AUT Journal of Electrical Engineering
10.22060/EEJ.2024.22977.5579

# Cache Assignment for a Flexible Mobile User in Wireless Heterogeneous Networks

Mohammad H. Amerimehr[1], Parisa Eslami[2], Nahid Amani[1,*], Sara Efazati[1]

[1]Department of Communication Technology, ICT Research Institute (ITRC), Tehran, Iran
[2] Department of Electrical and Computer Engineering, Islamic Azad University, Tehran, Iran
[*]Corresponding author, Email: n_amani@itrc.ac.ir

**Abstract:**

With the proliferation of smart mobile devices, there is an ever-increasing demand for multimedia content. To avoid congestion in backhaul links, mobile edge caching is a promising solution that can reduce delivery delays and improve users' quality of experience. In this regard, the requested content can be downloaded from a nearby small cell access point (also called helper) instead of a base station with a lower delay. In this paper, we address the problem of finding the optimal cache data placement to minimize the total delivery delay. We suppose the users are flexible in the sense that they request a set of multiple files from the library with a unique feature and are satisfied if any file within the requested set is received.

Moreover, in the system model, the interference and the mobility of users are considered. More precisely, the effect of interference from other helpers is incorporated in calculating the delivery delay, and a random waypoint model is exploited to address the mobility of users within the network. Because of the complexity of the problem, finding the optimal solution is NP-hard. We prove that the problem is in the form of maximizing a monotone submodular function subject to matroid constraints. We exploit this property to provide an efficient approximate solution (i.e., a greedy algorithm) that is guaranteed to perform within a constant of $\frac{1}{2}$ as well as the optimal solution. Simulation results validate the efficiency of our proposed algorithm.

**Keywords:**

Mobile edge caching (MEC), mobility-aware, flexible user, efficient caching strategy.

## 1. Introduction

Currently, with the rapid rise in demand for video-on-demand and live streaming services that are associated with content delivery networks (CDNs), the heterogeneous network of Fifth and Sixth Generation Mobile Networks (5G and 6G) and beyond is experiencing challenges related to bandwidth inefficiency, delays in content delivery, and resource allocation [1, 2]. Specifically, as the number of content requests grows, these efficiency-related aspects of the network decline due to the heavy traffic congestion, resulting in significantly reduced bandwidth availability. With the evolution of cellular network technologies, macro base stations (MBSs) have advanced significantly. Each generation of mobile networks brings improvements in data rates, network capacity, and latency. However, the network backbone and macro base station backhauls' capacity does not assist in removing these issues effectively. Limited available bandwidth in the backbone network can restrict the amount of data that can be transmitted simultaneously. Moreover, inefficient routing, network congestion, or suboptimal backhaul links can introduce latency and delay in data transmission. This delay can negatively affect interactive applications, particularly those requiring real-time responsiveness, such as voice calls or online gaming. As a result, bottlenecks can lead to packet loss, increased latency, and reduced network performance [3]. Therefore, it brings about a decrease in the quality of service (QoS) as well as users' quality of experience (QoE) in network traffic [4-6].

Caching holds promise for delivering desired material to users with minimal delay. By keeping data in quick storage technologies and close to the users who are seeking it, caches' primary goal is to accelerate the accessibility of data and cut down on transportation expenses [7, 8]. In mobile edge caching, caching servers are deployed at the edge of the network, such as base stations, access points, or small data centers [9, 10]. These servers store copies of popular content, including videos, images, software updates, and other frequently accessed data. When a user requests content, the caching server located nearest to the user can deliver the content directly, reducing the latency and network congestion associated with retrieving the data from a distant server.

**1-1- Related Works**

10.22060/EEJ.2024.22977.5579

The advantages of mobile edge caching have attracted lots of researchers recently [11-21].

Cao et al. designed a mobility-aware framework for joint routing and caching based on a learning approach [11]. They first formulated a network cost function and minimized it by using their proposed federated routing and popularity learning (FRPL) approach. Also, a novel content transmission protocol was suggested to avoid the retransmission of unnecessary data and improve the cache efficiency.

In [12], the authors proposed a cooperative service cache placement and routing scheme with the goal of maximizing the time utility with resource constraints like bandwidth, computation, storage capabilities, and deadline constraints for latency-sensitive applications. They proposed a heuristic algorithm to solve the problem of cooperative service placement.

Rui Wang et al. explored and took advantage of user mobility patterns and proposed a mobility-aware caching placement strategy. The strategy employed maximizes data offloading ratios. A dynamic programming algorithm was utilized to achieve benchmark performance and lower complexity; however, the algorithm was restricted to monotone submodular maximization. They proposed a time-efficient greedy algorithm and achieved a ratio of about $\frac{1}{2}$. Ultimately, they found that users moving at very low or very high speeds should cache the most popular files, while users moving at medium speeds should cache less popular files to avoid duplication [14].

Ge Ma et al. showed that geographic request distribution can be highly diverse in mobile systems and that requested content depends on changing locations and periods. These request patterns encourage joint caching strategies. They also use cellular and Wi-Fi caching solutions. Their caching strategy involved two-part cache storage, cross-location references, and content to cache. Results demonstrated that their algorithm was more effective than the least recently used (LRU) and Least Frequently Used (LFU) algorithms in terms of cache hit rate and service rate [15].

Meiyan Song et al. proposed a mobility-aware algorithm that dynamically adjusts caching and offloading decisions based on users' movement patterns [16]. Their mobility model focused on the statistical properties of contacts between mobile users concerning user-side recommendation and offloading. An exponential distribution was assumed for contact duration, and free movement was considered for users within a certain range but not in direct contact. Poisson point processes were adopted for modeling user, receiver, and device-to-device (D2D) transmitter locations. Content placement and offloading strategies were optimized to ensure efficient content delivery for mobile users. Simulations were utilized to evaluate the effectiveness of this approach.

Gul-E-Laraib et al. simultaneously examine user preferences and user mobility in caching methods for mobile edge computing servers (MEC) to determine content popularity. They proposed a Mobility and User Preference-aware Content (MUPAC) caching framework for MEC. A collaborative filtering model along with cosine similarity was utilized to establish relationships between multiple users. Time-series clustering was used to determine mobility patterns and trajectories. Overall, the results demonstrated that MUPAC improved cache speed, cache hit ratio, and delay. MUPAC increased the cache hit ratio by 12% and reduced delay by 5% when compared to LRU, LFU, and FIFO [17]. [18] addressed a wireless caching system and derived the throughput and average delay of the users. In their study, they considered the effect of interference from other helpers on system performance. However, they assumed that users were fixed. [19] investigated cache placement on small base stations (SBS) and a device-to-device (D2D) network and optimized the transmission power of SBSs and mobile devices to reduce energy cost. The authors in [20] proposed a mobility-aware data caching mechanism in which diverse data demands and user interference were taken into account. The proposed data caching mechanism is based on a greedy algorithm to maximize data offloading. The connectivity pattern among users has been modeled by the use of exponential distribution.

All of the aforementioned studies consider users with rigid requests. It means that each user requests for specific files. In our previous work [22], the idea of flexible users was proposed, where the users request a

10.22060/EEJ.2024.22977.5579

set of files with a unique feature without having a preference for any of the files within this set. The application of the flexible user concept has interesting scenarios, such as requesting any data from a database that has a specific feature. For example, a user may request some movies in the comedy genre without having a preference for other movie factors like directors or stars. In [22], the users were considered to be fixed and there was no mobility in the network. Cache placement is an idea that is very applicable and useful in mobile networks with mobile users. For a more realistic scenario, the users' mobility should be considered [19, 21]. Also, [22] suffers from the lack of nonidealities physical channel modeling, such as noise and interference.

### 1-2- Paper Contributions

Intending to overcome these challenges, in this paper we address cache placement in a heterogeneous cellular network in which flexible users with mobility and channel nonidealities such as noise and interference are considered. Table 1 summarizes the state-of-the-art related works. As described in Table 1, cache placement with these problem assumptions has not been addressed so far.

Table 1. Comparison of Related Works

| Reference | Problem | Objective | Solution type | Problem assumptions | | |
|---|---|---|---|---|---|---|
| | | | | User Mobility | Non-ideal channel | User flexibility |
| [9] | Cache placement | Min delay | approximation | ✗ | ✗ | ✗ |
| [11] | Mobility-aware routing strategy & cache placement | Min network cost | heuristic | ✓ | ✗ | ✗ |
| [14] | Mobility-aware cache placement in D2D network | Max data offloading ratio | approximation | ✓ | ✗ | ✗ |
| [16] | Joint User-side recommendation & D2D-assisted cache placement | Max operator's utility | approximation | ✓ | ✓ | ✗ |
| [17] | Cache placement based on user location & preference | Max cache hit ratio | heuristic | ✓ | ✗ | ✗ |
| [18] | Cache placement with random mobility | Max throughput | heuristic | ✗ | ✓ | ✗ |
| [19] | Mobility-aware cache placement in 5G network | Max cache hit ratio | approximation | ✓ | ✓ | ✗ |
| [21] | Mobility-aware cache placement in D2D network | Min system cost | approximation | ✓ | ✗ | ✗ |
| [22] | Cache placement for flexible users | Min delay | approximation | ✗ | ✗ | ✓ |
| This work | Cache placement for flexible users with mobility | Min delay | approximation | ✓ | ✓ | ✓ |

10.22060/EEJ.2024.22977.5579

Our contributions can be summarized as follows:

· We consider the cache placement problem in wireless heterogeneous networks with flexible users. Moreover, the mobility of users as well as channel nonidealities like noise and interference are incorporated in the system model.

· We formulate the problem of finding the optimal cache placement in order to minimize the average total delivery delay subject to cache size constraints. In this regard, we derive delay per user as a function of signal-to-interference & noise ratio (SINR) as well as the average total delivery delay in the network. Moreover, we employ a random waypoint user mobility model to characterize the mobility of users.

· Proving that finding the optimal solution is NP-hard, we seek an efficient approximation solution. By exploiting the problem structure, we provide a greedy algorithm that performs within a constant factor of $\frac{1}{2}$ as good as the optimal solution. In this regard, we show that the problem is monotone submodular with matroid constraints.

The rest of the paper is organized as follows: In Section 2, we introduce the system model, which involves constructing the network topology for a flexible user with mobility-aware features. Section 3 presents the problem formulation for calculating the average delivery delay per user as well as the average total delivery delay and determining the optimal popular data placements. Section 4 applies the aforementioned algorithms to solve the problem discussed in Section 3. In section 5, we present the numerical results. And finally, we provide the conclusion of the paper in Section 6. The symbols used in the system model are summarized in section 7.

## 2. System Model and Setup

As shown in Fig. 1, the cellular region is modeled as a square region with an initial random distribution of users $u_i$, $i \in \{1, \ldots, U\}$, and helpers $n_h$, $h \in \{1, \ldots, H\}$. An MBS supervises the helpers and has access to a data library ($\mathscr{L}$). The data library contains uniformly-sized files $\mathscr{L} = \{f_1, \ldots, f_F\}$ (where $f_l$ denotes the

10.22060/EEJ.2024.22977.5579

$l^{th}$ file) that adopt an extended Zipf distribution; this distribution is used for determining the popularity of multiple data through calculations based on probability mass functions, which is consistent with the definition of flexible users. Each helper has a coverage area, which is assumed to be a circle of radius $R$. The storage capacity of each helper is limited to $M$ files. Users within a coverage area are connected to the corresponding helper in that area. All users are connected to MBS to deliver the requested files, although the delivery delay is assumed to be notably greater compared to that of a helper. In this system model, users are connected to as many helpers as they are within their coverage areas. If the helpers connected to a user do not have any of the requested files, the user has to receive the files from the MBS with a larger delay. All the channels between users and helpers are assumed to be statistically independent. For simplicity, we assume that the links are the line of sight (LOS). The channel gain is given by

$$h_{u_i,n_j} = \sqrt{\beta(d_{u_i,n_j})^{-\kappa}} \tag{1}$$

where $\beta$, $\kappa$ and $d_{u_i,n_j}$ are the path loss at reference distance $1m$, the path loss exponent for the link and distance between the $i^{th}$ user and $j^{th}$ helper, respectively.

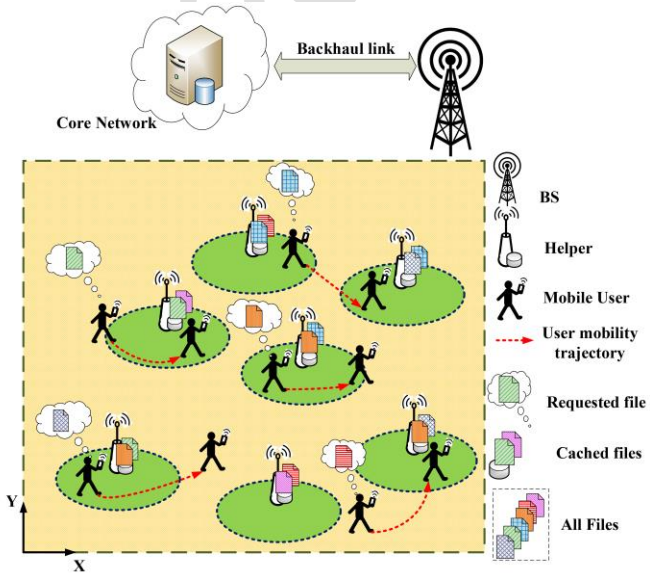


Fig. 1. System model

10.22060/EEJ.2024.22977.5579

Assuming that all the helpers transmit at the same frequency, the received SINR at the $i^{th}$ user from the $j^{th}$ helper (inside its coverage area) is derived as:

$$\gamma_{ji} = \frac{P|h_{ji}|^2}{\sum_{k \in \mathcal{F}(i), k \neq j} P|h_{ki}|^2 + \sigma^2} \tag{2}$$

where $P$ and $\sigma$ are the transmit power of the helpers and the standard deviation of Gaussian noise, respectively. Also, $\mathcal{F}(i)$ denotes the set of helpers connected to the $i^{th}$ user.

Let $L$ denote the size of the requested file. The delay for the $i^{th}$ user get a file from the $j^{th}$ helper (if cached, the requested file) can be derived as:

$$\phi_{n_j, u_i} = \frac{L}{r_{j,i}} = \begin{cases} \dfrac{L}{B\log_2(1+\gamma_{ji})}; j \in \mathcal{F}(i) \\ \infty \text{ ; otherwise} \end{cases} \tag{3}$$

where $r_{j,i}$ denotes the throughput of the channel between the $i^{th}$ user and the $j^{th}$ helper, and $B$ denotes the bandwidth of the channel.

We consider that each user requests a file at the beginning of a time slot to simplify the model. Content transmission takes place during the time slot. We assume that the network remains static during content transmissions. However, at the beginning of the next time slot, the network topology will be updated according to the mobility of the users. We employ a random waypoint user mobility model to characterize the mobility of the users [23]. In this model, a user selects a direction determined by an angle $\theta$ uniformly distributed between $[0, 2\pi]$, and the user moves in the selected direction for a random distance $d$ between $[0, d_{max}]$. $d_{max}$ can be determined by the random speed of a pedestrian between $[0, v_{max}]$ and the duration of a timeslot. The movements of mobile users are assumed to be independent of each other in the network. Hence, the location of user $u_i$ can be updated as follows:

10.22060/EEJ.2024.22977.5579

$$x_{u_i}^{(t+1)} = x_{u_i}^{(t)} + d\cos\theta$$
$$y_{u_i}^{(t+1)} = y_{u_i}^{(t)} + d\sin\theta \quad (4)$$

We then build the network topology and the user's requested set of files' popularity model accordingly. In order to form the network topology, we need to define two matrices that characterize the connections of different elements in this region. We first define a matrix $\mathbf{\Phi}$ that its elements $\phi_{n_m,u_i}$ determine the delay between helper $n_m$ and user $u_i$. Therefore, this matrix will have the dimensions of $(H+1)\times C$ since the MBS is considered an additional helper that is connected to all users by default. The delay for the connection between users and MBS is notated as $\phi_{MBS}$. We also define another matrix $\mathbf{\Omega}$ that implies the data placements in helpers in this region. This matrix is also known as the cache placement matrix. This matrix's elements $\omega_{f_l,n_m}$ indicate if a file $f_l$ is cached in helper $n_m$.

$$\bar{\Psi}_{u_i} = \sum_{S_v \in A(\Gamma)} \sum_{m=1}^{\delta_{u_i}-1} \phi_{n_{I(m,u_i)},u_i} \left( \underbrace{\left[1 - \prod_{f_l \in S_v} \left(1-\omega_{f_l,n_{I(m,u_i)}}\right)\right]}_{a} \underbrace{\left[\prod_{j=1}^{m-1} \prod_{f_l \in S_v} \left(1-\omega_{f_l,n_{I(j,u_i)}}\right)\right]}_{b} \cdot P_r(S_v) \right) + \phi_{0,u_i} \sum_{S_v \in A(\Gamma)} \left( \underbrace{\left[\prod_{f_l \in S_v}^{\delta_{u_i}-1} \prod_{f_l \in S_v} \left(1-\omega_{f_l,n_{I(j,u_i)}}\right)\right]}_{c} \cdot P_r(S_v) \right) \quad (5)$$

After building the network topology, we need to derive the user's requested set of files' popularity model. We will use the extended Zipf distribution for flexible users [22]. This distribution represents a probability mass function (PMF) of a subset of the library's files that is requested by a flexible user to receive one of its desired multiple files. The probability mass function of the flexible user is:

$$P_r(S_v) = \frac{\left(\sum_{f_l \in S_v} \text{rank}(f_l)\right)^{-\lambda}}{\sum_{S_v \in \mathcal{R}(K)} \left(\sum_{f_l \in S_v} \text{rank}(f_l)\right)^{-\lambda}}, \; S_v \in \mathcal{R}(K) \quad (6)$$

where $\mathcal{R}(K)$ denotes the set of flexible requests (preferred subsets) that contain exactly $K$ files and $\text{rank}(f_l) = l$.

### 3- Problem Formulation

In this section, we formulate the problem of finding the optimal popular data placements for helpers with mobile users. To achieve this, we first need to formulate the problem of calculating the average delivery

10.22060/EEJ.2024.22977.5579

delay per user ($\overline{\Psi}_{u_i}$) regarding all valid sets of file $S_v$ with a size of $K$ ($\forall S_v \in \Lambda(\Gamma)$) from the library. In general, the caching strategy will not be changed in the duration of the time slot $t$, since we assume that the network remains static during the time slot and topology updates take place at the beginning of the next slot. Hence, the system delay $\overline{\Psi}^t_{u_i}$ is considered constant during time slot $t$. Additionally, at each time slot, the users request new files, and the content will be delivered by the end of the slot. We optimize the caching strategy at each time slot in order to minimize the sum average delivery delay $\sum \overline{\Psi}^t_{u_i}$.

The aforementioned problem can be formulated as Optimization Problem 1. In this problem, $\delta_{u_i} - 1$, $n_{I(m,u_i)}, u_i$ represents the number of connected helpers to $u_i$ excluding MBS, and the helper with index $m$ that is characterized with the least delay to $u_i$, respectively. In Problem 1, the term $\prod_{f_l \in S_v} (1 - \omega_{f_l, n_{I(m,u_i)}})$ is an indicator function where it points to all feasible placements in matrix $\Omega$. The term (a) in Eq. (5) shows an indicator function that indicates the presence of at least one file from the set of requested files in the helper connected in $t$. In other words, it shows the first helper $n_{I(m,u_i)}, u_i$ that has at least one of the requested files cached in the aforementioned sorting order of helpers in the previous section. The term (b) in Eq. (5) shows none of the files requested by the user is cached in a helper that has less delivery delay compared to $n_{I(m,u_i)}, u_i$. Also, it indicates that either the user is not connected to time at $t$ or the helper does not have any of the requested files in the time slot $t$. Hence, we can determine the helper that has cashed at least one of the files requested by the user by multiplying the terms.

Additionally, the term (c) in (5) is also an indication function where it points to those requested files that are not cached in any helpers connected to the user at time slot $t$. As a result, the user needs to receive the file from MBS and experience the largest delay in the system model. To put all of these together, the term (7) indicates where the user receives one of the requested files from the helper that is connected. The term (8) implies when the user receives one of the requested files from MBS.

10.22060/EEJ.2024.22977.5579

$$\sum_{S_v \in \Lambda(\Gamma)} \sum_{m=1}^{\delta_{u_i}-1} \phi_{n_{I(m,u_i)},u_i} \{(a)\cdot(b)\cdot P_r(S_l)\} \tag{7}$$

$$\phi_{0,u_i} \sum_{S_v \in \Lambda(\Gamma)} (c)\cdot P_r(S_l) \tag{8}$$

Next, we formulate the process of finding the optimal popular data placements in helpers by considering the mobility of the user as in Problem 1 while considering each helper's cache block capacity ($C$).

**Problem 1.**

$$\underset{\boldsymbol{\Omega}}{maximize} \quad \sum_{i=1}^{U} (\phi_{0,u_i} - \bar{\Psi}_{u_i}) \tag{9}$$

$$subject\ to \quad \sum_{l=1}^{F} \omega_{f_l,n_m} \leqslant C, \quad \forall n_m \tag{9-a}$$

$$\boldsymbol{\Omega} \in \{0,1\}^{W\times H} \tag{9-b}$$

It is worth noting that minimizing the average total delivery delay in the system model can be achieved by maximizing the difference between the MBS's delay, which represents the highest delay in the system, and the average delivery delay per user calculated in Problem 1. Solving this problem is restricted to helpers' cache block capacity (9-a) as well as (9-b), which refers to binary elements of matrix **Ω**. We will show the NP-hardness of Problem 1 in the following lemma.

**Lemma 1.** *Problem 1 is NP-hard.*

*Proof:* For the special case of a non-flexible user (i.e., $K=1$), the structure of Problem 1 is similar to the optimization problem considered in [9]. Since the problem in [9] is proven to be NP-hard, thus, Problem 1 is also NP-hard.

**4- Proposed Algorithm**

Since Problem 1 is NP-hard, finding the optimal solution is intractable, especially when the size of the network grows. Instead, we exploit the problem structure to provide an efficient approximation solution (i.e., the greedy algorithm), which performs within a constant factor as well as the optimal solution. We

10.22060/EEJ.2024.22977.5579

will show that Problem 1 is monotone submodular with matroid constraints. To demonstrate that the constraints of Problem 1 adhere to the matroid framework, it is sufficient to identify feasible placements within a ground set consisting of disjoint sets. These placements can be matched with the algebraic mathematical definition of matroid constraints [24]. Furthermore, the monotonicity property of the objective function in Problem 1 can be demonstrated by introducing a new file to the cache placement. When prioritizing requests with lower delays for download, the value of the objective function cannot decrease. Hence, the objective function exhibits monotonic behavior. Now, we prove that Problem 1 is submodular.

**Theorem 1.** *The objective function of Problem 1 is submodular.*

*Proof:* If $V$ is a finite ground set, a submodular function is a set function $\mathrm{f}$ if for all sets $A, B \subset V$ with $A \subset B$ and every $x \in V \setminus B$, we have $\mathrm{f}(A \cup \{x\}) - \mathrm{f}(A) \geq \mathrm{f}(B \cup \{x\}) - \mathrm{f}(B)$. We define a ground set $V$ containing elements $x_{n_j, f_l}$, which denotes the placement of file $f_l$ into the cache of helper $n_j$. Let $\psi \subset V$ denote the placement set corresponding to matrix $\Omega$ such that $x_{n_j, f_l} \in \psi$ if and only if $\omega_{f_l, n_j} = 1$. We consider the set function $G^t_{u_i}(\psi) \triangleq \phi_{MBS} - \overline{\Psi}^t_{u_i}$ and prove the submodularity of $G^t_{u_i}$. For this purpose, we will show that if the placement set $\psi$ becomes larger, the marginal gain value (the amount of increase in $G^t_{u_i}$) of adding a new file to a helper decreases. We consider two placement sets, $\psi$ and $\psi'$, where $\psi \subset \psi' \subset V$. Consider adding a file $f_l$ to both placement sets in helper $n_j$ and focusing on user request $S_v$, which contains $f_l$. Let $\Delta\psi' \triangleq G^t_{u_i}(\psi' \cup \{x_{n_j, f_l}\}) - G^t_{u_i}(\psi')$ and $\Delta\psi \triangleq G^t_{u_i}(\psi \cup \{x_{n_j, f_l}\}) - G^t_{u_i}(\psi)$. Also, $h_{I(m, u_i)}$ and $h_{I(n, u_i)}$ denote the helper that is able to retrieve at least one file within request $S_v$ with the least delay, for the file placement $\psi$ and $\psi'$, respectively. Since $\psi \subset \psi'$, for file placement, $\psi'$ more files are cached in the helpers. Hence, the user $u_i$ can retrieve the request with less delay. Consequently, we infer that $\phi_{n_{I(m, u_i)}} \geq \phi_{n_{I(n, u_i)}}$.

10.22060/EEJ.2024.22977.5579

By adding $f_l$ to both placement sets in helper $n_j$, two cases are considerable:

1) $\phi_{n_{I(n,u_i)}} < \phi_{n_{I(j,u_i)}}$: we deduce that according to placement $\psi'$, user $u_i$ can download the request from a helper with less delay. Hence, $\Delta\psi' = 0$. If $\phi_{n_{I(m,u_i)}} < \phi_{n_{I(j,u_i)}}$, we also deduce $\Delta\psi = 0$. Otherwise, by adding $f_l$ to file placement $\psi$, the request can be downloaded with less delay from helper $n_j$. In this case, $\Delta\psi = (\phi_{n_{I(j,u_i)}} - \phi_{n_{I(m,u_i)}}) P_r(S_v)$. Clearly, $\Delta\psi \geq \Delta\psi'$.

2) $\phi_{n_{I(n,u_i)}} > \phi_{n_{I(j,u_i)}}$: Hence, the request can be downloaded from helper $n_j$ with less delay. As a result, $\Delta\psi' = (\phi_{n_{I(j,u_i)}} - \phi_{n_{I(n,u_i)}}) P_r(S_v)$. Similarly, we have $\Delta\psi = (\phi_{n_{I(j,u_i)}} - \phi_{n_{I(m,u_i)}}) P_r(S_v)$. Since $\phi_{n_{I(m,u_i)}} \geq \phi_{n_{I(n,u_i)}}$, we deduce that $\Delta\psi \geq \Delta\psi'$.

Hence, we have proved the submodularity of $G^t_{u_i}$. Since the objective function of Problem 1 is a summation of $G^t_{u_i}$, the proof is complete.

Since the objective function of Problem 1 is monotone submodular and the constraints are in the form of a matroid, it can be proved that the greedy algorithm performs within a constant factor of $\frac{1}{2}$ as well as the optimal solution [24, 25]. Consequently, we have the following lemma:

**Lemma 2.** *The proposed greedy algorithm achieves a* $\frac{1}{2}$*-approximation solution for Problem 1.*

The greedy algorithm starts with an empty cache placement set ($A = \varnothing$). At each iteration, a file that maximizes the marginal gain and satisfies the capacity constraints of helpers is added to the placement set. In other words, at each iteration, a feasible file placement that maximizes the reduction in total average delay compared to the previous iteration is selected and added to the placement set. In the beginning, no files are cached in the helpers, and the total delivery delay equals the total delay in downloading the requested files from MBS. At each iteration, caching a file in a helper reduces the total delivery delay since

10.22060/EEJ.2024.22977.5579

some users can receive their request with less delay from the helpers instead of the MBS. The greedy algorithm continues until all the caches are filled. Algorithm 1 summarizes our proposed greedy algorithm. In this algorithm, $\mathcal{F}(A)$ denotes the feasible set of elements (files) which can be added to placement set $A$, while satisfying the helper cache size constraints.

**Algorithm 1** – Greedy algorithm for proposed system model

**Input:** $V$
**Output:** $A$
**Initialize**
$A \leftarrow \emptyset$
**while** CacheContent($A$, $n_j$) $\leq C$ for some $j$ **do**
  $x^* \leftarrow \underset{x \in F(A)}{argmax} \sum_{u=1}^{U} \left( G^t_{u_i}(A \cup \{x\}) - G^t_{u_i}(A) \right)$
  $A \leftarrow A \cup \{x^*\}$
**end while**
function CACHECONTENT($A$, $n_j$)
  z ← 0
  **for all** $x_{k,fl} \in A$ **do**
    **if** $k = n_j$ **then**
      $z \leftarrow z + 1$
    **end if**
  **end for**
  **return** $z$
**end function**

The greedy algorithm is continuous for most $CH$ iterations, and in each iteration, at most $FH$ file placements are compared to find the highest marginal gain values. Moreover, computing each marginal gain requires evaluating the delivery delay for $U$ users. Consequently, the complexity of the greedy algorithm for Problem 1 is $\mathcal{O}(CH^2FU)$. On the other hand, the exhaustive search algorithm explores all the file placements, which are $\binom{F}{C}^H$ cases. Hence, the complexity of the exhaustive search is $\mathcal{O}\left(\binom{F}{C}^H U\right)$, which grows exponentially with the number of helpers. Consequently, the greedy algorithm is an effective solution that provides a good solution with affordable complexity.

10.22060/EEJ.2024.22977.5579

It is worth noting that pipage rounding is another algorithm for maximizing a monotone submodular problem subject to matroid constraints [24]. This algorithm converts the integral problem into a continuous problem using multilinear extension at the first step. Then, the fractional solution is converted to an integer solution using the pipage rounding technique. Although this algorithm is proven to achieve a $(1-\frac{1}{e})$-approximation which is slightly better than the greedy algorithm, its computational complexity is very high which makes it intractable especially when the size of the network grows. To be more specific, the computational complexity of the pipage algorithm is $\mathcal{O}(HF)^8$. As a result, the greedy algorithm is more suitable for our problem since the size of network is typically large.

## 5- Numerical Results

This section illustrates our system model and evaluates our proposed algorithm. The key parameters for our simulations are summarized in Table 2. In Fig. 2, we compare the network performance of both the greedy and exhaustive search algorithms. When each helper's storage capacity is one ($C=1$), we infer that the performance of the exhaustive algorithm is notably better than the greedy algorithm, which is achieved at the expense of much more complexity. However, as the storage capacity increases ($C=2$), the performance of both algorithms is very similar. Hence, in the scenario where each storage can store more than one file (which is a typical assumption), a greedy algorithm can achieve almost the same performance with much lower computational complexity. This is very important when the size of the network grows and the computational complexity of an exhaustive algorithm is so high, which makes it intractable.

10.22060/EEJ.2024.22977.5579

Table 2. Simulation Parameters

| Parameters | Values |
|---|---|
| Side lengths of square | 100 m |
| Radius of a helper's transmission region | 50 m |
| Location of base station | [200,200] m |
| Base station's transmission power | 10 W |
| Helper's transmission power | 5 W |
| Path loss exponent ( $\kappa$) | 2 |
| Zipf parameter ( $\lambda$) | 22 |
| $d_{max}$ | 50 m |

We also compare the performance of our proposed greedy algorithm with that of random and popularity caching methods. In random caching, $C$ files have been randomly selected and stored in each helper. In popularity caching, the most $C$ popular files have been placed in each helper [19]. As illustrated in Fig. 3, it can be deduced that a greedy algorithm can achieve almost optimal cache performance. Moreover, there is a big gap between the performance of greedy algorithms and the performance of random and popularity algorithms.



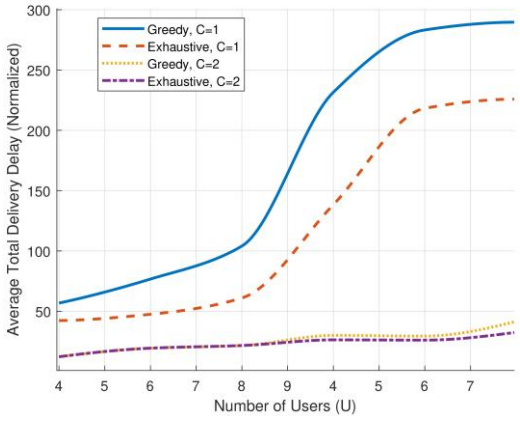


Fig. 2. The comparison between Exhaustive search and Greedy algorithms. $F$ = 5, $H$ = 3, $K$ = 2

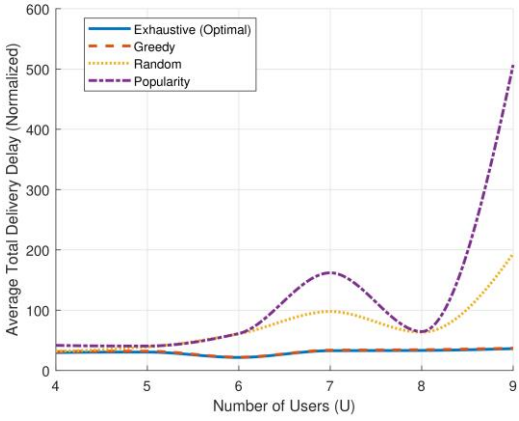


Fig. 3. The comparison between exhaustive search, greedy, random, and popularity algorithms. $F$ = 5, $H$ = 4, $K$ = 2

Now, we will consider the effect of each element in the network and analyze its performance for the greedy algorithm. The first factor is the number of users in the network. Fig. 4 shows the network total average

10.22060/EEJ.2024.22977.5579

delivery delay as the number of users increases. This is because we consider the sum of the delivery delays for all users.

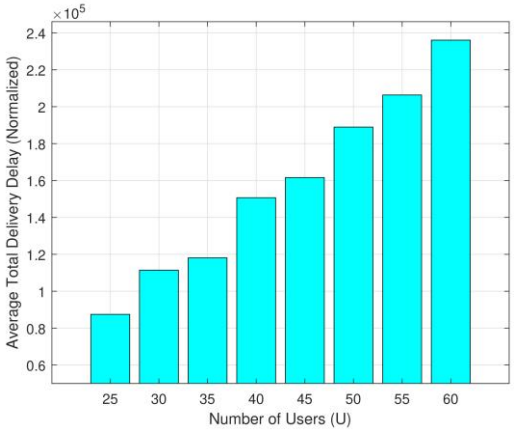


Fig. 4. The effect of increasing the number of users on average total delivery delay. $F$ = 10, $H$ = 20, C = 1, $K$ = 2

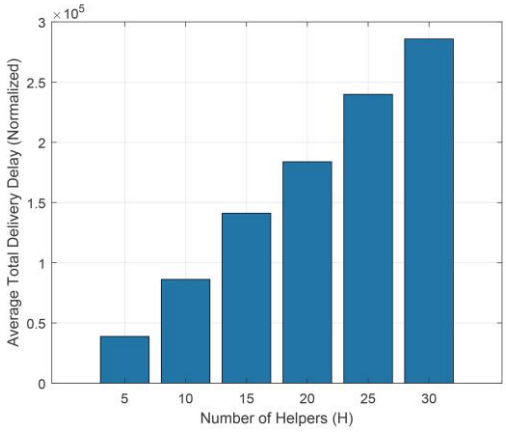


Fig. 5. The effect of increasing the number of helpers on average total delivery delay. $F$ = 10, $U$ = 50, $C$ = 1, $K$ = 2

Another factor to consider is the number of helpers. Fig. 5 is based on the increase in the number of helpers in the network. Increasing the number of helpers will introduce more interference for the users which are in the transmission range of multiple helpers. Hence, the delay increases with the number of helpers. On the other hand, if we consider a noise-limited system and ignore the interference among helpers, by increasing the number of helpers, users will be connected to more helpers, which leads to an increase in the probability of the files being received by users. Therefore, users are not required to request their files from MBS. In this case, delay decreases as the number of helpers increases (Fig. 6). It is worth noting that the delay in the noise-limited scenario is lower since we neglect the interference from other helpers and only consider the noise, which contributes to a higher data rate and a lower delivery delay.

10.22060/EEJ.2024.22977.5579

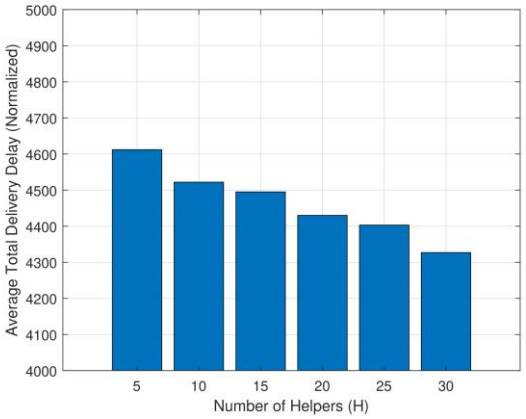


Fig. 6. The effect of increasing the number of helpers on average total delivery delay (no interference)

$F = 10$, $U = 50$, $C = 1$, $K = 2$

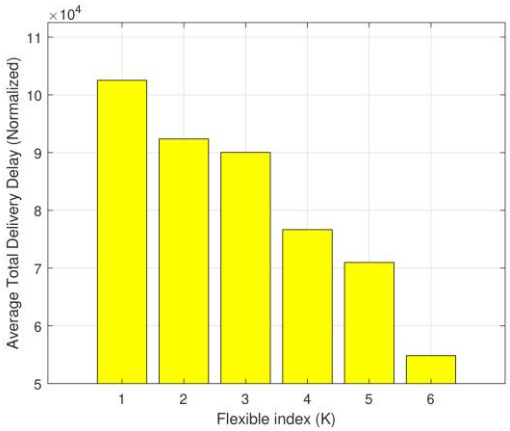


Fig. 7. The effect of increasing the flexible index on average total delivery delay. F = 10, H = 15, U = 40, C = 1

We also assess the impact of the flexible index on the average total delivery delay in Fig. 7. It is worth noting that requesting multiple files from the network will incur less delay. That is because the probability of the requested files being stored in the helpers increases. Consequently, the concept of flexible users contributes to a lower delivery delay.

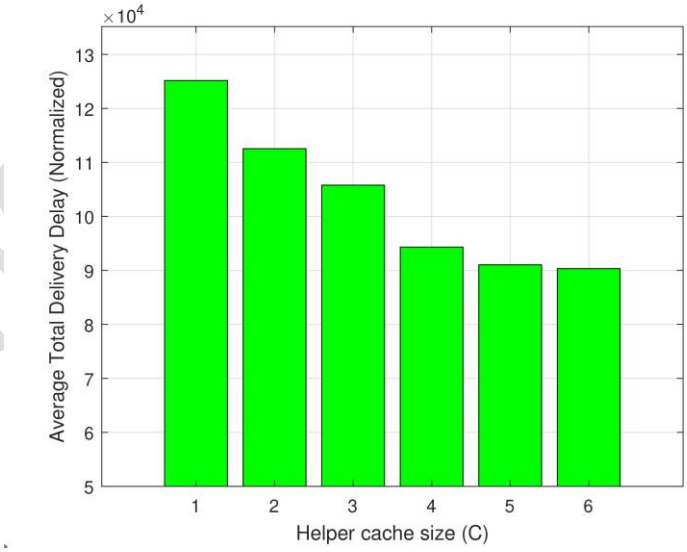


Fig. 8: The effect of increasing helper cache size on average total delivery delay.

F = 10, H = 15, U = 40, K = 2.

10.22060/EEJ.2024.22977.5579

Fig. 8 illustrates the effect of helper cache size on the average total delivery delay. As expected, by increasing the helper's cache size, the opportunity to download a file from a helper instead of a base station increases. Consequently, the average total delivery delay decreases as the helper's cache size increases.

## 6- Conclusion

In this paper, we address the problem of finding the optimal cache placement in wireless heterogeneous networks. We proposed an efficient approximated algorithm (greedy) that achieves provable performance with low computational complexity. Simulation results showed that the greedy algorithm achieves similar performance compared to the exhaustive search algorithm when the helpers' storage capacity is greater than one. When each helper's storage capacity is one, the performance of exhaustive search is notably better at the expense of much more computational complexity. We also investigated the effect of the number of helpers on the total delivery delay. In a noise-limited system, increasing the number of helpers results in a lower delivery delay as there is more opportunity to download the requested file from a nearby helper rather than from the base station. On the other hand, for an interference-limited system, increasing the number of helpers may contribute to a larger delivery delay due to their potential for increasing interference. Simulation results showed that the flexibility of users can significantly reduce the delivery delay. For example, the total delivery delay is reduced by approximately 50% when the flexible index changes from 1 to 6.

10.22060/EEJ.2024.22977.5579

**7- Nomenclature**

Below is a list of English symbols followed by Greek symbols used in this paper.

| | |
|---|---|
| $C$ | Helper cache size |
| $F$ | Number of files |
| $H$ | Number of helpers |
| $K$ | Flexible index |
| $\mathscr{L}$ | Library of the requested files |
| $P_r(S_v)$ | Probability mass function of a flexible request $S_v$ |
| $S_v$ | A user's flexible request (preferred subset) |
| $U$ | Number of users |

***Greek symbols***

| | |
|---|---|
| $\Lambda(K)$ | Set of flexible requests containing K files |
| $\phi_{n_j,u_i}$ | Delay between helper $(n_j)$ and user $(u_i)$ |
| $\omega_{f_l,n_j}$ | Indicator result representing the caching of file $f_l$ by helper $n_j$ |

**References**

[1] M. Ghaznavi, E. Jalalpour, M.A. Salahuddin, R. Boutaba, D. Migault, S.J.I.C.S. Preda, Tutorials, Content delivery network security: A survey, 23(4) (2021) 2166-2190.
[2] G. Pan, S. Xu, S. Zhang, X. Chen, Y.J.I.T.o.C. Sun, S.f.V. Technology, Quality of Experience Oriented Cross-layer Optimization for Real-time XR Video Transmission, (2024).
[3] C. Madapatha, B. Makki, A. Muhammad, E. Dahlman, M.-S. Alouini, T.J.I.O.J.o.t.C.S. Svensson, On topology optimization and routing in integrated access and backhaul networks: A genetic algorithm-based approach, 2 (2021) 2273-2291.
[4] T. Kimura, T. Kimura, A. Matsumoto, K.J.I.A. Yamagishi, Balancing quality of experience and traffic volume in adaptive bitrate streaming, 9 (2021) 15530-15547.
[5] Y. Li, X. Zhang, C. Cui, S. Wang, S.J.I.T.o.C. Ma, S.f.V. Technology, Fleet: Improving quality of experience for low-latency live video streaming, 33(9) (2023) 5242-5256.
[6] T. Lyko, M. Broadbent, N. Race, M. Nilsson, P. Farrow, S.J.M.T. Appleby, Applications, Improving quality of experience in adaptive low latency live streaming, 83(6) (2024) 15957-15983.
[7] G. Hasslinger, M. Okhovatzadeh, K. Ntougias, F. Hasslinger, O.J.C.N. Hohlfeld, An overview of analysis methods and evaluation results for caching strategies, 228 (2023) 109583.
[8] H. Li, M. Sun, F. Xia, X. Xu, M.J.T.S. Bilal, Technology, A survey of edge caching: Key issues and challenges, 29(3) (2023) 818-842.
[9] K. Shanmugam, N. Golrezaei, A.G. Dimakis, A.F. Molisch, G.J.I.T.o.I.T. Caire, Femtocaching: Wireless content delivery through distributed caching helpers, 59(12) (2013) 8402-8413.
[10] J. Yao, T. Han, N.J.I.C.S. Ansari, Tutorials, On mobile edge caching, 21(3) (2019) 2525-2553.
[11] Y. Cao, S. Maghsudi, T. Ohtsuki, T.Q.J.I.T.o.C. Quek, Mobility-aware routing and caching in small cell networks using federated learning, (2023).
[12] M.K. Somesula, S.K. Mothku, S.C.J.I.S.J. Annadanam, Cooperative service placement and request routing in mobile edge networks for latency-sensitive applications, 17(3) (2023) 4050-4061.
[13] M.K. Somesula, R.R. Rout, D.V.J.A.H.N. Somayajulu, Greedy cooperative cache placement for mobile edge networks with user preferences prediction and adaptive clustering, 140 (2023) 103051.
[14] R. Wang, J. Zhang, S. Song, K.B.J.I.T.o.W.C. Letaief, Mobility-aware caching in D2D networks, 16(8) (2017) 5001-5015.
[15] G. Ma, Z. Wang, M. Zhang, J. Ye, M. Chen, W.J.I.J.o.S.A.i.C. Zhu, Understanding performance of edge content caching for mobile video streaming, 35(5) (2017) 1076-1089.
[16] M. Song, H. Shan, Y. Fu, H.H. Yang, F. Hou, W. Wang, T.Q.J.I.T.o.W.C. Quek, Joint user-side recommendation and D2D-assisted offloading for cache-enabled cellular networks with mobility consideration, 22(11) (2023) 8080-8095.
[17] S.K.u. Zaman, T. Maqsood, F. Rehman, S. Mustafa, M.A. Khan, N. Gohar, A.D. Algarni, H.J.E. Elmannai, Content caching in mobile edge computing based on user location and preferences using cosine similarity and collaborative filtering, 12(2) (2023) 284.
[18] I. Avgouleas, N. Pappas, V.J.E.J.o.W.C. Angelakis, Networking, A wireless caching helper system with heterogeneous traffic and random availability, 2021(1) (2021) 69.
[19] M. Chen, Y. Hao, L. Hu, K. Huang, V.K.J.I.T.o.W.C. Lau, Green and mobility-aware caching in 5G networks, 16(12) (2017) 8347-8361.
[20] M. Sheraz, S. Shafique, S. Imran, M. Asif, R. Ullah, M. Ibrar, A. Bartoszewicz, S.J.E. Mobayen, Mobility-Aware Data Caching to Improve D2D Communications in Heterogeneous Networks, 11(21) (2022) 3434.

[21] G. Shan, Q.J.I.A. Zhu, Sociality and mobility-based caching strategy for device-to-device communications underlying heterogeneous networks, 7 (2019) 53777-53791.
[22] P. Eslami, M.H. Amerimehr, S.P.J.I.A. Shariatpanahi, A new framework for mobile edge caching by proposing flexible user in heterogeneous cellular networks, 8 (2020) 188938-188950.
[23] S. Anokye, D. Ayepah-Mensah, A.M. Seid, G.O. Boateng, G.J.I.S.J. Sun, Deep reinforcement learning-based mobility-aware UAV content caching and placement in mobile edge networks, 16(1) (2021) 275-286.
[24] G. Calinescu, C. Chekuri, M. Pal, J.J.S.J.o.C. Vondrák, Maximizing a monotone submodular function subject to a matroid constraint, 40(6) (2011) 1740-1766.
[25] M.L. Fisher, G.L. Nemhauser, L.A. Wolsey, An analysis of approximations for maximizing submodular set functions—II, Springer, 1978.