mathcal{A})$        ; $\forall x \in \mathcal{X}$
16:         $x^* \leftarrow \underset{x \in \mathcal{X}}{\text{argmax}}\ \rho(x)$
17:         $\mathcal{A} \leftarrow \mathcal{A} \cup \{x^*\}$
18:     end if
19: end while
20: function CacheContent$(\mathcal{A}, h_j)$
21:     $t \leftarrow 0$
22:     for all $x_{i,w_f} \in \mathcal{A}$ do
23:         if $i = h_j$ then
24:             $t \leftarrow t + 1$
25:         end if
26:     end for
27:     return $t$
28: end function
```

it holds that $\Delta_x(\mathcal{A}^{(k)}) \geq \Delta_x(\mathcal{A}^{(k')})$ for all $x \in \mathcal{V}$; Therefore, we maintain an upper bound $\rho(x)$ (initially set to $\infty$) on the marginal value of all elements sorted in decreasing order. In the first iteration, similar to the standard greedy algorithm, we set $\mathcal{A}^{(0)} = \emptyset$ and calculate the element with the largest marginal value ($x^*$) to construct $\mathcal{A}^{(1)}$. We also calculate $\rho(x)$ for all the elements. In the next iterations, say iteration $k$, we calculate the feasible element at the top of the list, i.e., $\tilde{x} = \underset{x \in \mathcal{F}(\mathcal{A}^{(k-1)})}{\text{argmax}}\ \rho(x)$. We then update the upper bound for $\tilde{x}$, i.e., $\rho(\tilde{x}) = \Delta_{\tilde{x}}(\mathcal{A}^{(k-1)})$ If after the update $\rho(\tilde{x}) > \rho(x)$ for all feasible element $x \neq \tilde{x}$, submodularity guarantees that $\tilde{x}$ is the element with the largest marginal value. Therefore, $\mathcal{A}^{(k)} := \mathcal{A}^{(k-1)} \cup \{\tilde{x}\}$. Hence, instead of computing $\Delta_x(\mathcal{A}^{(k-1)})$ for every feasible element, we only need one calculation, which significantly reduces the computational complexity. On the other hand, if $\rho(\tilde{x})$ is not larger than all the other elements $\rho(x)$, we just need to update $\rho(x)$ for the elements that are above $\rho(\tilde{x})$, due to submodularity. In other words, in general, instead of calculating $\Delta_x$ for every feasible elements, we just need to calculate $\Delta_x$ for those ones that are above the threshold determined as $\rho(\tilde{x})$. This leads to a significant reduction in computational complexity, since usually the recalculation is necessary only for a small fraction of the elements. This algorithm continues until each helper cache block is filled. This procedure is summarized in Algorithm 2.

In conclusion, speedy greedy algorithm invokes the problem structure to eliminate unnecessary calculations and reduce computational complexity. Based on the above discussion, speedy greedy algorithm (like the standard greedy algorithm) provides $\frac{1}{2}-$approximation solution for Problem 1 (lemma 4).

### C. PIPAGE ROUNDING ALGORITHM

Calinescu et. al. provide a randomized algorithm for maximization of a general monotone submodular function subject to matroid constraints [41]. This algorithm consists of two steps. First, the integral problem is turned into a continuous problem using multilinear extension. Second, using the pipage rounding technique, the fractional solution is converted to an integral solution. This algorithm achieves a $(1 - \frac{1}{e})-$approximation. It is worth noting that although this algorithm gives a better performance guarantee than the greedy solution, its computational complexity is very high which makes it inappropriate for implementation especially when the size of network grows. Specifically, running time of the pipage algorithm is $O((NF)^8)$. On the other hand, for the greedy algorithm, running time is $O(F^2\ N^2\ U)$ [20]. Consequently, greedy solution is more suitable in general, since the size of problem is typically large.

It should be noted that a low complexity deterministic algorithm which provides a $(1 - \frac{1}{e})-$approximation is available for the special case where the delay to receive the requested data from helpers is equal for all users, i.e., $z_{h_j,u_i}$ for all $h_j \in \beta_{u-i} - 1$ and $u_i \in U$ equates $z_{1,1}$ and also $z_{1,1} < z_{h_0,u_i}$. In this case, the Problem 1 is simplified as $D_{u_i}$ is derived as:

$$\overline{D}_{u_i} = z_{1,1} \sum_{\mathcal{S}_l \in \mathcal{R}(K)} \{[1 - \prod_{w_f \in \mathcal{S}_l} (1 - q_{w_f,h_j})] \cdot P_r(\mathcal{S}_l)\} + z_{0,u_i} \sum_{\mathcal{S}_l \in \mathcal{R}(K)} [\prod_{w_f \in \mathcal{S}_l} (1 - q_{w_f,h_j})] \cdot P_r(\mathcal{S}_l) \quad (11)$$

This deterministic algorithm involves solving a linear relaxation problem and pipage rounding [45]. We refer the interested reader to [20] for further details. The running time of the algorithm tends to $O((U+N)^{3.5}F^{3.5})$. Despite the simplicity of the algorithm, it is not interesting for our scenario as we assume that delay to receive the requested data from helpers depends on their distance (eq (1)).

## VI. NUMERICAL RESULTS

In this section, we illustrate our system model and evaluate the performance of the proposed algorithms. We first build a cellular region as depicted in Fig. 5. In this region, the MBS's and helpers' coverage ranges are 100 and 35 meters, respectively. Users' and helpers' locations are first randomly determined and then set to be fixed for comparison purposes. Fixed parameters in modeling the popular data based on generalized Zipf are as in Table 2. Users' delay for receiving data from either MBS or helpers are calculated from (1). In order to investigate the performance of the greedy algorithm (and speedy greedy), we compare the results with the exhaustive search, which leads to the optimum data placement. This comparison can be observed in Fig. 6. As explained before, exhaustive search algorithm can only be applied to small-scale networks, and has no practical applications. Hence, we only conduct a preliminary comparison with the greedy algorithm for a given scenario.

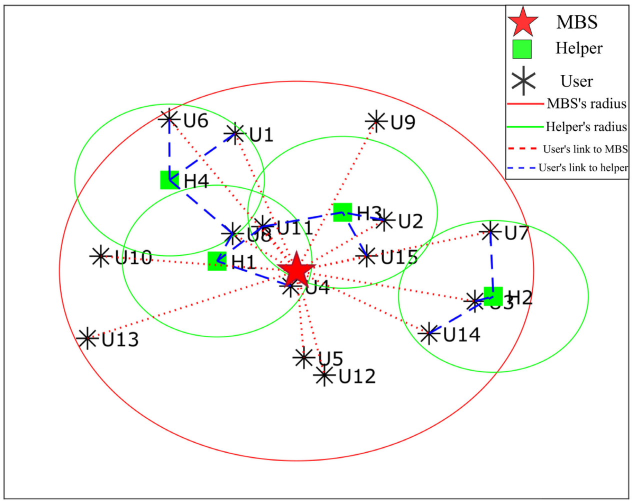


**FIGURE 5.** This network area represents one of the simplified system model defined in section III. In this area, there are 15 users ($U = 15$), 4 helpers ($H = 4$), and a MBS.

**TABLE 2.** Fixed parameters in greedy and speedy-greedy analysis.

| Parameter | Value |
|---|---|
| $F$ | 15 |
| $\lambda$ | 2 |
| $\alpha$ | 1 |
| $C$ | $10^{-3}$ |
| $r_0$ | 100 m |
| $r_j$ | 35 m |

Fig. 6 shows the total average delivery delays for all users where the horizontal axis shows the impact of the value of the exponent characterizing the generalized Zipf distribution. As $\lambda$ increases, the popularity of the first file approaches 1. This is where the performance of the two algorithms tends to merge. Moreover, We expect that the average delivery delay decreases while the popularity of requested data increases. That is because as the popularity of a file increases, caching the file is beneficial to a larger number of users. This can eventually result in more notable decrease in the total average delivery delay.

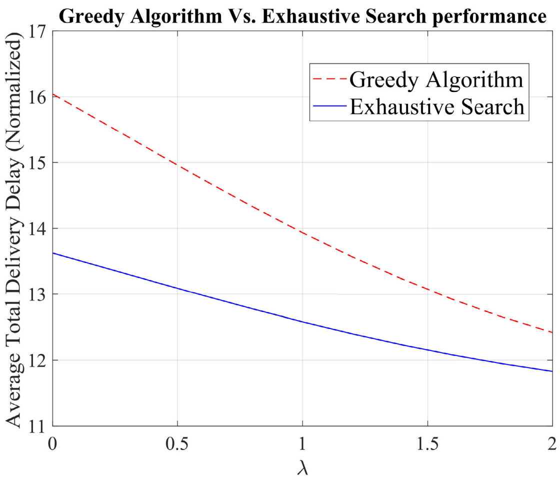
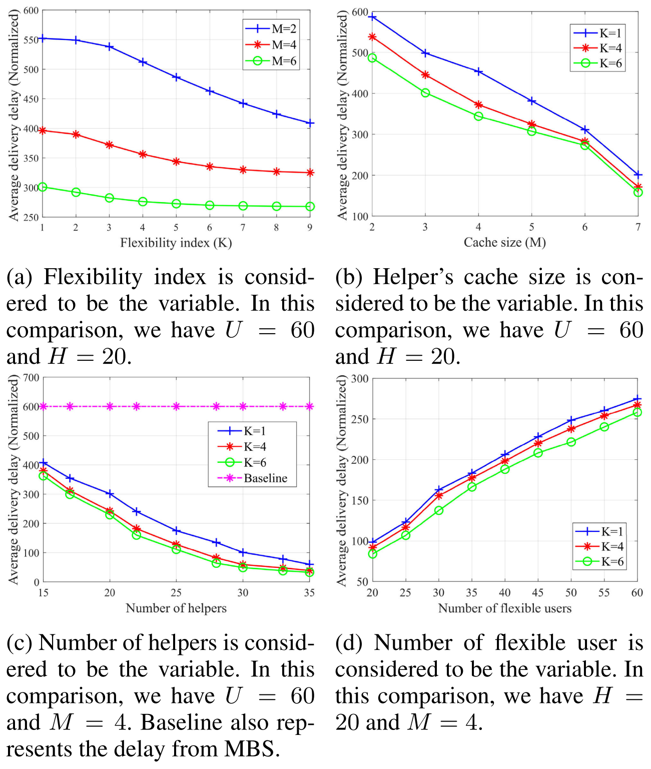


**FIGURE 6. Delivery delay decrease comparison for exhaustive search and greedy algorithms to find the optimal data placement. In this comparison, we have set $U = 10$, $K = 2$ and $M = 2$.**



(a) Flexibility index is considered to be the variable. In this comparison, we have $U = 60$ and $H = 20$.

(b) Helper's cache size is considered to be the variable. In this comparison, we have $U = 60$ and $H = 20$.

(c) Number of helpers is considered to be the variable. In this comparison, we have $U = 60$ and $M = 4$. Baseline also represents the delay from MBS.

(d) Number of flexible user is considered to be the variable. In this comparison, we have $H = 20$ and $M = 4$.

**FIGURE 7. Greedy algorithm's performance with the effect of different parameters of the system model.**

## A. GREEDY ALGORITHM ANALYSIS

In order to comprehensively evaluate the performance of this algorithm, we investigate the impact of the following important factors on the total average delivery delay in the cellular region. Moreover, in order to compare our results with one of the prominent existing works in literature, we set $K = 1$ to represent the system model provided in [20].

### 1) FLEXIBILITY INDEX EFFECT

Fig. 7a shows the effect of flexibility index on the total average delivery delay. By increasing the flexibility index, the total average delivery delay notably decreases. This implies the fact that if the size of the flexible requested data set is larger, it is more likely that the user is satisfied with the data cached in a nearby helper, instead of downloading its request from the MBS or a distant helper. Our system model's experiment shows decrease in average delivery delay in the curves where $M = 2$, $M = 4$ and $M = 6$ by 15%, 14% and 11%, respectively for the points where $K > 1$ compared to $K = 1$ (i.e. the result in [20].)

### 2) CACHE SIZE EFFECT

The effect of cache size on total average delay is illustrated in Fig. 7b. This figure shows that increasing the helper's cache size will more drastically affect on decreasing total average delivery delay. Based on the greedy algorithm's operation, we know that as the number of iterations of the algorithm increases, it more closely reaches to optimal result where it minimizes the average delivery delay. For example, the curves for $K = 4$ and $K = 6$ indicate lower average delivery delay by 18% and 26%, respectively compared to the curve where $K = 1$. On the other hand, our defined system model in this experiment decreases the average delivery delay by 22%. Increasing the cache size allows us to cache a larger number of popular files in the helpers. Hence, users can download their requests with a smaller delay. It should be noted that increasing the cache size induces the storage cost. Therefore, we are allowed to increase it up to a particular limit.

### 3) NUMBER OF HELPERS EFFECT

The Number of helpers is one of the most important factors on total average delivery delay. It is clear that as the number of helpers increases, the total average delivery delay will decrease because the network will be capable to cache more popular data and cover more number of flexible users. Fig. 7c shows how drastically number of helpers will affect the total average delivery delay. In this figure, the curves where $K = 4$ and $K = 6$ show lower average delivery delay by 23% and 31%, respectively compared to the curve where $K = 1$. Hence, we can say that this experiment decreased the system model's average delivery delay by 27%.

### 4) NUMBER OF FLEXIBLE USERS EFFECT

It is important to evaluate how the number of flexible users will affect the total average delivery delay based on the flexibility index. Fig. 7d shows the increase of total average delivery delay when number of flexible users increases. It is clear that by randomly adding flexible users in the cellular region, the number of requested data increases. This can finally result in increasing the total average delivery delay. Even though increasing the number of flexible users increases the average

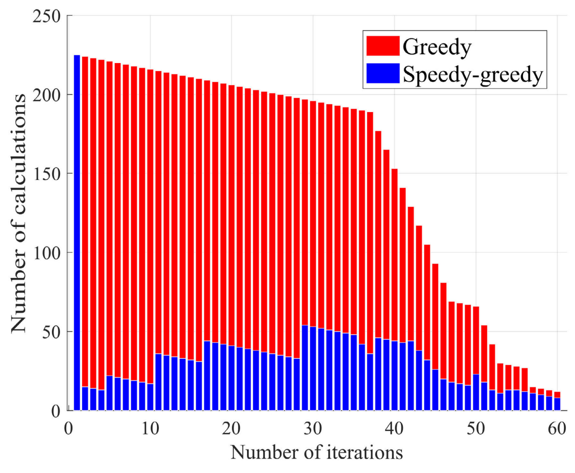


**FIGURE 8. Computational complexity comparison of greedy and proposed speedy-greedy algorithms where $H = 15$, $M = 4$, $U = 60$, $F = 15$ and $K = 5$.**

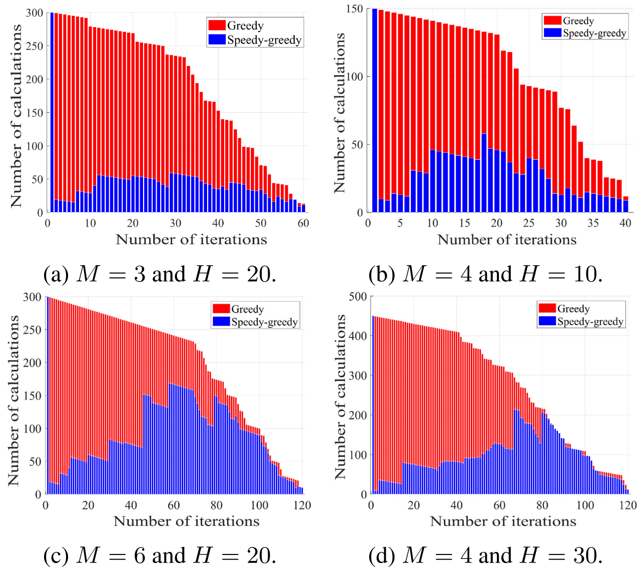


**FIGURE 9. Computational complexity comparison of greedy and speedy-greedy algorithms with number of helpers and their cache sizes as variables in system model.**

delivery delay, increasing the flexibility index contributes to lower average delivery delay for a certain number of users. To be specific, the curve where $K = 4$ and $K = 6$ indicates lower average delivery delay by 9% and 15%, respectively compared to the curve where $K = 1$.

### B. SPEEDY-GREEDY ANALYSIS

As mentioned earlier, speedy-greedy algorithm achieves the same performance as the standard greedy algorithm with less computational complexity. We herein compare the computational complexity of the speedy-greedy and standard greedy algorithm. In order to have a fair comparison with the greedy algorithm's performance, we set the variable parameters of both algorithms to be the same. The purpose of this comparison is to show how much less computational complexity is required by the speedy-greedy algorithm. Fig. 8 shows a general computational complexity comparison of the greedy and speed-greedy algorithm in terms of the required number of iterations for a given scenario.

Two of the factors that significantly affect the computational cost are number of helpers and their cache size. Therefore, we investigate the required number of iterations for different values of these factors in Fig. 9.

As can be seen, the speedy-greedy algorithm reduces the computational cost by 70% where $H = 10$ and $M = 4$. Similarly, the computational cost is reduced by 64% for the speedy-greedy algorithm with the same number of cache sizes while $H = 30$. On the other hand, by changing the value of the helper's cache size, the computational cost for speedy-greedy algorithm is reduced by 77% and 55% where $M = 3$ and $M = 6$, respectively with $H = 20$.

In order to compare our work with existing works in this area, we set $K = 1$ to represent the system model provided in [20]. This speedy greedy algorithm achieves the same performance as the greedy algorithm provided in [20]. However, as depicted in Fig. 10, the computational complexity of our proposed speedy-greedy algorithm is 74% and 63% lower in Fig. 10a and Fig. 10b, respectively.

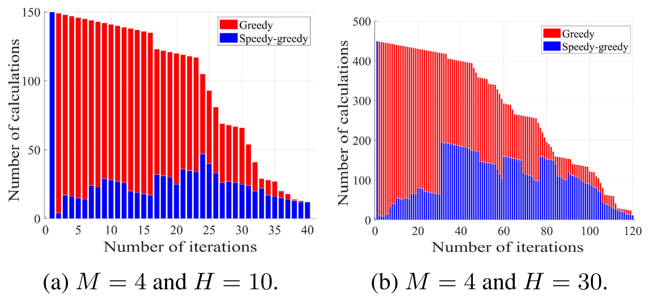


**FIGURE 10. Computational complexity comparison of greedy and speedy-greedy algorithms where $K = 1$.**

## VII. CONCLUSION

In this research, we proposed the concept of the flexible user where it requests a preferred subset of data in the library and is satisfied by receiving any data within this subset. We derived the generalized Zipf distribution in order to model the users' request probability. We addressed the problem of optimal cache placement in order to minimize the total average delivery delay. We showed that although finding the optimal solution is an NP-hard problem, but the problem exhibited submodularity property that can be leveraged to provide an efficient approximated algorithm (greedy) with provable performance. We also proposed another algorithm, speedy-greedy, which significantly reduces the computational complexity while achieving the same performance. According to our numerical results, the computational complexity is reduced by at least 50% for different flexibility indexes greater than 2. This algorithm reduces computational complexity by at least 60% where the flexibility index equals 1. Numerical results indicate that our proposed ideas of flexible user will end up decreasing the average data delivery to users by 19% in comparison with works in the literature. This will result in more flexibility at users' side as well as requesting for multiple files. All of these benefits together increases users' QoE in the peak-traffic duration of a HetNet utilizing mobile edge caching.

## REFERENCES

[1] A. Kiani and N. Ansari, "Edge computing aware NOMA for 5G networks," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 1299–1306, Apr. 2018.

[2] J. G. Andrews, S. Buzzi, W. Choi, S. V. Hanly, A. Lozano, A. C. Soong, and J. C. Zhang, "What will 5G be?" *IEEE J. Sel. Areas Commun.*, vol. 32, no. 6, pp. 1065–1082, Jun. 2014.

[3] M. Agiwal, A. Roy, and N. Saxena, "Next generation 5G wireless networks: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 3, pp. 1617–1655, 3rd Quart., 2016.

[4] M. R. Rahimi, J. Ren, C. H. Liu, A. V. Vasilakos, and N. Venkatasubramanian, "Mobile cloud computing: A survey, state of art and future directions," *Mobile Netw. Appl.*, vol. 19, no. 2, pp. 133–143, Apr. 2014.

[5] N. Chalaemwongwan and W. Kurutach, "Mobile cloud computing: A survey and propose solution framework," in *Proc. 13th Int. Conf. Electr. Eng./Electron., Comput., Telecommun. Inf. Technol. (ECTI-CON)*, Jun. 2016, pp. 1–4.

[6] J. Yao, T. Han, and N. Ansari, "On mobile edge caching," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2525–2553, 3rd Quart., 2019.

[7] X. Sun and N. Ansari, "Latency aware workload offloading in the cloudlet network," *IEEE Commun. Lett.*, vol. 21, no. 7, pp. 1481–1484, Jul. 2017.

[8] T. Minh Nguyen, W. Ajib, and C. Assi, "Designing wireless backhaul heterogeneous networks with small cell buffering," *IEEE Trans. Commun.*, vol. 66, no. 10, pp. 4596–4610, Oct. 2018.

[9] L. Li, G. Zhao, and R. S. Blum, "A survey of caching techniques in cellular networks: Research issues and challenges in content placement and delivery strategies," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 3, pp. 1710–1732, 3rd Quart., 2018.

[10] E. Bastug, M. Bennis, and M. Debbah, "Living on the edge: The role of proactive caching in 5G wireless networks," *IEEE Commun. Mag.*, vol. 52, no. 8, pp. 82–89, Aug. 2014.

[11] Y. Cui and D. Jiang, "Analysis and optimization of caching and multicasting in large-scale cache-enabled heterogeneous wireless networks," *IEEE Trans. Wireless Commun.*, vol. 16, no. 1, pp. 250–264, Jan. 2017.

[12] K. Hamidouche, W. Saad, M. Debbah, J. B. Song, and C. S. Hong, "The 5G cellular backhaul management dilemma: To cache or to serve," *IEEE Trans. Wireless Commun.*, vol. 16, no. 8, pp. 4866–4879, Aug. 2017.

[13] M. Alreshoodi and J. Woods, "Survey on QoE*QoS* correlation models for multimedia services," 2013, *arXiv:1306.0221*. [Online]. Available: http://arxiv.org/abs/1306.0221

[14] H. J. Kim, D. H. Lee, J. M. Lee, K. H. Lee, W. Lyu, and S. G. Choi, "The QoE evaluation method through the QoS-QoE correlation model," in *Proc. 4th Int. Conf. Netw. Comput. Adv. Inf. Manage.*, vol. 2, Sep. 2008, pp. 719–725.

[15] J. G. Andrews, H. Claussen, M. Dohler, S. Rangan, and M. C. Reed, "Femtocells: Past, present, and future," *IEEE J. Sel. Areas Commun.*, vol. 30, no. 3, pp. 497–508, Apr. 2012.

[16] M. Newman, "Power laws, Pareto distributions and Zipf's law," *Contemp. Phys.*, vol. 46, no. 5, pp. 323–351, Sep. 2005.

[17] Z. Hu, Z. Zheng, T. Wang, L. Song, and X. Li, "Caching as a service: Small-cell caching mechanism design for service providers," *IEEE Trans. Wireless Commun.*, vol. 15, no. 10, pp. 6992–7004, Oct. 2016.

[18] D. Malak, M. Al-Shalash, and J. G. Andrews, "Optimizing content caching to maximize the density of successful receptions in device-to-device networking," *IEEE Trans. Commun.*, vol. 64, no. 10, pp. 4365–4380, Oct. 2016.

[19] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *IEEE Trans. Inf. Theory*, vol. 60, no. 5, pp. 2856–2867, May 2014.

[20] K. Shanmugam, N. Golrezaei, A. G. Dimakis, A. F. Molisch, and G. Caire, "FemtoCaching: Wireless content delivery through distributed caching helpers," *IEEE Trans. Inf. Theory*, vol. 59, no. 12, pp. 8402–8413, Dec. 2013.

[21] X. Sun and N. Ansari, "EdgeIoT: Mobile edge computing for the Internet of Things," *IEEE Commun. Mag.*, vol. 54, no. 12, pp. 22–29, Dec. 2016.

[22] M. Gregori, J. Gómez-Vilardebó, J. Matamoros, and D. Gunduz, "Wireless content caching for small cell and D2D networks," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 5, pp. 1222–1234, May 2016.

[23] K. Zahoor, K. Bilal, A. Erbad, and A. Mohamed, "Service-less video multicast in 5G: Enablers and challenges," *IEEE Netw.*, vol. 34, no. 3, pp. 270–276, May 2020.

[24] B. Liu, K. Poularakis, L. Tassiulas, and T. Jiang, "Joint caching and routing in congestible networks of arbitrary topology," *IEEE Internet Things J.*, vol. 6, no. 6, pp. 10105–10118, Dec. 2019.

[25] I. Parvez, A. Rahmati, I. Guvenc, A. I. Sarwat, and H. Dai, "A survey on low latency towards 5G: RAN, core network and caching solutions," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 3098–3130, 2018.

[26] N. Golrezaei, A. F. Molisch, A. G. Dimakis, and G. Caire, "Femtocaching and device-to-device collaboration: A new architecture for wireless video distribution," *IEEE Commun. Mag.*, vol. 51, no. 4, pp. 142–149, Apr. 2013.

[27] E. Baccour, A. Erbad, K. Bilal, A. Mohamed, and M. Guizani, "PCCP: Proactive video chunks caching and processing in edge networks," *Future Gener. Comput. Syst.*, vol. 105, pp. 44–60, Apr. 2020.

[28] K. Bilal, E. Baccour, A. Erbad, A. Mohamed, and M. Guizani, "Collaborative joint caching and transcoding in mobile edge networks," *J. Netw. Comput. Appl.*, vol. 136, pp. 86–99, Jun. 2019.

[29] L. Chen, C. Shen, P. Zhou, and J. Xu, "Collaborative service placement for edge computing in dense small cell networks," *IEEE Trans. Mobile Comput.*, early access, Oct. 7, 2019, doi: 10.1109/TMC.2019.2945956.

[30] J. Li, T. Khoa Phan, W. Koong Chai, D. Tuncer, G. Pavlou, D. Griffin, and M. Rio, "DR-cache: Distributed resilient caching with latency guarantees," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2018, pp. 441–449.

[31] B. Lv, L. Huang, and R. Wang, "Joint downlink scheduling for file placement and delivery in cache-assisted wireless networks with finite file lifetime," *IEEE Trans. Commun.*, vol. 67, no. 6, pp. 4177–4192, Jun. 2019.

[32] S. Ioannidis and E. Yeh, "Adaptive caching networks with optimality guarantees," *IEEE/ACM Trans. Netw.*, vol. 26, no. 2, pp. 737–750, Apr. 2018.

[33] P. Sermpezis, T. Giannakas, T. Spyropoulos, and L. Vigneri, "Soft cache hits: Improving performance through recommendation and delivery of related content," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 6, pp. 1300–1313, Jun. 2018.

[34] J. Shuja, K. Bilal, E. Alanazi, W. Alasmary, A. Alashaikh, and A. Y. Zomaya, "Applying machine learning techniques for caching in edge networks: A comprehensive survey," 2020, *arXiv:2006.16864*. [Online]. Available: http://arxiv.org/abs/2006.16864

[35] G. Plastiras, M. Terzi, C. Kyrkou, and T. Theocharidcs, "Edge intelligence: Challenges and opportunities of near-sensor machine learning applications," in *Proc. IEEE 29th Int. Conf. Appl.-Specific Syst., Architectures Processors (ASAP)*, Jul. 2018, pp. 1–7.

[36] S. Bommaraveni, T. X. Vu, S. Chatzinotas, and B. Ottersten, "Active content popularity learning and caching optimization with hit ratio guarantees," *IEEE Access*, vol. 8, pp. 151350–151359, 2020.

[37] H. S. Goian, O. Y. Al-Jarrah, S. Muhaidat, Y. Al-Hammadi, P. Yoo, and M. Dianati, "Popularity-based video caching techniques for cache-enabled networks: A survey," *IEEE Access*, vol. 7, pp. 27699–27719, 2019.

[38] P. G. Sankaran, N. U. Nair, and P. John, "A family of bivariate Pareto distributions," *Statistica*, vol. 74, no. 2, pp. 199–215, 2014.

[39] K. V. Mardia, "Multivariate Pareto distributions," *Ann. Math. Statist.*, vol. 33, no. 3, pp. 1008–1015, Sep. 1962.

[40] A. Krause and D. Golovin, "Submodular function maximization," *Tractability, Pract. Approaches Hard Problems*, vol. 3, no. 19, pp. 71–104, 2012.

[41] G. Calinescu, C. Chekuri, M. Pál, and J. Vondrák, "Maximizing a monotone submodular function subject to a matroid constraint," *SIAM J. Comput.*, vol. 40, no. 6, pp. 1740–1766, Jan. 2011.

[42] L. A. Wolsey and G. L. Nemhauser, *Integer and Combinatorial Optimization*, vol. 55. Hoboken, NJ, USA: Wiley, 1999.

[43] M. L. Fisher, G. L. Nemhauser, and L. A. Wolsey, "An analysis of approximations for maximizing submodular set functions—II," *Math. Program. Study*, vol. 8, no. 1, pp. 73–88, 1978.

[44] M. Minoux, "Accelerated greedy algorithms for maximizing submodular set functions," in *Optimization techniques* (Lecture Notes in Control and Information Sciences). Berlin, Germany: Springer, 1978, pp. 234–243.

[45] A. A. Ageev and M. I. Sviridenko, "Pipage rounding: A new method of constructing algorithms with proven performance guarantee," *J. Combinat. Optim.*, vol. 8, no. 3, pp. 307–328, Sep. 2004.

**PARISA ESLAMI** (Member, IEEE) received the M.Sc. degree from Islamic Azad University, Tehran, Iran, in 2020. During her graduate study, she specifically worked on modeling, evaluation, and analysis of literature approaches for mobile edge caching in heterogeneous networks alongside with proposing a new framework of delivery data in mobile cache computing. She is currently working on employing new methods of machine learning, such as LSTM-DNN and Residual NN, queuing theory, and coded caching to improve the proposed framework. She has researched on frequency-domain array signal processing to decrease BER in spatial-multiplexing MIMO systems.

**MOHAMMAD HOSSEIN AMERIMEHR** received the B.Sc. degree (Hons.) in electrical engineering from the Isfahan University of Technology, Isfahan, Iran, in 2005, and the M.Sc. and Ph.D. degrees in electrical engineering from the Sharif University of Technology, Tehran, Iran, in 2007 and 2014, respectively. He was a Visiting Researcher with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON, Canada, from July 2011 to March 2012. Since 2016, he has been a Faculty Member with the ICT Research Institute (ITRC), Iran. His research interests include wireless communications, analytical modelling of communication networks, and network optimization.

**SEYED POOYA SHARIATPANAHI** received the B.Sc., M.Sc., and Ph.D. degrees from the Department of Electrical Engineering, Sharif University of Technology, Tehran, Iran, in 2006, 2008, and 2013, respectively. He is currently an Assistant Professor with the School of Electrical and Computer Engineering, University of Tehran. Before joining the University of Tehran, he was a Researcher with the Institute for Research in Fundamental Sciences (IPM), Tehran. His research interests include information theory, network science, wireless communications, and complex systems. He was a recipient of the Gold Medal at the National Physics Olympiad, in 2001.