electronics

MDPI

Article

DCEC: D2D-Enabled Cost-Aware Cooperative Caching in MEC Networks

Jingyan Wu [1,2], Jiawei Zhang [1] and Yuefeng Ji [1,*]

[1] State Key Laboratory of Information Photonics and Optical Communications, Beijing University of Posts and Telecommunications, Beijing 100876, China; wujy@bupt.edu.cn (J.W.); zjw@bupt.edu.cn (J.Z.)
[2] College of Information Engineering, Henan University of Science and Technology, Luoyang 471023, China
* Correspondence: jyf@bupt.edu.cn

Abstract: Various kinds of powerful intelligent mobile devices (MDs) need to access multimedia content anytime and anywhere, which places enormous pressure on mobile wireless networks. Fetching content from remote sources may introduce overly long accessing delays, which will result in a poor quality of experience (QoE). In this article, we considered the advantages of combining mobile/multi-access edge computing (MEC) with device-to-device (D2D) technologies. We propose a D2D-enabled cooperative edge caching (DCEC) architecture to reduce the delay of accessing content. We designed the DCEC caching management scheme through the maximization of a monotone submodular function under matroid constraints. The DCEC scheme includes a proactive cache placement algorithm and a reactive cache replacement algorithm. Thus, we obtained an optimal content caching and content update, which minimized the average delay cost of fetching content files. Finally, simulations compared the DCEC network architecture with the MEC and D2D networks and the DCEC caching management scheme with the least-frequently used and least-recently used scheme. The numerical results verified that the proposed DCEC scheme was effective at improving the cache hit ratio and the average delay cost. Therefore, the users' QoE was improved.

Keywords: edge caching; cooperative caching; D2D-enabled; MEC; delay cost

Citation: Wu, J.; Zhang, J.; Ji, Y. DCEC: D2D-Enabled Cost-Aware Cooperative Caching in MEC Networks. *Electronics* **2023**, *12*, 1974. https://doi.org/10.3390/electronics12091974

Academic Editors: Peiying Zhang, Haotong Cao and Keping Yu

Received: 25 March 2023
Revised: 21 April 2023
Accepted: 21 April 2023
Published: 24 April 2023

1. Introduction

With the explosive growth of intelligent mobile devices (MDs) on networks, new applications are constantly emerging. A huge amount of multimedia content is generated and spread throughout the mobile networks every day. First, the demand for multimedia content in mobile wireless networks is exploding. More and more mobile users (MUs) would like to view this content online. Second, some of the contents have large sizes due to their high-definition. It is obvious that the demand for the ever-increasing quality and quantity of multimedia content poses unprecedented challenges to mobile networks. Accessing these contents would consume multiple network resources, e.g., transmit power, bandwidth, and backhaul links. Especially in the same region, different MUs with different kinds of MDs may require some of the same popular contents in the same period, which may even lead to network congestion.

To overcome these challenges, mobile/multi-access edge computing (MEC) is proposed, taking advantage of computing, communication, and caching resources at the edge of networks [1–7] over hybrid fiber–wireless (FiWi) access networks [8,9]. Specifically, by proactively caching more popular content in advance, edge caching has the potential to reduce the content accessing latency, backhaul links' usage, and cloud computing load [10]. In this way, popular content is cached at base stations (BSs) or access points (APs) at off-peak times. MUs' requests for the same content can be satisfied easily without retransmitting the content from remote cloud servers via backhaul links [11,12].

In fact, a BS's or an AP's caching capacity is usually not unlimited, which might degrade the network performance in some way [13,14]. To this end, device-to-device

(D2D) communications are expected to enhance the abilities of the networks [1,15–18]. Note that a large number of MDs are widely distributed in the current mobile networks. The caching capacity and computing resources of these MDs are ubiquitous, as well as the edge servers [19]. In some cases, besides caching at the BSs, more popular content files may be cached at the MDs and can be requested by more MUs conveniently [20]. D2D communications are usually free or inexpensive and can obtain efficient spectrum utilization. Caching at the MDs leverages the powerful storage capacity of MDs, which is probably the lowest-cost and fastest-growing network resource [15]. It is also important to know that D2D communication can use the traditional cellular band and the unlicensed mmWave band [15,21] to transmit data between MDs. These processes do not require the participation of intermediate nodes. It can be said that D2D communications extend the coverage of traditional cellular networks, as well as provide a solution to the scare spectrum resources of cellular networks [22]. At present, content files' distributed caching at BSs and MDs is regarded as a promising solution [23]. Combining D2D communication with edge caching is recognized as an effective technique in mobile wireless networks [16,20,24].

Nevertheless, having separate caching scheme designs cannot make good use of caching resources. In order to meet this challenge, cooperative caching schemes have been designed to improve the performance of networks [19,25–29]. It has been proven that cooperative caching is an effective method to reduce the cost of content accessing delay and the traffic burden on mobile wireless networks [25,30,31]. Moreover, most requests are highly latency-sensitive, and users would like to turn off the slow and poorly performing applications [10]. Therefore, the maximum delay tolerance for the content caching strategy should be considered.

Therefore, in this paper, we considered the optimal content caching management problem in D2D-enabled cooperative edge caching (DCEC). A strategy of collaborative caching among MDs and edge servers in the hierarchical MEC networks is proposed. By leveraging the maximization of a monotone submodular function under matroid constraints, we minimized the average delay cost at the caching nodes for optimal performance. Our main contributions are summarized as follows:

- We introduce the DCEC network architecture combining the advantages of the MEC and D2D technologies. Each requested content file can be served cooperatively by the caches at the MDs, by the caches at BSs, or the content library in central cloud servers.
- We propose the average delay cost model for requested content files by MUs and formulated the content caching optimization problem. We proved that this is the problem of maximizing a monotone submodular function under matroid constraints. To solve such a problem, we adopted the proactive cache placement algorithm and reactive cache replacement algorithm, aiming to minimize the average delay cost.
- Extensive simulations compared the DCEC network architecture with the MEC and D2D networks and the caching management scheme with the least-frequently used and the least-recently used scheme. The numerical results verified that the proposed DCEC architecture and scheme were effective at improving the cache hit ratio and reducing the average delay cost.

The organization of this paper is as follows. Section 2 reviews the related works. Section 3 represents the system model, including the DCEC network model, parameter settings, cooperative caching model, and delay cost model. Section 4 formulates the content caching optimization problem and makes the transformation. Then, we developed an efficient caching management scheme to solve this problem. Related simulation results are provided in Section 5. Finally, Section 6 concludes this paper.

2. Related Work

Edge caching can reduce access latency without duplicate transmissions from a remote central cloud. It has been widely studied from different aspects. Among them, there are various caching schemes enhancing different performances of the proposed networks. Tran et al. [11,32] pointed out that MEC has the ability of edge caching and edge processing.

The higher bit-rate versions of the requested content could be transcoded to the lower bit-rate ones. With transcoding, edge servers need not cache all the resolution versions of the content. The implementation of edge processing improves the efficiency of edge caching. Chiang et al. [33] considered partial caching, which only caches some segments of the popular content or a few resolution versions of the same content file, which could cache more content at the edge nodes while ensuring users' QoE. Liu et al. [9] jointly considered the optimal server placement, as well as content caching in MEC networks to minimize the average delay and the usage of bandwidth. Another potential scheme is cooperative caching, which improves the cache hit ratio by adopting cache resources at distributed edge nodes [33]. Bhar et al. [34] studied edge caching by designing opportunistic caching schemes to achieve high energy efficiency. Nevertheless, all of the above works focused on two-tier edge computing networks.

Some works considered different performance parameters. Li et al. [35] considered the various sizes of caching content and formulated the problem of maximizing the cache hit ratio of edge caching and minimizing the average delay cost of fetching content files. Due to the limitations of storage and costs, not all of the content files are able to be cached in the edge servers, and it is critical to prioritize the files before deciding which are worth caching [36]. In these above works, they did not consider the cooperation among MDs.

Some other works were based on different network architectures. Tran et al. [30] designed a cooperative caching structure in cloud radio access networks (C-RANs). The authors in [23] proposed a cooperative caching scheme, where caching resources were provided jointly by macro base stations and small base stations. Wang et al. [27] considered the optimal cooperative caching problem among edge servers in FiWi access networks, aiming to minimize the average latency cost. They did not leverage the ubiquitous caching capacity of MDs throughout the edge networks.

Therefore, it is very important to figure out the right caching strategy in three-tier cooperative caching MEC networks. Wang et al. [37] considered D2D-enabled cooperative edge caching and jointly optimized the selection of nodes and cache replacement with federated learning considering the computing and storage capacity limitations. Zhang et al. [38] considered the inter-tier (across different tiers) and intra-tier (within each tier) collaborative hierarchical caching in MEC networks and derived the maximum D2D pair number. Edge servers cooperatively cached to maximize the caching gain. The inter-tier caching minimized the average hop number, whereas the intra-tier caching maximized the cache hit ratio. Li et al. [21] investigated the hierarchical edge caching in both MDs and BSs considering the different cache sizes, the social behaviors, and the preference of the MUs. They aimed to reduce the average traffic load of the content requests.

However, the above works did not consider the maximum delay tolerance for caching strategies. Some works have been performed on this issue [28,39]. M.K. Somesula et al. designed caching methods considering deadlines, user mobility, etc. [29]. Due to the limited caching capabilities, the edge nodes could not cache all the content. Since new content is continually generated and the popularity of the content changes over time [26], it is necessary to update the cached content adaptively. Therefore, in this paper, we propose the DCEC architecture in the hierarchical three-tier MEC networks. We considered the cooperative edge caching among BSs and MDs with the constraint of the maximum delay tolerance.

3. System Model

In this section, Figure 1 illustrates the outline of the following two sections, the system model and DCEC cache management strategy. In the system model, we first introduced the proposed DCEC network architecture and present the DCEC system model and cooperative caching model in detail. We analyze four ways of fetching content files, which are local caching, D2D caching, edge caching, and cloud serves. Then, we obtained the delay cost model. In the section on the DCEC cache management strategy, we first formulated an optimal caching problem. Then, we transformed this optimization problem and solved it

with the maximization of the monotone submodular function under matroid constraints. Furthermore, we designed a DCEC cache management scheme consisting of a proactive cache placement algorithm and a reactive cache replacement algorithm. Based on the DCEC, we obtained the optimal cache placement and replacement.

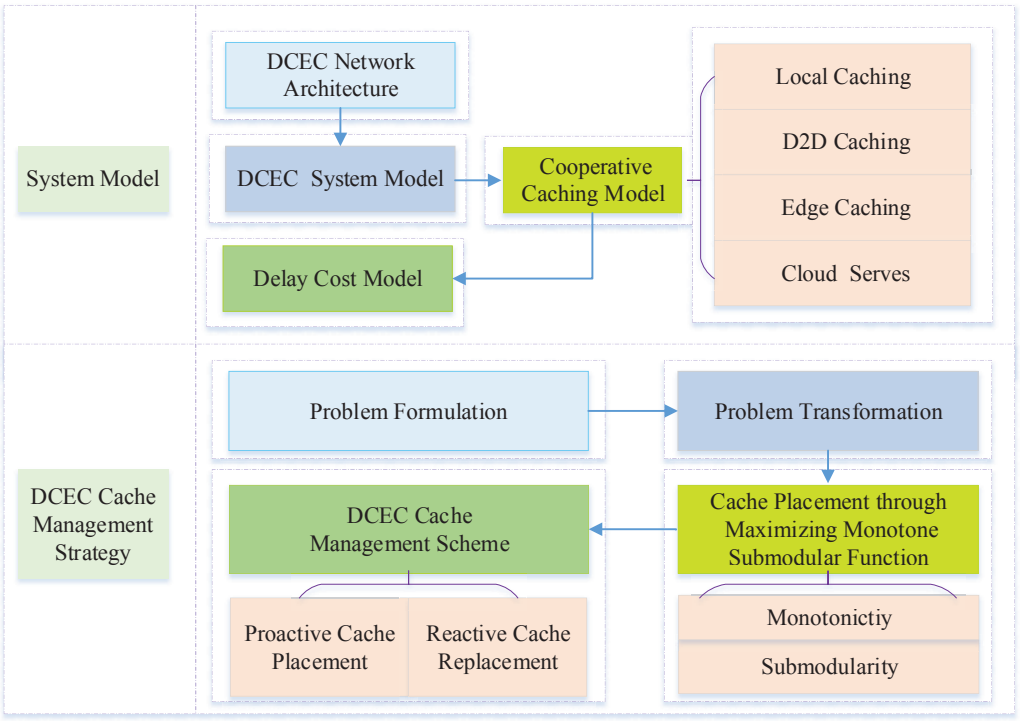

Figure 1. The outline of the system model and DCEC cache management strategy sections.

3.1. DCEC Network Architecture

As illustrated in Figure 2, we present a D2D-enabled cooperative edge caching architecture, which includes three tiers of heterogeneous nodes, i.e., MDs, BSs, and content library, in the remote central cloud. The cooperative edge caching networks that we considered consisted of N MDs. Each MD has a limited computing capacity and storage capacity and the capability of D2D communication. They can communicate directly with their neighboring MDs through WiFi or Bluetooth [1,20,21] and can be connected to the closest BS via low-latency wireless links. In this way, they can share content files with each other.

In the edge tier, each BS is furnished with several edge servers and can be connected to the cloud tier via high-bandwidth backhaul links. Optical backhauling is an efficient way [40,41] to offer high throughput and low latency while ensuring low energy consumption. In addition, with artificial-intelligence (AI)-driven autonomous optical networks [42], the flexibility and reliability of the networks can be effectively enhanced. In the meantime, the powerful computing capability of the edge servers can be applied to calculate the popularity of the content in the content library. By leveraging AI and/or other means [43–47], we can estimate the popularity of the content and be aware of what content is more popular or less popular in the content library; we can estimate what content people prefer, what content they seek, and even their profiles [30]. As the edge caching nodes, the edge servers at the BSs are able to perform the corresponding content cache management.

To simplify the model, we considered one BS in the following derivation. In practice, neighboring BSs equipped with limited caching capacity can cooperate with each other and share the cached popular content files through optical fibers. The service range of this particular BS covers all the MDs. The remote central cloud is equipped with many cloud content servers, which own a content library. The overall storage capacity of the cloud servers is large enough to store all the content. Content providers can use the edge caching

resources of MEC networks to carry the requested content near the MUs. We assumed that all the requested content files can be served within the coverage area of the BS. The FiWi access technology guarantees the connection throughout the MEC networks.

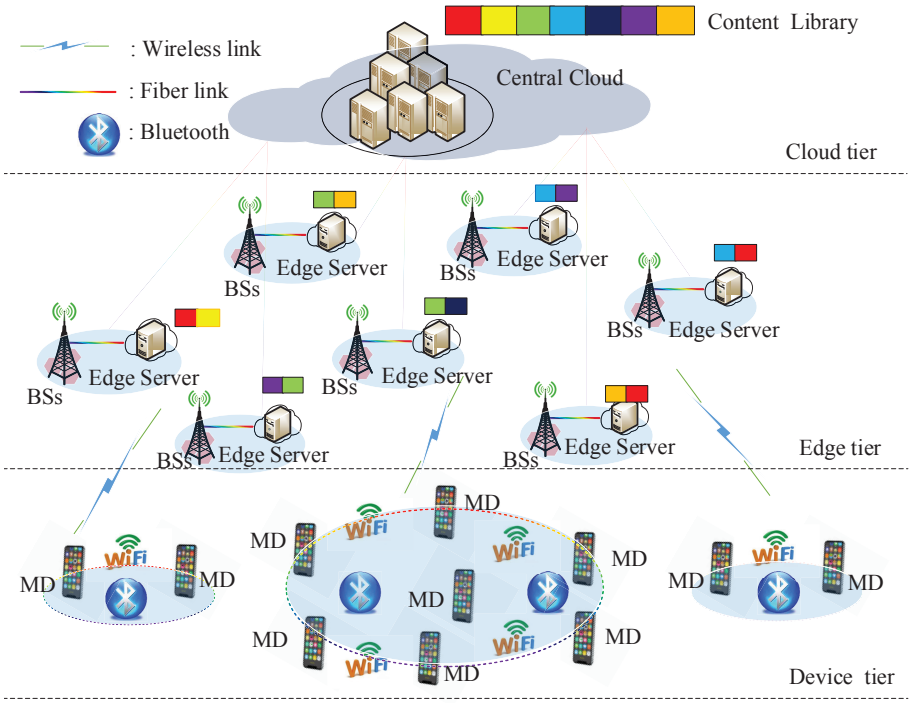

Figure 2. DCEC network architecture.

In our DCEC networks, the popular content may be cached in the MDs in the device tier or in the edge servers in the edge tier. We assumed that the copies of the same content file can be pre-stored at multiple caching nodes, whereas only one copy can be stored at the same caching nodes. MDs can help each other when they own the corresponding requested content. The helpers may be rewarded with accumulated reward points (ARPs). MDs with more ARPs have higher priority for being served. On the one hand, the BS can collect essential information about the D2D connectivity through device discovery. On the other hand, MDs themselves can actively report information [48]. If the network situation is undesirable and the requested content file is delay-sensitive, MDs would like to choose D2D cooperation due to the short communication delay.

The cooperative caching process is shown in Figure 3. Each MU can send content requests, including information about the MD, i.e., its location and geographic mobility pattern, and the properties of the content files requested, such as the content ID [21], the size, the popularity [35], and so on. According to this information, the BSs have the ability to decide where the content requests are served. The requested content file of MD n can be fetched from MD itself and from neighboring MDs via D2D cooperative caching, or from the edge server via edge caching, or further from the content library in the remote cloud servers [21].

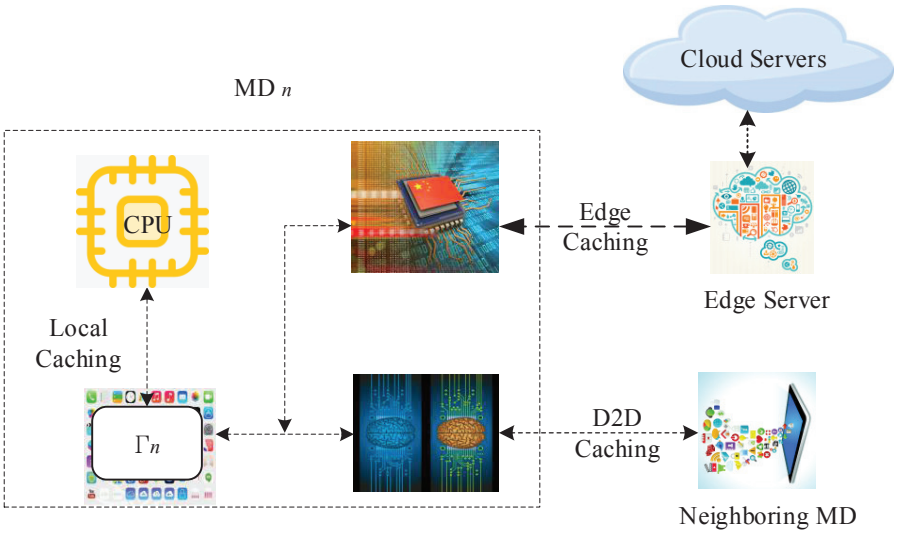

Figure 3. Cooperative caching.

3.2. DCEC System Model

As depicted in Figure 2, N MDs are labeled as the set $\mathcal{N} = \{1, 2, \ldots, N\}$. We assumed that all fragments of thecontent files in the content library in the cloud have an equal size. Accordingly, $\mathcal{I} = \{1, 2, \ldots, I\}$ denote the index set of all available content files requested by the MUs. Thus, the content file i that is requested by MU n is described as $\Gamma_n^i = (T_n^i, P_n^i, U_n^i)$, $i \in \mathcal{I}, n \in \mathcal{N}$, where, T_n^i stands for the delay cost of fetching Γ_n^i, P_n^i stands for the popularity of Γ_n^i, and U_n^i stands for the user with MD n who requests Γ_n^i.

The popularity of the content files can be estimated in advance. We denote the popularity of content file i as $P^i \in [0, 1]$, $i \in \mathcal{I}$. It was assumed that the content popularity changes slowly during a relatively long period at the cell level. Thus, $P_n^i = P^i, n \in \mathcal{N}$. In practice, the same content can have different popularities at different locations [11]. The user who holds the MD n is labeled as $U_n, n \in \mathcal{N}$. We assumed that the MUs in the DCEC networks obtain service from the closest BS with the maximum signal intensity. All MUs who are likely to send content requests are denoted as $\mathcal{U} = \{U_1, U_2, \ldots, U_N\}$. We also denote the maximum delay tolerance for any of the content files as T_{max}.

In our DCEC networks, the storage capacity of distributed MDs is denoted as S_n, $n \in \mathcal{N}$, and the cache set that caches the content files at MD n is denoted as $\mathbb{X}_n, n \in \mathcal{N}$. Similarly, the storage capacity of the typical BS is labeled as S_{N+1}, and the cache set at the BS is labeled as $\mathbb{X}_{N+1}$. Generally, the storage capacity and caching capacity of the BS are much higher than that of the MD. Thus, $S_{N+1} \gg S_n$, $|\mathbb{X}_{N+1}| \gg |\mathbb{X}_n|, n \in \mathcal{N}$. In particular, a feasible cache placement decision set is denoted as $\mathbb{X} = \{\mathbb{X}_1, \mathbb{X}_2, \ldots, \mathbb{X}_N, \mathbb{X}_{N+1}\}$, which must satisfy the constraints of the storage capacity, i.e., $|\mathbb{X}_n| \leq S_n, n \in \mathcal{N} \cup \{N+1\}$.

The actual caching capacity increases linearly with the number of MDs throughout the network [15]. Though the limited caching resources of a single MD, the accumulative caching capacity gathered from many MDs is big enough to storing the content [26]. For the purpose of clearly describing the content cache decisions, the basic cache placement set [30] is defined as follows:

$$\mathbb{P} = \{f_1^1, f_1^2, \ldots, f_1^I, \ldots, f_{N+1}^1, f_{N+1}^2, \ldots, f_{N+1}^I\}, \tag{1}$$

where f_n^i denotes the content file i, which is cached in MD n, while f_{N+1}^i indicates the file i, which is cached in the BS. The finite basic set $\mathbb{P}$ can be divided into $N+1$ disjoint subsets, $\mathbb{P}_1, \mathbb{P}_2, \ldots, \mathbb{P}_N, \mathbb{P}_{N+1}$. The cache placement subset $\mathbb{P}_n = \{f_n^1, f_n^2, \ldots, f_n^I\}$ represents the set of all content files that might be cached in MD n. $\mathbb{P}_{N+1} = \{f_{N+1}^1, f_{N+1}^2, \ldots, f_{N+1}^I\}$ represents the set of all content files that might be cached in the BS. Thus, $\mathbb{P} = \{\mathbb{P}_1, \mathbb{P}_2, \ldots, \mathbb{P}_N, \mathbb{P}_{N+1}\}$ indicates the locations of the cache content files in all caching nodes. Significantly, Γ_n^i, f_n^i, and f_{N+1}^i are just the description of the same content file i at different geographic

locations. Due to the fact that the content caching set $\mathbb{X}_n$ is the subset of $\mathbb{P}_n$, we can obtain $\mathbb{X}_n \subseteq \mathbb{P}_n, n \in \mathcal{N} \cup \{N+1\}$. For the convenience of the analysis, variable c_n^i is defined as

$$c_n^i = \begin{cases} 1 & f_n^i \in \mathbb{X}_n, \\ 0 & otherwise, \end{cases} \tag{2}$$

where, $\forall i \in \mathcal{I}, n \in \mathcal{N} \cup \{N+1\}$.

3.3. Cooperative Caching Model

In this section, we present how the DCEC networks serve a mobile user cooperatively in detail when the request of content file Γ_n^i is sent. As a requester, when U_n sends a request for Γ_n^i, there are four ways to fetch the content file i, as illustrated in Figure 3, which are summarized as follows:

- Local caching: The requested content file Γ_n^i is already stored in the cache set $\mathbb{X}_n$ of MD n, then the request can be served locally. The involved delay cost of obtaining the content file can be regarded as zero.
- D2D caching: When Γ_n^i cannot be served locally, MD n would broadcast the request to its neighboring MDs (helpers), and the potential helpers immediately search for the Γ_n^i in their own caching spaces $\mathbb{X}_m, m \in \mathcal{N} \backslash n$. If there is a copy of Γ_n^i in a typical MD, then it is transmitted to the requester through direct D2D links.
- Edge caching: Otherwise, the MDs will send the user requests to the serving BS. Once the BS receives the request, it will search for content file Γ_n^i in its edge cache space $\mathbb{X}_{N+1}$. On the condition that a copy of Γ_n^i exists, it is transmitted to the requester through the fronthaul links.
- Cloud server: If none of the above caching nodes store a copy of content file Γ_n^i, the request can be further forwarded to the cloud servers. Then, the requester can be served via backhaul links.

The local caching and D2D caching have a high response speed and do not occupy the fronthaul and backhaul links. Typically, our DCEC network processes requests based on the priority of the MD itself, the neighboring D2D-enabled MDs, the serving BS, and the cloud servers. When the requested content file is queried in a caching node, the request is not forwarded to a higher level.

For the purpose of formulating our cooperative content caching more easily, the binary variable $x_{m,n}^i$ is introduced. The detailed definition of $x_{m,n}^i$ is described below:

- $x_{m,n}^i = 1$ indicates that Γ_n^i is served by the neighboring MD m, $m \in \mathcal{N} \backslash \{n\}$. It also indicates that the content file Γ_n^i can be fetched from the MD itself (e.g., $n = m$). Thus, the request can be satisfied locally. Otherwise, $x_{m,n}^i = 0$.
- $x_{N+1,n}^i = 1$ indicates that Γ_n^i is served by $\mathbb{X}_{N+1}$ at the BS; otherwise, $x_{N+1,n}^i = 0$.
- $x_{0,n}^i = 1$ indicates that Γ_n^i is fetched from the content library at the remote cloud servers; otherwise, $x_{0,n}^i = 0$.

Thus, the requested content file Γ_n^i is constrained by

$$\sum_{m=0}^{N+1} x_{m,n}^i = 1, \forall i \in \mathcal{I}, n \in \mathcal{N}. \tag{3}$$

In (3), it can be seen that, for the requested content file Γ_n^i, only one of the binary variables can be 1 at any time. This guarantees that every Γ_n^i can be served.

3.4. Delay Cost Model

On the basis of the above analysis, we calculated the average delay cost of the requested content files of the MUs. We denote $t_{0,n}$ as the delay cost generated by a requested content file fetched from the content library in the cloud to MD n. Similarly, for edge caching, let

$t_{N+1,n}$ represent the delay of transferring the requested content between the BS and MD n. In this paper, it was assumed that the $t_{N+1,n}$ for all the requesters was same. In addition, for D2D caching, let $t_{m,n}$ denote the delay incurred in transferring a content file between MD m and MD n. For local caching, the corresponding delay of fetching the content file was zero, i.e., $t_{n,n} = 0$. Let $t_{0,N+1}$ denote the delay for the transmitting of a content file in the content library to the BS. Thus, we can obtain $t_{0,n} = t_{0,N+1} + t_{N+1,n}$.

Considering the huge difference in the communication distance, it is obvious that transmitting requested content files between the content library and BS generate a much greater delay, i.e., $t_{0,N+1} \gg t_{m,n}$. For this purpose, MDs are more willing to fetch the content files in the nearer caching node for a better QoE. To sum up, we can calculate the delay cost of fetching Γ_n^i as follows:

$$T_n^i = \sum_{m \in \mathcal{N}} x_{m,n}^i t_{m,n} + x_{N+1,n}^i t_{N+1,n} + x_{0,n}^i t_{0,n}, \quad \forall i \in \mathcal{I}, n \in \mathcal{N}. \tag{4}$$

Equation (4) indicates that MD n spends time fetching content file i under the cooperation of the cloud, edge, and MDs.

From the a priori knowledge of content popularity P_n^i, we can obtain the average delay cost of fetching content files requested by user U_n:

$$D_n = \sum_{i \in \mathcal{I}} P_n^i T_n^i, n \in \mathcal{N}. \tag{5}$$

It is known from (5) that D_n is proportional to P_n^i and T_n^i. In order to reduce the delay cost, for higher popularity content files, we should cache them at the MDs and/or edge servers for easier access by the MUs. Meanwhile, this pattern can also alleviate network congestion and reduce the network communication resource consumption.

The above parameters that we used in this article are summarized in Table 1.

Table 1. Notations and corresponding definitions.

Notation	Definition
N	the number of MDs
$\mathcal{N}$	the set of MDs
n	the index of MD
I	the maximum index of requested content file
$\mathcal{I}$	the set of indexes of requested content files
i	the index of requested content file
Γ_n^i	the content file i requested by MD n
T_n^i	the delay cost of fetching Γ_n^i
P_n^i	the popularity of content file i cached in MD n
P^i	the popularity of content file i
U_n^i	the user with MD n who requests Γ_n^i
U_n	the user who holds MD n
$\mathcal{U}$	the set of MUs that may initiate a content request
T_{max}	the maximum delay tolerance
S_n	the storage capacity of MDs
$\mathbb{X}_n$	the feasible cache placement decision set of MDs
S_{N+1}	the storage capacity of BS
$\mathbb{X}_{N+1}$	the cache set of BS
$\mathbb{P}$	the cache placement basic set
$\mathbb{P}_n$	the cache placement subset at MD n
$\mathbb{P}_{N+1}$	the cache placement subset at BS
f_n^i	the content file i cached in cache $\mathbb{X}_n$
f_{N+1}^i	the content file i cached in BS
c_n^i	indicator variable for whether content file i is cached in cache $\mathbb{X}_n$
c_{N+1}^i	indicator variable for whether content file i is cached in BS

Table 1. *Cont.*

Notation	Definition
$t_{0,n}$	the delay cost from cloud servers to MD n
$t_{0,N+1}$	the delay cost from cloud servers to BS
$t_{N+1,n}$	the delay cost from BS to MD n
$t_{m,n}$	the transmission delay from MD n to MD m
T_n^i	the delay cost of fetching Γ_n^i
D_n	the average delay cost of MU U_n

4. DCEC Cache Management Strategy

4.1. Problem Formulation

On the basis of the previous analysis, we modeled a content caching optimization problem, represented as follows:

$$P1: \min_{\mathbb{X}} \sum_{i\in\mathcal{I}} \sum_{U_n\in\mathcal{U}} \sum_{n\in\mathcal{N}} P_n^i T_n^i, \tag{6}$$

$$s.t.$$

$$x_{m,n}^i \in \{0,1\}, \forall m \in \mathcal{N}\cup\{0,N+1\}, n\in\mathcal{N}, i\in\mathcal{I}, \tag{7}$$

$$\sum_{m=0}^{N+1} x_{m,n}^i = 1, \forall i\in\mathcal{I}, n\in\mathcal{N}, \tag{8}$$

$$T_n^i < T_{max}, \forall i\in\mathcal{I}, n\in\mathcal{N}, \tag{9}$$

$$c_n^i \in \{0,1\}, \forall i\in\mathcal{I}, n\in\mathcal{N}\cup\{N+1\}, \tag{10}$$

$$x_{m,n}^i \le c_n^i, \forall n\in\mathcal{N}, i\in\mathcal{I}, \tag{11}$$

$$\sum_{i\in\mathcal{I}} c_n^i \le \mathcal{S}_n, \forall n\in\mathcal{N}. \tag{12}$$

The objective function $P1$ indicates the average delay cost of fetching content files by all MUs throughout the DCEC networks. The meaning of these above constraints is described below.

Equations (7) and (8) guarantee that every Γ_n^i can be served only in one location. Constraint (9) ensures that each request must be served within the maximum delay tolerance; otherwise, the request fails. Constraint (10) states whether the content file is stored in some caching node. Constraint (11) ensures that a content file can be available only when it is cached in the proper node ahead of time. Constraint (12) imposes that the overall caching capacity is no greater than the total storage capability.

4.2. Problem Transformation

In order to make it possible to figure out the objective function, we further converted the formation of the problem $P1$. As for the constraint (8), we can have

$$x_{0,n}^i = 1 - \sum_{m\in\mathcal{N}} x_{m,n}^i - x_{N+1,n}^i, \forall i\in\mathcal{I}, n\in\mathcal{N}. \tag{13}$$

Then, substituting them into (4), we can obtain

$$\begin{aligned} T_n^i &= \sum_{m\in\mathcal{N}} x_{m,n}^i t_{m,n} + x_{N+1,n}^i t_{N+1,n} + (1 - \sum_{m\in\mathcal{N}} x_{m,n}^i - x_{N+1,n}^i) t_{0,n} \\ &= t_{0,n} - \sum_{m\in\mathcal{N}} x_{m,n}^i (t_{0,n} - t_{m,n}) - x_{N+1,n}^i (t_{0,n} - t_{N+1,n}) \\ &= t_{0,n} - \sum_{m=1}^{N+1} x_{m,n}^i (t_{0,n} - t_{m,n}) \\ &= t_{0,n} - \mathcal{E}_n^i, \end{aligned} \tag{14}$$

where $\mathcal{E}_n^i$ can be expressed as

$$\mathcal{E}_n^i = \sum_{m\in\mathcal{N}} x_{m,n}^i (t_{0,n} - t_{m,n}) + x_{N+1,n}^i t_{0,N+1} = \sum_{m=1}^{N+1} x_{m,n}^i (t_{0,n} - t_{m,n}). \tag{15}$$

Here, $\mathcal{E}_n^i$ can be regarded as the delay cost savings of serving U_n, who request content file Γ_n^i, which only depends on the binary optimization variable. As a consequence, minimizing T_n^i under Constraints (7)–(12) is equivalent to maximizing $\mathcal{E}_n^i$. Thus, the problem $P1$ can be transformed as

$$\begin{aligned} P2: \max_{\mathbb{X}, x_{m,n}^i} & \sum_{i\in\mathcal{I}} \sum_{U_n\in\mathcal{U}} \sum_{n\in\mathcal{N}} P_n^i \mathcal{E}_n^i, \\ s.t.\ & (9), (10), (11), (12), \end{aligned} \tag{16}$$

$$x_{m,n}^i \in \{0,1\}, \forall m \in \mathcal{N} \cup \{N+1\}, n \in \mathcal{N}, i \in \mathcal{I}, \tag{17}$$

$$\sum_{m=1}^{N+1} x_{m,n}^i \leq 1, \forall i \in \mathcal{I}, n \in \mathcal{N}. \tag{18}$$

The objective function in $P2$ denotes the total utility value. Our target was to maximize this utility value under the constraints (9)–(12) and (17)–(18). This problem is NP-complete [30]; it is impractical to work it out. In this case, we tried to obtain a suboptimal solution. It belongs to a kind of problem that maximizes a monotone submodular function under the constraint of the matroid [29,30,49]. Next, we derive it as follows.

4.3. Cache Placement through Maximizing Monotone Submodular Function

In this section, we present that the constraints to $P2$ can be separated into the independent sets. Firstly, for the finite basic cache placement set $\mathbb{P}$ in (1), each cache placement decision set $\mathbb{X}_n$ is a subset of the content placement set, i.e.,

$$\mathbb{X}_n = \mathbb{X} \cap \mathbb{P}_n. \tag{19}$$

Given this, the constraint of caching capacity in (12) is equivalent to $\mathbb{X} \subseteq \mathbb{I}$, where

$$\mathbb{I} = \{\mathbb{X} \subseteq \mathbb{P} : |\mathbb{X} \cap \mathbb{P}_n| \leq S_n, \forall n = 1, \ldots, N, N+1\}. \tag{20}$$

Based on the definition of matroids [23,30,49], we let the capacity constraint in (20) form a partition matroid pair $\mathcal{M} = (\mathbb{P}, \mathbb{I})$. As far as (2) is concerned, the set $\{c_n^i : i \in \mathcal{I}\}$ can be denoted as the Boolean representation of $\mathbb{X}_n$. For the objective function in $P2$, we have Lemma 1.

Lemma 1. *The objective function in P2 is a monotone submodular function.*

Proof of Lemma 1. For each content file Γ_n^i and user MD, $\forall i \in \mathcal{I}$, $n \in \mathcal{N}$, we define the new variables $t_{0,n} - t_{m,n} = t_{m,n}^i, \forall m \in \mathcal{N} \cup \{N+1\}$; denote the objection function in $P2$ as $z(\mathbb{X})$, which can be expressed as

$$z(\mathbb{X}) = \sum_{i \in \mathcal{I}} \sum_{U_n \in \mathcal{U}} \sum_{n \in \mathcal{N}} P_n^i \mathcal{E}_n^i = \sum_{i \in \mathcal{I}} \sum_{U_n \in \mathcal{U}} \sum_{n \in \mathcal{N}} P_n^i \sum_{m=1}^{N+1} x_{m,n}^i t_{m,n}^i. \tag{21}$$

As the property of submodular functions is that the sum of monotone submodular functions is also monotone submodular [30], we just need to verify that the set function $z_n^i(\mathbb{X}) = V_n^i$ is monotone submodular, $\forall i \in \mathcal{I}$, $n \in \mathcal{N}$. With (21), we can indicate $z_n^i(\mathbb{X})$ as

$$z_n^i(\mathbb{X}) = \max_{x_{m,n}^i} \sum_{m=1}^{N+1} x_{m,n}^i t_{m,n}^i, \tag{22}$$

$$s.t.\ (9), (10), (11), (17), and (18).$$

The set function $z_n^i(\mathbb{X})$ can be expressed as

$$z_n^i(\mathbb{X}) = \max\{0, t_{1,n}^i c_{1,n}^i, \ldots, t_{N+1,n}^i c_{N+1,n}^i\}. \tag{23}$$

Thus, for $z_n^i(\mathbb{X})$, we prove its monotonicity and submodularity, respectively:

- Monotonicity: In an intuitive manner, adding a new content file to the decision set $\mathbb{X}$ will not weaken the set function [30]. Given a new file $f_s^i \in \mathbb{P} \backslash \mathbb{X}$, we denote $\mathbb{X}_s^i = \mathbb{X} \cup \{f_s^i\}$. Due to $\mathbb{X}_s^i \subseteq \mathbb{X}$, It is easy to obtain $z_n^i(\mathbb{X}_s^i) \geq z_n^i(\mathbb{X})$. As a result, $z_n^i(\mathbb{X})$ is a monotonic function, $\forall \mathbb{X} \subseteq \mathbb{P}$.
- Submodularity: For another cache placement decision set $\mathbb{Q}$, $\mathbb{Q} \subseteq \mathbb{X}$, we denote $\mathbb{Q}_s^i = \mathbb{Q} \cup \{f_s^i\}$. Since $z_n^i(\mathbb{X})$ is monotone, we have

$$z_n^i(\mathbb{X}) \geq z_n^i(\mathbb{Q}). \tag{24}$$

The marginal value incurred by adding the new file f_s^i to the sets $\mathbb{X}$ and $\mathbb{Q}$ can be represented as

$$z_{n,\mathbb{X}}^i(f_s^i) = z_n^i(\mathbb{X}_s^i) - z_n^i(\mathbb{X}), \tag{25}$$

$$z_{n,\mathbb{Q}}^i(f_s^i) = z_n^i(\mathbb{Q}_s^i) - z_n^i(\mathbb{Q}). \tag{26}$$

In order to prove $z_n^i(\cdot)$ is a submodular function, we can show that $z_{n,\mathbb{Q}}^i(f_s^i) \geq z_{n,\mathbb{X}}^i(f_s^i)$ [30]. For the convenience of expression, we denote that $\Theta_n^{is} = z_{n,\mathbb{Q}}^i(f_s^i) - z_{n,\mathbb{X}}^i(f_s^i)$. Thus, we need to prove that $\Theta_n^{is} \geq 0$ equivalently. For (23), the discussion is divided into three cases:

- $t_{m,n}^i < z_n^i(\mathbb{Q})$:
  For this case, it is easy to see that no value increased when adding a new content file f_s^i. Then, we can obtain $z_n^i(\mathbb{X}_s^i) = z_n^i(\mathbb{X})$ and $z_n^i(\mathbb{Q}_s^i) = z_n^i(\mathbb{Q})$; thus,

$$\Theta_n^{is} = 0. \tag{27}$$

- $t_{m,n}^i > z_n^i(\mathbb{X})$:
  In this case, we can have $z_n^i(\mathbb{X}_s^i) = z_n^i(\mathbb{Q}_s^i) = t_{m,n}^i$. Thus, based on the inequality (24), we obtain

$$\Theta_n^{is} = z_n^i(\mathbb{X}) - z_n^i(\mathbb{Q}) \geq 0. \tag{28}$$

- $z_n^i(\mathbb{Q}) \leq t_{m,n}^i \leq z_n^i(\mathbb{X})$
  For this one, we can obtain $z_n^i(\mathbb{X}_s^i) = z_n^i(\mathbb{X})$ and $z_n^i(\mathbb{Q}_s^i) = t_{m,n}^i$. Moreover,

$$\Theta_n^{is} = t_{m,n}^i - z_n^i(\mathbb{Q}) \geq 0. \tag{29}$$

In conclusion, we can obtain $\Theta_n^{is} \geq 0$ in any of these cases. Hence, $z_n^i(\cdot)$ is a monotone submodular function according to its definition [49]. □

We know that the constraints in (20) can be represented by a matroid pair. Based on the proof in Lemma 1, we can see the problem described by *P*2 is the problem of monotone submodular function maximization under matroid constraints. There are many possible algorithms to solve such problems. A popular approach is to use the greedy algorithm due to its high efficiency and low complexity.

4.4. DCEC Cache Management Scheme

The DCEC cache management scheme we propose includes two parts. The first part is the proactive cache placement of the content files under the constraints of caching capacity. The other one is the cache replacement when the cache is missing and a new content file from the cloud is added [50]. Accordingly, we propose a proactive cache placement algorithm and a reactive cache replacement algorithm in the following.

Algorithm 1 begins from an empty set for the cache placement solution. Then, it adds new files that have the maximum utility to the current content caching set $\mathbb{X}$. Finally, when the caches at the BS and MDs are full, the iteration stops, and the output content caching decision $\mathbb{X}$ is optimal. This process can be described as follows.

Algorithm 1 Proactive cache placement algorithm.

1. Initialization:
$\mathbb{P}_n = \{f_n^1, f_n^2, \ldots, f_n^I\}, \mathbb{X}_n = \emptyset, n \in \mathcal{N} \cup \{N+1\}$
$\mathbb{P} = \{\mathbb{P}_1, \mathbb{P}_2, \ldots, \mathbb{P}_{N+1}\}, \mathbb{X} = \{\mathbb{X}_1, \mathbb{X}_2, \ldots, \mathbb{X}_{N+1}\}$
2. Iteration:
while $\mathbb{P} \neq \emptyset$ **do**
$f_{n'}^{j'} = \arg\max_{f_n^j \in \mathbb{P} \backslash \mathbb{X}} [z(\mathbb{X} \cup \{f_n^j\}) - z(\mathbb{X})]$
$\mathbb{X} \leftarrow \mathbb{X} \cup \{f_{n'}^{j'}\}$
if $|\mathbb{X}_{n'}| = \mathcal{S}_{n'}$ **then**
$\mathbb{P} \leftarrow \mathbb{P} \backslash \mathbb{P}_{n'}$
end if
end while
3. Output: Content caching decision $\mathbb{X}$

At first, Algorithm 1 is initialized with an empty cache placement set in Part 1 and implements the iteration in Part 2. Here, the iteration will not terminate until $\mathbb{P}$ is empty. Within every iteration, we select file j' in cache $\mathbb{X}_{n'}$, denoted as $f_{n'}^{j'}$, which provides the maximum utility value. Therefore, $f_{n'}^{j'}$ can be taken as the candidate caching file in the scope of the unplaced files $\{f_n^j \in \mathbb{P} \backslash \mathbb{X}\}$. Then, $f_{n'}^{j'}$ is added to the current cache placement set. If the caches are full, content files will no longer be placed.

The cache placement problem is solved in Algorithm 1. Proactive caching can be completed during the off-peak period. However, when a requested content file does not exist in the available caches, there would be a cache missing. Then, the new requested content file will be fetched from remote content servers in the cloud DC. As all the caches are full, on the condition that such replacement improves the utility value, the DCEC networks will replace the least-utility-value file with the new one. The reactive cache replacement algorithm is described as follows.

The first part in Algorithm 2 describes a cache missing as the requested content file f_n^i is not in the caches. The iteration will repeat $N+1$ times. At each iteration, the least-utility-value content file $f_{n'}^{j'}$ is selected. If the utility value increases after replacement, update the files' set. Thus, the newest content caching decision is $\mathbb{X}$. The DCEC scheme ensures that the replacement does not decrease the utility value.

Algorithm 2 Reactive cache replacement algorithm.

1. Input: For the requested new content file i:
$f_n^i \notin \mathbb{X}, \forall n \in \mathcal{N} \cup \{N+1\}$
2. Iteration:
for $n = 1 : N+1$ **do**
$f_{n'}^{j'} = \arg\min_{f_n^j \in \mathbb{X}} [z(\mathbb{X}) - z(\mathbb{X} - f_n^j)]$
if $z(\mathbb{X} \backslash \{f_{n'}^{j'}\} \cup \{f_{n'}^i\}) > z(\mathbb{X})$ **then**
$\mathbb{X} \leftarrow \mathbb{X} \backslash \{f_{n'}^{j'}\} \cup \{f_{n'}^i\}$
end if
end for
3. Output: Content caching decision $\mathbb{X}$

5. Evaluation

In this section, the simulation results are presented to verify the advantage of the proposed hierarchical DCEC architecture and the proposed DCEC caching management strategy. We compared the DCEC network architecture with the D2D and MEC networks and compared the DCEC caching management scheme with the least-frequently used (LFU) scheme, as well as the least-recently used (LRU) scheme.

5.1. Simulation Settings

In our simulation, the programming software we used was MATLAB R2020b. The operating system was Windows 7. The RAM was 8.00 GB, and the processor was an Intel Core i5-6500 CPU @ 3.20 GHz. The D2D-enabled edge caching networks had four BSs. Each BS had a radius of 500 m. There were 80 MDs randomly distributed within the coverage of each BS. We assumed that each MD can establish a wireless link to its serving BS and a D2D link with a neighboring MD. The maximum range of each D2D link was set as 50 m. There were 1000 different content files available for the MUs, the size of each content file being 10 MB. The maximum delay tolerance of each task was 1 s [51,52]. The latency of fetching content files between the central cloud and BS, the BS and MDs, and the D2D was randomly assigned in the ranges [100, 200] ms, [20, 50] ms, and [5, 10] ms, respectively [53]. We set the cache sizes of each MD and BS as 0.05 GB and 4 GB [10,37]. The popularity of requested content file Γ_n^i followed the Zipf distribution:

$$P^i = \frac{1}{i^\alpha \sum_{j \in \mathcal{I}} j^{-\alpha}}, \quad \forall i \in \mathcal{I}, \tag{30}$$

where $\alpha \in (0, 1)$ [15].

Note that all the variables were independent for all the MDs. The main network parameters in our simulation are summarized in Table 2.

Table 2. Simulation parameters.

Parameter	Value
Radius of BS	500 m
The maximum D2D range	50 m
The square region	2000 m × 2000 m [54]
MD caching capacity	0.05 GB
Edge caching capacity	4 GB

Figure 4 depicts the random distribution of the MUs in the coverage of the four BSs within the square area. The square of different colors represent users within the range of BS. The square region ranged from −1000 m to 1000 m, both horizontally and vertically.

Each BS had a radius of 500 m. Their coverage did not overlap. The 80 MUs were randomly distributed within the coverage of each BS. Each MU held a mobile device.

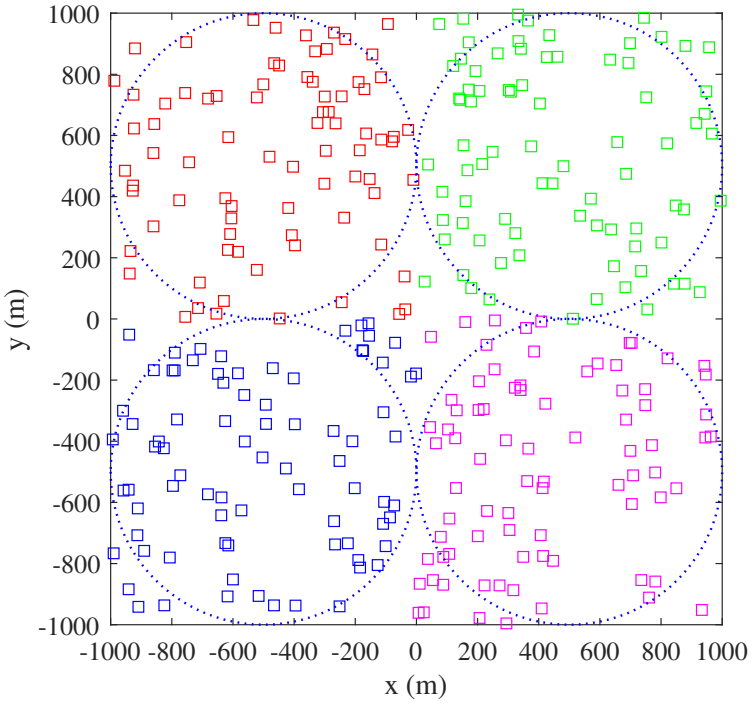

Figure 4. The users' distribution in the coverage of the BSs.

5.2. Performance of DCEC Architecture

In our DCEC network architecture, we considered hierarchical cooperation among the local MDs and edge caches. Content files can be fetched locally from the MD itself or its neighboring MDs, from the edge caches, or from the remote cloud. The performance of our proposed DCEC network architecture was compared with two other networks:

- D2D networks: In D2D networks, we considered cooperative caching only among neighboring MDs.
- MEC networks: In MEC networks, we only considered caching at the edge servers.

In the following simulations, we considered two metrics to evaluate our proposed network architecture:

- Cache hit ratio: The parameter cache hit ratio is an important indicator, which is used to measure whether a cache management strategy is efficient. When a content file is requested by an MU, if the content file is found in the caches, it is considered as a cache hit; otherwise, the content file has to be fetched from the content library in the remote cloud servers; this process is a cache miss. Thus, the cache hit ratio can be represented as follows:

$$CacheHitRatio = \frac{HitNumber}{HitNumber + MissNumber}. \tag{31}$$

- Average delay cost: the average delay cost of fetching content files for all MUs.

As depicted in Figure 5, we first compared the relationship between the average delay cost with the number of MDs according to (5). The average delay cost is expressed as the sum of the product of content popularity and the delay cost of fetching content. We can see that the more MDs there were, the higher the average delay cost was. Since the DCEC network architecture utilizes the MDs' caching resources for capacity enhancement, the

delay cost was lower than the other two networks. We can also observe that the average delay cost of the DCEC networks increased more slowly than the MEC and D2D networks.

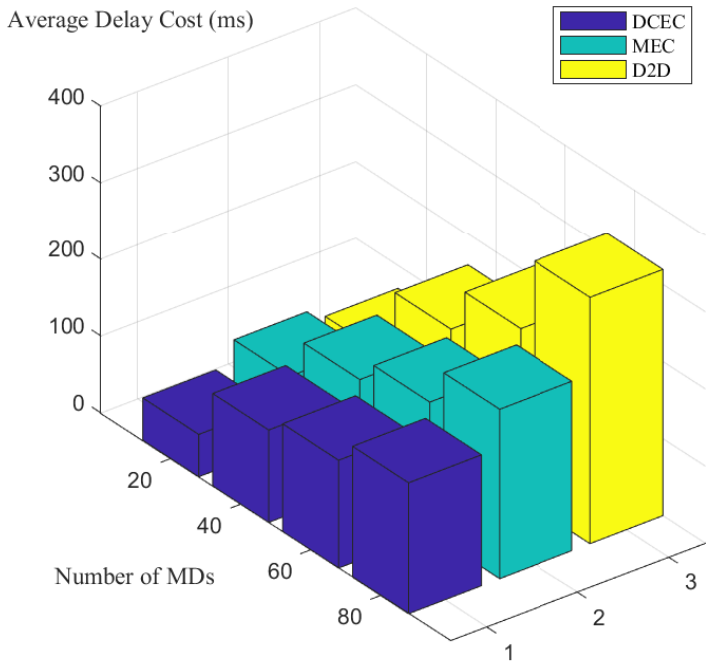

Figure 5. Average delay cost with different numbers of MDs.

As Figure 6 illustrates, this paper compared the cache hit ratio of the three network architectures with different numbers of cached content files. We can see that the cache hit ratio increased as the number of cached content files increased, and the cache hit ratio of DCEC was higher than those of the MEC and D2D networks.

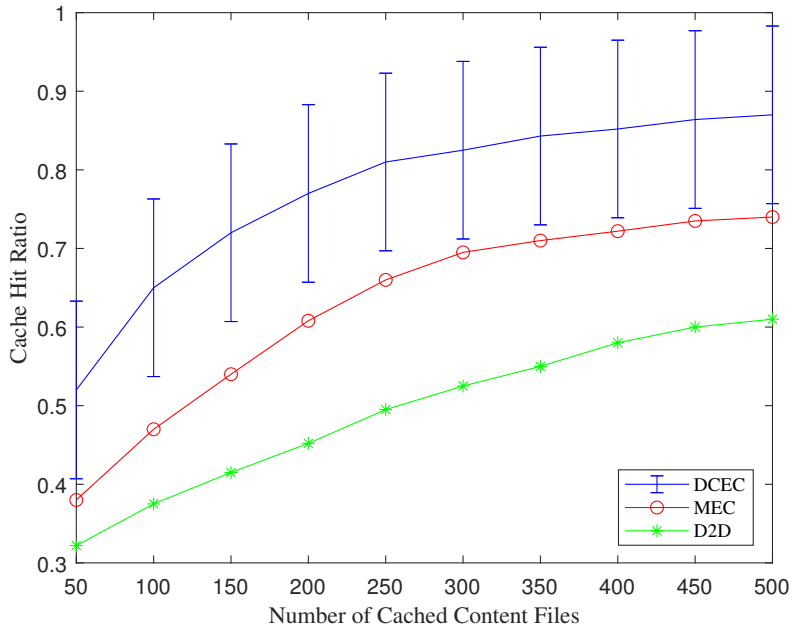

Figure 6. Cache hit ratio of different caching architectures.

Figure 7 depicts the average delay cost with different numbers of cached content files. We can observe that the average delay cost decreased with the increase of the edge cache

content files. From Figures 6 and 7, it is easy to see that the proposed DCEC architecture outperformed the D2D and MEC architectures.

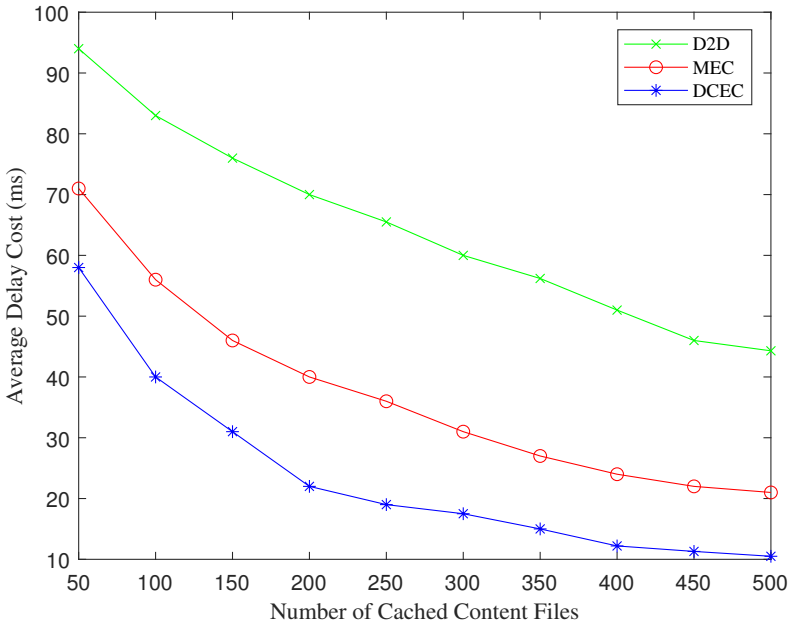

Figure 7. Average delay cost of fetching content files.

5.3. Performance of DCEC Cache Management Scheme

We compared our DCEC cache management scheme with the LFU and LRU using the mathematical analyses. When the caches are full, the LFU fetches content from the cloud and replaces the file that is least frequently used; the LRU replaces the file that is least recently used.

Figures 8 and 9 represent the cache hit ratio and average delay cost of the different algorithms with different numbers of cached content files, respectively. As the edge cache content files increased, the cache hit ratio increased correspondingly, whereas the average delay cost decreased. Since the DCEC scheme ensures that the utility value does not decrease, we can see that the cache hit ratio was greater than the other two schemes, and the average delay cost was lower than the other two schemes. On the basis of those results, we can conclude that the proposed DCEC architecture and scheme outperformed the others.

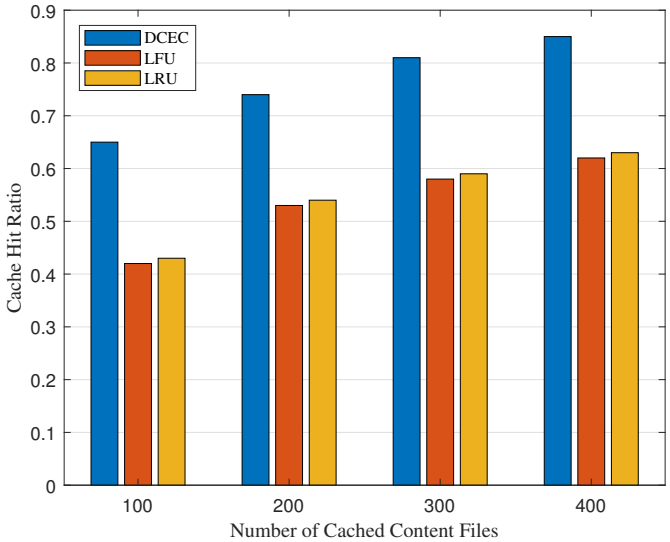

Figure 8. Cache hit ratio of different algorithms.

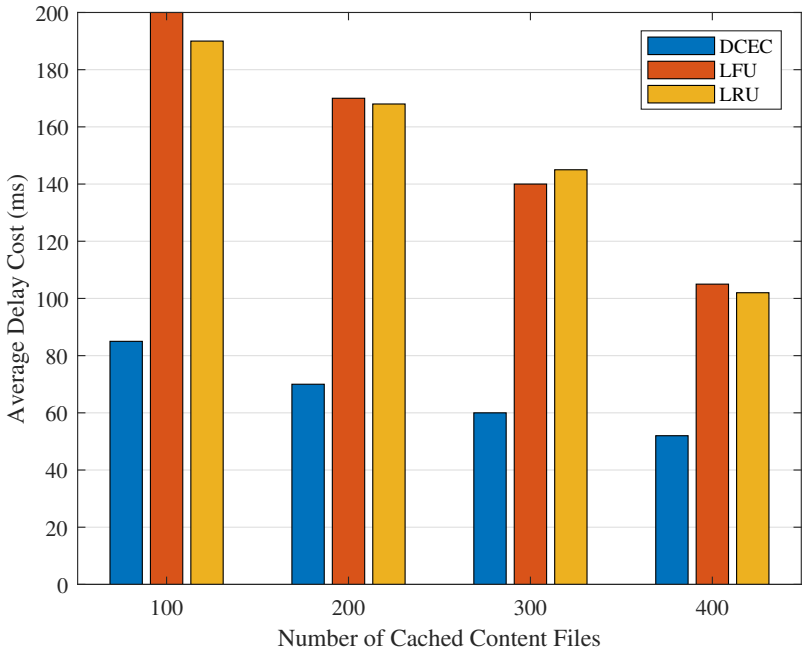

Figure 9. Average delay cost of different schemes.

6. Conclusions

In this article, we considered jointly exploiting the advantages of cooperative caching and D2D communication technologies in MEC networks, where the caching resources in edge nodes and D2D-enabled MDs can be efficiently utilized. We proposed the DCEC network architecture, in which each request for a content file can be served cooperatively. The formulated problem aimed to minimize the average delay cost. To work out this problem, we adopted monotone submodular function maximization to select the caching content files. The DCEC cache management scheme was proposed to find the optimal caching decision. The simulation results verified that the proposed DCEC architecture and DCEC cache management scheme were effective. Our proposals outperformed the MEC and D2D networks, the LFU and LRU schemes with the cache hit ratio and the average delay cost. Thus, users' QoE was improved. Therefore, in our future work, the DECE architecture will be further investigated in other application scenarios, such as wearable devices, navigation services, and so on. We also can consider federated learning to improve edge intelligence.

Author Contributions: Conceptualization, J.W., J.Z. and Y.J.; methodology, J.W., J.Z. and Y.J.; validation, J.W. and J.Z.; formal analysis, J.W. and J.Z.; investigation, J.W., J.Z. and Y.J.; resources, Y.J. and J.Z.; data curation, J.W. and J.Z.; writing—original draft preparation, J.W.; writing—review and editing, J.W., J.Z. and Y.J.; visualization, J.W.; supervision, J.Z. and Y.J.; project administration, J.Z. and Y.J.; funding acquisition, Y.J. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Nature Science Foundation of China (No. 62271078).

Data Availability Statement: The data that support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Wu, J.; Zhang, J.; Xiao, Y.; Ji, Y. Cooperative Offloading in D2D-Enabled Three-Tier MEC Networks for IoT. *Wirel. Commun. Mob. Comput.* **2021**, *2021*, 9977700:1–9977700:13. [CrossRef]
2. Jiang, F.; Wang, K.; Dong, L.; Pan, C.; Xu, W.; Yang, K. AI Driven Heterogeneous MEC System with UAV Assistance for Dynamic Environment: Challenges and Solutions. *IEEE Netw.* **2021**, *35*, 400–408. [CrossRef]
3. Liu, D.; Yang, C. Cache-enabled heterogeneous cellular networks: Comparison and tradeoffs. In Proceedings of the ICC 2016, Kuala Lumpur, Malaysia, 22–27 May 2016; pp. 1–6.
4. Plachy, J.; Becvar, Z.; Strinati, E.C.; di Pietro, N. Dynamic Allocation of Computing and Communication Resources in Multi-Access Edge Computing for Mobile Users. *IEEE Trans. Netw. Serv. Manag.* **2021**, *18*, 2089–2106. [CrossRef]
5. Chen, B.; Yang, C.; Molisch, A.F. Cache-Enabled Device-to-Device Communications: Offloading Gain and Energy Cost. *IEEE Trans. Wirel. Commun.* **2017**, *16*, 4519–4536. [CrossRef]
6. Li, J.; Peng, Z.; Xiao, B.; Hua, Y. Make smartphones last a day: Pre-processing based computer vision application offloading. In Proceedings of the SECON 2015, Seattle, WA, USA, 22–25 June 2015; pp. 462–470.
7. Hauswald, J.; Manville, T.; Zheng, Q.; Dreslinski, R.G.; Chakrabarti, C.; Mudge, T.N. A hybrid approach to offloading mobile image classification. In Proceedings of the ICASSP 2014, Florence, Italy, 4–9 May 2014; pp. 8375–8379.
8. Wang, X.; Ji, Y.; Zhang, J.; Bai, L.; Zhang, M. Low-Latency Oriented Network Planning for MEC-Enabled WDM-PON Based Fiber-Wireless Access Networks. *IEEE Access* **2019**, *7*, 183383–183395. [CrossRef]
9. Liu, Z.; Zhang, J.; Wu, J. Joint Optimization of Server Placement and Content Caching in Mobile Edge Computing Networks. In Proceedings of the ICNCC 2019: The 8th International Conference on Networks, Communication and Computing, Luoyang China, 13–15 December 2019.
10. Ghosh, S.; Agrawal, D.P. A high performance hierarchical caching framework for mobile edge computing environments. In Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC), Nanjing, China, 29 March–1 April 2021; pp. 1–6.
11. Tran, T.X.; Hajisami, A.; Pandey, P.; Pompili, D. Collaborative Mobile Edge Computing in 5G Networks: New Paradigms, Scenarios, and Challenges. *IEEE Commun. Mag.* **2017**, *55*, 54–61. [CrossRef]
12. Xie, R.; Tang, Q.; Wang, Q.; Liu, X.; Yu, F.R.; Huang, T. Satellite-Terrestrial Integrated Edge Computing Networks: Architecture, Challenges, and Open Issues. *IEEE Netw.* **2020**, *34*, 224–231. [CrossRef]
13. Sang, Z.; Guo, S.; Wang, Y. Collaborative Video Cache Management Strategy in Mobile Edge Computing. In Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC), Nanjing, China, 29 March–1 April 2021; pp. 1–6.
14. Zhang, X.; Ren, Y.; Lv, T.; Hanzo, L. Caching Scalable Videos in the Edge of Wireless Cellular Networks. *IEEE Netw.* 2022, *early access*. [CrossRef]
15. Ji, M.; Caire, G.; Molisch, A.F. Wireless Device-to-Device Caching Networks: Basic Principles and System Performance. *IEEE J. Sel. Areas Commun.* **2016**, *34*, 176–189. [CrossRef]
16. Yi, C.; Huang, S.; Cai, J. Joint Resource Allocation for Device-to-Device Communication Assisted Fog Computing. *IEEE Trans. Mob. Comput.* **2021**, *20*, 1076–1091. [CrossRef]
17. Kim, J.; Kim, T.; Hashemi, M.; Brinton, C.G.; Love, D.J. Joint Optimization of Signal Design and Resource Allocation in Wireless D2D Edge Computing. In Proceedings of the INFOCOM 2020, Toronto, ON, Canada, 6–9 July 2020; pp. 2086–2095.
18. Li, J.; Peng, Z.; Gao, S.; Xiao, B.; Chan, H.C.B. Smartphone-assisted energy efficient data communication for wearable devices. *Comput. Commun.* **2017**, *105*, 33–43. [CrossRef]
19. Tran, T.X.; Hajisami, A.; Pompili, D. Cooperative Hierarchical Caching in 5G Cloud Radio Access Networks (C-RANs). *IEEE Netw.* **2016**, *31*, 35–41. [CrossRef]
20. Wu, D.; Zhou, L.; Cai, Y.; Qian, Y. Collaborative Caching and Matching for D2D Content Sharing. *IEEE Wirel. Commun.* **2018**, *25*, 43–49. [CrossRef]
21. Li, X.; Wang, X.; Wan, P.; Han, Z.; Leung, V.C.M. Hierarchical Edge Caching in Device-to-Device Aided Mobile Networks: Modeling, Optimization, and Design. *IEEE J. Sel. Areas Commun.* **2018**, *36*, 1768–1785. [CrossRef]
22. Yuan, P.; Huang, R. Integrating the device-to-device communication technology into edge computing: A case study. *Peer Peer Netw. Appl.* **2021**, *14*, 599–608. [CrossRef]
23. Sang, Z.; Guo, S.; Wang, Q.; Wang, Y. GCS: Collaborative video cache management strategy in multi-access edge computing. *Ad Hoc Netw.* **2021**, *117*, 102516. [CrossRef]
24. Baccour, E.; Erbad, A.; Mohamed, A.; Guizani, M.; Hamdi, M. CE-D2D: Collaborative and Popularity-aware Proactive Chunks Caching in Edge Networks. In Proceedings of the 16th International Wireless Communications and Mobile Computing Conference (IWCMC), Limassol, Cyprus, 15–19 June 2020; pp. 1770–1776.
25. Kafiloglu, S.S.; Gür, G.; Alagöz, F. Cooperative Caching and Video Characteristics in D2D Edge Networks. *IEEE Commun. Lett.* **2020**, *24*, 2647–2651. [CrossRef]
26. Zhang, K.; Leng, S.; He, Y.; Maharjan, S.; Zhang, Y. Cooperative Content Caching in 5G Networks with Mobile Edge Computing. *IEEE Wirel. Commun.* **2018**, *25*, 80–87. [CrossRef]
27. Wang, N.; Shao, W.; Bose, S.K.; Shen, G. MixCo: Optimal Cooperative Caching for Mobile Edge Computing in Fiber-Wireless Access Networks. In Proceedings of the 2018 Optical Fiber Communications Conference and Exposition (OFC), San Diego, CA, USA, 11–15 March 2018.

28. Somesula, M.K.; Rout, R.R.; Somayajulu, D.V.L.N. Cooperative cache update using multi-agent recurrent deep reinforcement learning for mobile edge networks. *Comput. Netw.* **2022**, *209*, 108876. [CrossRef]
29. Somesula, M.K.; Rout, R.R.; Somayajulu, D.V.L.N. Contact duration-aware cooperative cache placement using genetic algorithm for mobile edge networks. *Comput. Netw.* **2021**, *193*, 108062. [CrossRef]
30. Tran, T.X.; Le, D.V.; Yue, G.; Pompili, D. Cooperative Hierarchical Caching and Request Scheduling in a Cloud Radio Access Network. *IEEE Trans. Mob. Comput.* **2018**, *17*, 2729–2743. [CrossRef]
31. Zhao, T.; He, L.; Huang, X.; Li, F. QoE-Driven Secure Video Transmission in Cloud-Edge Collaborative Networks. *IEEE Trans. Veh. Technol.* **2022**, *71*, 681–696. [CrossRef]
32. Tran, T.X.; Pandey, P.; Hajisami, A.; Pompili, D. Collaborative multi-bitrate video caching and processing in Mobile-Edge Computing networks. In Proceedings of the Wireless On-demand Network Systems & Services (WONS), Jackson, WY, USA, 21–24 February 2017; pp. 165–172.
33. Chiang, Y.; Hsu, C.; Wei, H. Collaborative Social-Aware and QoE-Driven Video Caching and Adaptation in Edge Network. *IEEE Trans. Multim.* **2021**, *23*, 4311–4325. [CrossRef]
34. Bhar, C.; Agrell, E. Energy- and Bandwidth-Efficient, QoS-Aware Edge Caching in Fog-Enhanced Radio Access Networks. *IEEE J. Sel. Areas Commun.* **2021**, *39*, 2762–2771. [CrossRef]
35. Li, Q.; Shi, W.; Xiao, Y.; Ge, X.; Pandharipande, A. Content Size-Aware Edge Caching: A Size-Weighted Popularity-Based Approach. In Proceedings of the IEEE Global Communications Conference (GLOBECOM 2018), Abu Dhabi, United Arab Emirates, 9–13 December 2018; pp. 206–212.
36. Miao, F.; Chen, D.; Jin, L. Multi-level PLRU Cache Algorithm for Content Delivery Networks. In Proceedings of the 10th International Symposium on Computational Intelligence and Design (ISCID 2017), Hangzhou, China, 9–10 December 2017; pp. 320–323.
37. Wang, X.; Li, R.; Wang, C.; Li, X.; Taleb, T.; Leung, V.C.M. Attention-Weighted Federated Deep Reinforcement Learning for Device-to-Device Assisted Heterogeneous Collaborative Edge Caching. *IEEE J. Sel. Areas Commun.* **2021**, *39*, 154–169. [CrossRef]
38. Zhang, X.; Zhu, Q. Collaborative Hierarchical Caching over 5G Edge Computing Mobile Wireless Networks. In Proceedings of the IEEE International Conference on Communications (ICC), Kansas City, MO, USA, 20–24 May 2018; pp. 1–6.
39. Somesula, M.K.; Rout, R.R.; Somayajulu, D.V.L.N. Deadline-aware caching using echo state network integrated fuzzy logic for mobile edge networks. *Wirel. Netw.* **2021**, *27*, 2409–2429. [CrossRef]
40. Kachris, C.; Tomkos, I. A Survey on Optical Interconnects for Data Centers. *IEEE Commun. Surv. Tutor.* **2012**, *14*, 1021–1036. [CrossRef]
41. Liu, Z.; Jiawei, Z.; Yanan, L.; Lin, B.; Yuefeng, J. Joint Jobs Scheduling and Lightpath Provisioning in Fog Computing Micro Datacenter Networks. *J. Opt. Commun. Netw.* **2018**, *10*, B152–B163. [CrossRef]
42. Ji, Y.; Gu, R.; Yang, Z.; Li, J.; Li, H.; Zhang, M. Artificial intelligence-driven autonomous optical networks: 3S architecture and key technologies. *Sci. China Inf. Sci.* **2020**, *63*, 160301. [CrossRef]
43. Mu, J.; Jing, X.; Zhang, Y.; Gong, Y.; Zhang, R.; Zhang, F. Machine Learning-Based 5G RAN Slicing for Broadcasting Services. *IEEE Trans. Broadcast.* **2022**, *68*, 295–304. [CrossRef]
44. Zhang, P.; Wang, C.; Jiang, C.; Han, Z. Deep Reinforcement Learning Assisted Federated Learning Algorithm for Data Management of IIoT. *IEEE Trans. Ind. Inform.* **2021**, *17*, 8475–8484. [CrossRef]
45. Mu, J.; Gong, Y.; Zhang, F.; Cui, Y.; Zheng, F.; Jing, X. Integrated Sensing and Communication-Enabled Predictive Beamforming With Deep Learning in Vehicular Networks. *IEEE Commun. Lett.* **2021**, *25*, 3301–3304. [CrossRef]
46. Cui, Y.; Jing, X.; Mu, J. Integrated Sensing and Communications Via 5G NR Waveform: Performance Analysis. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2022), Virtual, 23–27 May 2022; pp. 8747–8751.
47. Zhang, P.; Pang, X.; Kumar, N.; Aujla, G.S.; Cao, H. A Reliable Data-transmission Mechanism using Blockchain in Edge Computing Scenarios. *IEEE Internet Things J.* **2022**, *9*, 14228–14236. [CrossRef]
48. Xu, C.; Pu, L.; Lin, G.; Wu, W.; Di, W. Exploiting Massive D2D Collaboration for Energy-Efficient Mobile Edge Computing. *IEEE Wirel. Commun.* **2017**, *24*, 64–71.
49. Călinescu, G.; Chekuri, C.; Pál, M.; Vondrák, J. Maximizing a Monotone Submodular Function Subject to a Matroid Constraint. *SIAM J. Comput.* **2011**, *40*, 1740–1766. [CrossRef]
50. Yao, Y.; Xiao, B.; Wang, W.; Yang, G.; Zhou, X.; Peng, Z. Real-Time Cache-Aided Route Planning Based on Mobile Edge Computing. *IEEE Wirel. Commun.* **2020**, *27*, 155–161. [CrossRef]
51. He, Y.; Ren, J.; Yu, G.; Cai, Y. D2D Communications Meet Mobile Edge Computing for Enhanced Computation Capacity in Cellular Networks. *IEEE Trans. Wireless Commun.* **2019**, *18*, 1750–1763. [CrossRef]
52. Pham, X.; Nguyen, T.; Nguyen, V.; Huh, E. Joint Service Caching and Task Offloading in Multi-Access Edge Computing: A QoE-Based Utility Optimization Approach. *IEEE Commun. Lett.* **2021**, *25*, 965–969. [CrossRef]

53. Li, X.; Wang, X.; Xiao, S.; Leung, V.C.M. Delay performance analysis of cooperative cell caching in future mobile networks. In Proceedings of the 2015 IEEE International Conference on Communications (ICC 2015), London, UK, 8–12 June 2015; pp. 5652–5657.
54. Balasubramanian, V.; Aloqaily, M.; Reisslein, M.; Scaglione, A. Intelligent Resource Management at the Edge for Ubiquitous IoT: An SDN-Based Federated Learning Approach. *IEEE Netw. Mag. Comput. Commun.* **2021**, *35*, 114–121. [CrossRef]