Clustering-Based Device-to-Device Cache Placement ☆

Ahmet Cihat Kazez[a], Tolga Girici[a,*]

[a] *TOBB University of Economics and Technology, Sogutozu Cad. No 43, Sogutozu 06560, Ankara, Turkey.*

Abstract

In this work we consider the problem of optimal cache placement in a D2D enabled cellular network. There are a number of helper devices in the area, which use their cached contents to help other users and offload traffic from the base station. The goal of cache placement is maximizing the offloaded traffic. We first formulate and optimally solve the cache placement problem as a mixed integer linear program. Then we propose a distributively implementable algorithm that clusters helpers. Helpers in each cluster collectively decide the contents to be cached, based on the content popularity. Numerical evaluations show that the proposed cache placement scheme always performs within 5% of the optimal result and it is robust to popularity profile and cache capacity.

Keywords: Caching, Wireless, D2D, Clustering, Integer Programming

1. Introduction

Ongoing developments in the wireless communications technology have led to widespread use of smart devices. Apart from computers and cellular phones, tablets, wearable devices and even vehicles can be connected to the Internet. This variety results in an unprecedented increase in demand for enriched content such as real time applications and video streaming. Excessive increase in demand may cause a considerable decrease in quality of service at the user side.

☆Fully documented templates are available in the elsarticle package on CTAN.

*Corresponding author

Email address: tgirici@etu.edu.tr (Tolga Girici)

In order to avoid such an outcome, deploying more base stations, increasing the amount of allocated bandwidth and physical layer improvements in wireless communication technology can be proposed as solutions. However, for dense network scenarios, these solutions may be inadequate to alleviate high cellular load and alternative approaches are required such as Device-to-Device (D2D) communications.

D2D communications is a technology that enables direct communication of nearby devices, without using the base station as a relay. This technology has a potential of significantly decreasing delay and increasing throughput. With these potentials, D2D communications is seen as one of the 10 enabling technologies of 5G [1]. D2D technology will lead to several proximity based services (ProSe) such as Local Services, Emergency Communications, IoT enhancement, terminal relaying, indoor positioning [2]. Another promising application of D2D communications is content delivery. D2D technology allows content caching at the network edge such as user terminals and helpers. Caching popular contents at various nodes in the network is an attractive solution in order to alleviate the peak load [3]. With this solution nodes can directly supply the future demands of other users via D2D links without increasing the backhaul traffic. Of course, one of the challenging points in D2D caching is the limited cache capacity of mobile devices. An advantage is that D2D transmissions can be performed with low power within a short proximity. Moreover, as an advantage of short range, in D2D devices can reuse the cellular bandwidth which improves spectral efficiency significantly. Taking these into account, fundamental problem is the cache placement (i.e. which device to cache which content).

The works in [4] [5] and [6] are fundamental studies in wireless D2D/femtocell caching. In [4] the authors prove that a simple randomized device caching policy (where the devices independently choose the file to cache, according to the popularity distribution) achieves the optimal scaling of behavior of D2D transmission opportunities in terms of total number of nodes. Note that, this is only a guarantee in terms of the *scaling* of *expected* number of D2D content deliveries with increasing number of users. However, *actual* performance can be very different in a given topology. In fact, we see in our simulations that our proposed clustering-based caching scheme outperforms such a popularity-based scheme. The authors in [5] proposed a femtocell cache placement algorithm based on bipartite matching. They prove that the proposed algorithm guarantees a delay performance within a factor 2 of the optimum. Although this is a valuable theoretical result, a factor of 2 is quite large. Moreover, the proposed algorithm is not distributed. Hence [4] and [5] only consider the throughput scaling laws and approximation ratios instead of the actual performances. Dissimilarly to the [5], in [6] authors introduced a coded caching solution, where rateless coded fragments of the contents are stored at distributed caches. However the proposed method is centralized. Moreover, throughput-outage trade of in D2D communications has been investigated in [7]. Authors define a *cluster*, which denotes the set of neighbors that a device asks for a content. The choice of the cluster size strikes a tradeoff between throughput and cache hit. As a result, the works [5], [6] and [7] find some fundamental performance results, however they do not propose a distributively implementable algorithm. On the other hand, our proposed algorithm is both amenable to distributed implementation, and it is tested by varying all parameters for robustness.

In the presence of D2D caching, the number of devices is potentially very large. A centralized cache placement, scheduling and resource allocation by the base station would increase the control message overhead to prohibitive levels. Therefore distributed solutions are sought in the literature. In [8] authors evaluated a distributed caching mechanism in order to maximize data offloading in an interference aware network. Proposed algorithm is based on the receiver-transmitter matching subject to interference constraint and compared to optimal solution which is derived based on known network information. Users try to find out the most valuable partner in order to match and become a transmitter-receiver pair. This scheme is based on the one-to-one matching and does not allow multiple access. In other words, users cannot request contents from the other users in their neighborhood unless they are matched. Such a constraint hampers the content diversity which improves the offloading rate of the network. Benefits of the content diversity are explained in the following sections. Besides, in [8] a device does not proactively cache a content just to help others. It only caches a content that it previously requested and obtained for itself. The authors in [9] devise a random access policy to maximize the cache-hit. However, they assume that the cache placement as given. In [10] optimal cache placement problem with different user mobilities, cache sizes and content popularities are considered. The authors proved that optimization problem is NP-hard and proposed an infrastructure-aided and distributed data offloading algorithm. The main assumption here is that the devices are mobile and D2D transmission can happen when the two devices are in content. In our work we assume fixed devices, or devices with intermittent mobility. Algorithms and solutions for a mobile network case is a direction for future research.

Game theoretical approaches that give caching decisions in a decentralized manner also take place in literature. Works in [11],[12], [13] and [14] propose cache placement algorithms using Stackelberg game model. In [11] and [13] D2D caching is not considered. In [12] Stackelberg framework provides an incentive mechanism in order to overcome the unwilling and selfish nature of D2D users and lead them to cache content and help others. In a previous work [14] we have implemented the algorithm in [12]. Simulation results have shown that for a realistic number of devices and contents Stackelberg-based algorithms require hundreds of iterations for convergence or an acceptable performance. On the other hand in our algorithm, once the clusters are formed, cache placement is easily performed.

There is also a group of works on reducing latency. In [5] authors focused on the cache placement problem, aiming at minimization of average download time. Assuming known content popularities, coded and uncoded caching schemes are considered. Authors in [15] analyze the average network delay minimization. In this work, the nearby devices form a cluster and cache contents in a disjoint manner. That is, the same content is not cached twice in a cluster. Our approach, as will be explained in the subsequent sections, has some similarities. However, we assume the existence of a set of designated helper nodes and perform clustering on the helpers. As opposed to [15], in our case caching popular contents more than once in a cluster significantly improves the performance. In [15] users in the same cluster can exchange cached contents via D2D links, inter cluster cooperation is also possible between users via cellular links. Similar to these works, optimal caching decision is found centrally in a way that minimizes latency in [16].

A group of works in the literature use stochastic geometry in finding some fundamental results on cache placement. The work in [17] uses stochastic geometry and proposes a low-complexity random cache placement and scheduling algorithm. Authors in [18] prove that spatially correlated caching improves the hit probability. In these works the main assumption is a Poisson Point Process (i.e. uniform) user distribution. This technique can be used to mathematically derive the cache hit probability averaged over all possible topologies. They define an *exclusion region*, wherein if a user caches a content, then other users in close proximity do not cache that content. This exclusion region avoids redundant caching and provides content diversity. They [18] optimize the performance with respect to only the exclusion region. However, this is a very simplistic approach and there is room for significant improvement. As will be seen in the following sections, we also utilize spatially correlated caching. However, in reality, only a fraction of cellular users would be eager to participate in caching and content delivery. We call these users as *helpers* and helpers from clusters. Clustering helpers is crucial since it provides content diversity and avoids duplicate caching of less popular contents at helpers that serve common users. However, contrary to [18] close helpers can still cache the same content if that content is very popular. The reason is that these helpers do not serve exactly the same set of users. Clustering approach reveals the advantages when proposed algorithm is compared to conventional caching schemes and the optimal solution. Meanwhile, clustering was also chosen as a method in [19], [20], [21]. However, in these works there are no helpers and only the users are clustered. Therefore, these works significantly differ from our work. The contributions of this paper are summarized as follows,

- We propose a distributed cache placement algorithm for D2D enabled cellular networks in order to maximize the network's average offloading. Optimization problem is modeled as Mixed Integer Linear Programming (MILP) and optimal solution is provided.

- The derived optimal solution is centralized and expected to be handled by BS, requiring knowledge about network topology. Such kind of solutions cause computational complexities and signaling overhead. In order to obviate these drawbacks we concentrated on distributed cache placement approaches since they provide scalable and practical solutions. For this purpose, we proposed a clustering-based cache placement algorithm.

- We have analyzed the performance of the proposed algorithm in the presence of varying cache size, number of network elements, skewness parameter of Zipf distribution and number of contents. Results of these extensive simulations reveal that the proposed algorithm is robust with reasonable performance compared to conventional caching schemes and optimal caching.

The rest of the paper is organized as follows. Section 2 presents the system model. Section 3 formulates the optimal caching problem. Section 4 provides details of the proposed Clustering and Cache Placement Algorithms. Section 5 includes simulations with the varying parameters. Section 6 concludes the paper and Section 7 mentions the issues that can be considered as future work.

2. System Model

We assume an LTE cell containing a base station that serves $U + H$ users, which are composed of U ordinary users and H helper users. Helpers are able to cache contents and serve neighboring users, using device-to-device (D2D) communications. In D2D communications , two users that are closer than a range R_D are able to communicate directly. We assume there are C cacheable contents. Each helper can store C_c contents. We assume a uniform content popularity profile throughout the network. Content popularities are Zipf distributed. Without loss of generality the contents are ordered in decreasing order of popularity. Let p_{uc} be the probability that user u requests content c. Zipf rule suggests that p_{uc} is proportional to $\frac{1}{c^{\alpha}}$ for all users u. Here, α is the skewness exponent for the Zipf distribution. A higher α means a more imbalanced popularity. We assume that when a content is requested by an ordinary user, this request can be heard by a helper. If the content is available at the helper, then a D2D transmission starts for content retrieval. If the requested content is not available at any helper, then it is obtained from the base station. In this system model, we aim to maximize the ratio of contents obtained from the helpers (i.e. offloading). A helper can help an ordinary user, only if they are neighbors. Let binary parameter a_{hu} be the neighborhood parameter, which takes value one if helper h and user u are neighbors, zero otherwise. In the proposed solutions, we implicitly assume that the base station is able to measure the popularity of contents and regularly inform the helpers about the content popularity profiles.

We assume that D2D transmissions do not create interference to the actual cellular transmissions. This can be possible by D2D overlay, where D2D transmissions use a separate frequency band. We assume that sufficient bandwidth is allocated to D2D transmissions.

Another possibility of implementing D2D transmissions is using a different technology. For example, WiFi-Direct is an extension of the classical WiFi technology for D2D transmissions. This technology enables short range, low power, high bandwidth communications, even in the absence of Internet connection. WiFi-direct can be implemented on top of the LTE cellular infrastructure, without any major change in LTE protocols [22]. With this technology, neighboring users can form groups and perform direct communication.

3. Problem Formulation

We define two sets of binary variables. Let x_{hc} be the binary variable that takes value 1 if content c is cached by helper h. Let y_{huc} be the binary variable that takes value 1 if helper h is authorized to help ordinary user u if it requests content c. We define the following optimization problem,

$$\max_{\mathbf{x},\mathbf{y}} \left\{ \sum_{u=1}^{U} \sum_{c=1}^{C} p_{uc} \sum_{h=1}^{H} y_{huc} \right\} \tag{1}$$

s.t.

$$\sum_{c=1}^{C} x_{hc} \leq C_c, \forall h = 1, \ldots, H \tag{2}$$

$$\sum_{h=1}^{H} y_{huc} \leq 1, \forall u = 1, \ldots, U, c = 1, \ldots, C \tag{3}$$

$$y_{huc} \leq a_{hu} x_{hc}, \forall u = 1, \ldots, U, c = 1, \ldots, C, h = 1, \ldots, H \tag{4}$$

Objective in (1) is the average received help (i.e. offloading) throughout the cellular area. Constraint (2) is the cache capacity of each helper node. Constraint (3) enforces that each node can only get help from at most one node, for each content. Therefore the quantity $\sum_{h=1}^{H} y_{huc}$ is a binary quantity. It becomes one, if node u receives the content from any helper and becomes zero, otherwise. Finally, constraint (4) enforces that an ordinary node u can receive help from helper h, in terms of content c, only if helper h caches that content and is a neighbor of node u.

The proposed problem and its constraints are all linear in terms of the decision variables. Moreover, the variables are all binary. Therefore this is a mixed-integer linear program (MILP). It can be solved using off-the shelf solvers such as CPLEX.

4. Proposed Algorithm

Maximizing the traffic offloading is directly related to maximizing the hit probability. This is the probability of a user finding a requested content from a neighboring helper. A very simple cache placement solution would be each helper caching the most popular contents. Although this solution will be a benchmark in our simulations, it is certainly not the best solution. Consider two helpers located very close to each other. In this case it would be better to cache different contents at these nodes. Caching this way improves the chance of an ordinary node to find a requested content at a helper neighbor. Therefore, we propose a clustering-based cache placement solution. In our solution, helpers form cliques. Helpers in a clique are in direct communication range with each other. After clusters are formed, each cluster will independently perform cache placement in a way that provides content diversity and maximizes hit probability.

The reason of using cliques (clusters) is to facilitate collaboration of helpers and provide content diversity in a cluster. There are vast number of clustering methods developed for wireless ad hoc and sensor networks [23]. Our definition of clique is a set of helpers all having *one hop* distance between each other. This choice of *single hop* is for the following reasons: 1) Helpers in a clique can communicate directly with each other and hierarchically allocate contents to their cache. For example helpers in a clique can send the popularity and cache capacity information to a designated clusterhead, and the clusterhead can perform cluster cache allocation. 2) Helpers are very close to each other, so that they serve mostly common users. A user can easily find a proximate helper that caches a desired popular content.

Algorithm 1 shows the pseudocode of the clustering algorithm. Algorithm accepts the neighborhood and distance information as inputs. Channel gains can also be used instead of distances. Algorithm consists of two nested loops. In each outer loop, a separate clique is formed. Set $\mathcal{H}$ is initialized as all helpers. Each time a helper joins a cluster, it is excluded from $\mathcal{H}$. Line 4 finds the helper pair with minimum distance, among the remaining helpers. If these helpers are neighbors, then a new cluster is initialized (Lines 6,7). Inner loop (Lines 9-18) tries to add as many helpers to this cluster as possible. Line 10 finds the set of helpers that can directly communicate (i.e. neighbors) to each existing helpers in the cluster. If there are no such helpers, then the cluster is finalized (Line 16). If there are such helpers, then the algorithm finds the one with least total distance to the existing helpers and adds it to the cluster (Lines 12,13). In other

Algorithm 1 Clustering Algorithm

Initialize $\mathcal{H} = \{1, 2, ..., H\}$, `loop1` $= 0$, `loop2` $= 0$, $g = 0$;
Input: Neighborhood information and distances $a_{hh'}, d_{hh'} \forall h, h' \in \mathcal{H}$.
while `loop1` $= 0$ **do**
    Find $\min_{h,h' \in \mathcal{H}} \{d_{h,h'}\}$
    if $a_{hh'} = 1$ **then**
        $g = g + 1$
        New clique: $\mathcal{G}_g = \{h, h'\}$
        Update: $\mathcal{H} = \mathcal{H} \setminus \{h, h'\}$
        while `loop2`=0 **do**
            Find set: $\mathcal{N} = \{i | i \in \mathcal{H}, a_{ih} = 1, \forall h \in \mathcal{G}_g\}$
            if $\mathcal{N} \neq \emptyset$ **then**
                Find $i^* = \min_{i \in \mathcal{N}} \{\sum_{h \in \mathcal{G}_g} d_{ih}\}$
                Add $\mathcal{G}_g = \mathcal{G}_g \cup \{i^*\}$
                Update: $\mathcal{H} = \mathcal{H} \setminus \{i^*\}$
            else
                `loop2`=1;
            end if
        end while
    else
        `loop1`=0
    end if
end while
Return: All cliques $\mathcal{G}_1, \mathcal{G}_2, ..., \mathcal{G}_g$.

words, this algorithm attempts to find maximal cliques. Algorithm returns the set of helpers in each cluster. The remaining helpers cannot form any cliques, since they are isolated.

Our clustering algorithm addresses the "maximal clique" problem in the literature, where we form cliques that can not be enlarged. Forming maximal cliques facilitates allocating as many different contents as possible to a helper cluster, which facilitates content diversity. This provides great benefits especially in the case of low Zipf skewness α. This way, less popular contents can also be cached in a cluster. Our simulations verify the success of this approach, wherein the proposed algorithm performs almost as good as the MILP-based optimal solution. The maximal clique problem is NP-complete [24], [25]. However we need a easy-to-implement clustering algorithm that forms clusters as large as possible, in order to provide content diversity in cache placement.

We did not include a fully-described distributed clustering protocol in this paper. However such a clustering algorithm can be approximately implemented in a distributed manner. For example, the authors in [26] also used the idea of maximal cliques and proposed a distributed clustering algorithm. The nodes first collect the connectivity information with their neighbors. Each node waits a random amount of time before starting a cluster advertisement. This time duration is inversely proportional to the number of neighbors. Hence the node with highes number of neighbors advertises first as a cluster head. A node receiving an advertisement joins in a cluster if it satisfies the single-hop condition with every existing node in the cluster. The helper that has the most number of ordinary user neighbors becomes the clusterhead.

Algorithm 2 presents the pseudocode of the cache placement algorithm. Algorithm consists of a for loop (Lines 3-21), where cache placement of cluster occurs in each turn. Let N_h be the number of helpers in cluster g (Line 4). The total cache capacity in this cluster is denoted by `total`. The for loop in Lines 6-11 determines of how many copies of each content will be cached in the cluster. Line 7 finds an integer number for each content. The sum of these numbers barely exceeds the total cache capacity. Line 12 sorts helpers in the

Algorithm 2 Cache Placement Algorithm

Initialize $x_{hc} = 0, \forall h = 1, ..., H, c = 1, ..., C$
Input: All cliques $\mathcal{G}_1, \mathcal{G}_2, ..., \mathcal{G}_g$, Content popularity: p_{uc}, Number of contents C, Cache capacity C_c
for all clusters g **do**
  Number of helpers in the cluster $N_h = |\mathcal{G}_g|$
  `total`$=N_h \times C_c$
  for i=1:C **do**
    $t_c = \min\left(N_h, \texttt{round}\left(\frac{p_{uc}}{p_{ui}}\right)\right), \forall c$
    if then $\sum_{c=1}^{C} t_c \geq \texttt{total}$
      `Exit loop`
    end if
  end for
  Sort helpers in cluster g as $(1), (2), ..., (N_h)$, according to number of neighboring ordinary users
  for $(h) = (1) : (N_h)$ **do**
    for c **do**=1:C
      if $t_c > 0$ **then**
        Set $x_{(h),c} = 1$
        $t_c = t_c - 1$
      end if
    end for
  end for
end for
for all helpers $h \notin \bigcup_{i=1}^{g} \mathcal{G}_i$ **do**
  Set $x_{hc} = 1$ for $c = 1, 2, ..., C_c$
end for
Return: $x_{hc}, \forall h = 1, ..., H, c = 1, ..., C$

cluster, where the helpers with more ordinary neighbors gain priority. The loop in Lines 13-20 scans the helpers according to their priority. This loop fills the caches of the helpers starting from the most popular content. Finally the loop in line 22-24 fills the helpers that are not included in any cluster (i.e. isolated helpers). Each of these isolated helpers cache the most popular contents. The content popularity information can be obtained from a centralized server, or it can be estimated by the base station and informed to the helpers.

The following examples shows the main loop of the algorithm (Lines 3-21) in action:

Suppose that there are $C = 10$ contents with popularities (skewness $\alpha = 1$) $\mathbf{p_{uc}} = [0.341, 0.171, 0.114, 0.085, 0.068, 0.057, 0.049, 0.043, 0.038, 0.034]$. Let us consider a cluster of 3 users, $\{1, 2, 3\}$ where each user can cache $C_c = 2$ contents. So total cache capacity of this cluster is $3 \times 2 = 6$ contents and t_c stands for the temporary cached contents by the corresponding cluster.

- We first divide $\mathbf{p_{uc}}$ by 0.341 and round. The results becomes $t_c = [1100000000]$. The sum is smaller than 6.
- We then divide $\mathbf{p_{uc}}$ by 0.171 and round. The results becomes $t_c = [2111000000]$. The sum is smaller than 6.
- We then divide $\mathbf{p_{uc}}$ by 0.114 and round. The results becomes $t_c = [3211110000]$. The sum is greater than 6. We exit the loop in Lines 6-11.
- Assume that, after performing the operation in Line 12, the order does not change.
- Node 1 caches contents 1,2. Update t_c as $t_c = [2111110000]$.
- Node 2 caches contents 1,2. Update t_c as $t_c = [1011110000]$.
- Node 3 caches contents 1,3. Update t_c as $t_c = [0001110000]$.
- All the capacity in the cluster is filled. The algorithm passes to the next cluster.

Table 1: Simulation Parameters

Simulation Parameters	**Value**
Number of contents (C)	[30, 40]
Number of ordinary users (U)	[300,1000]
Number of helper users (H)	[40, 45, 50, 55, 60]
Zipf Skewness Parameter (α)	[0.6, 1, 1.5]
D2D Maximum Range (R_D)	50 meters
Cache capacity (C_c)	[2, 4, 6, 8] contents
Radius of cellular area (R_{max})	250 meters
Content Size (S_c)	1 unit

As seen in this example, content 1, which is very popular, is cached at each helper in the cluster. Content 2 is cached in two helpers and content 3 is cached in only one helper. Other contents are not cached. This method both maximizes hit probability and provides content diversity.

5. Simulation Results

In this section we will compare the performances of the proposed algorithm with that of the optimal solution, along with some benchmarks. Table 1 shows the simulation parameters. We will compare the following methods,

- MILP-based optimal solution.
- Proposed clustering-based algorithm.
- Popularity based caching: This is a simple scheme, where each helper independently caches the most popular contents.
- One Copy: This is a hybrid scheme, where helpers are clustered using the proposed clustering method. However any content is cached by at most one helper in a cluster.

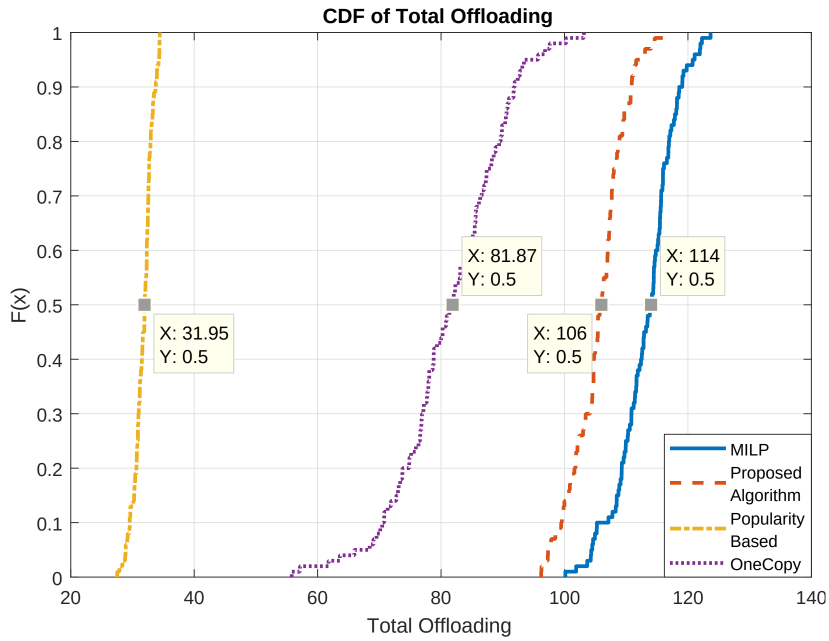

Figure 1: Cumulative Distribution Function for Total Offloading (U=300, C=30, C_c=4, α=0.6)

Figure 1 shows the empirical cumulative distribution function (CDF) of the total offloading for 300 ordinary users, 50 helpers, 30 contents where $\alpha = 0.6$. Median offloading results show that our proposed method provides an offload within 7% of the optimal.This is a very promising result. Moreover, the offloading provided by the popularity based caching method is only one third of the optimal. This shows the importance of clustering. One Copy algorithm is somewhere in between, but it can still provide only 70% of the optimal offloading. This shows the success of our proposed cache placement algorithm in providing high hit probability.

Figure 2 shows the performance for higher skewness of content popularities where $\alpha = 1.5$. This points to a more imbalanced popularity profile. In this case a small fraction of contents occupy a large fraction of total popularity. Our algorithm performs even better in this case, within 5% of the optimal performance. Popularity based caching can only provide half of the optimal offloading.

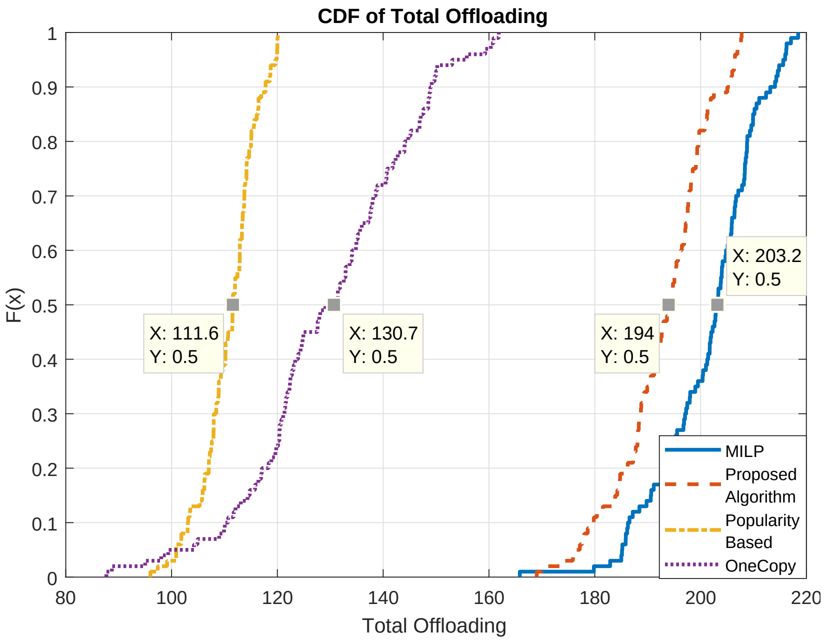

Figure 2: Cumulative Distribution Function for Total Offloading (U=300, C=30, C_c=4, α=1.5)

Relative performance of One Copy method is worse (64% of the optimal median offloading). This is because One Copy method caches unnecessary contents, that have low popularity.

Figures 3 and 4 show the empirical CDF of the offloading for the case of higher ($U = 1000$) ordinary users. As expected, this results in increased total offloading for all compared methods. Our proposed algorithm approaches even closer to the optimum, with a median offloading performance within 4% and 2.5% of the optimal for $\alpha = 0.6$ and 1.5.

In order to form Table 2, we ran the simulations for our proposed algorithm and optimal MILP solution for 100 uniformly distributed random topology which includes helpers and ordinary users. Then divided the total offloading of the proposed algorithm to that of the optimal. Table shows the value of this obtained quantity for different cache capacities and number of helpers. Here *max*, denotes the 95th percentile of the considered topologies, while *min* de-

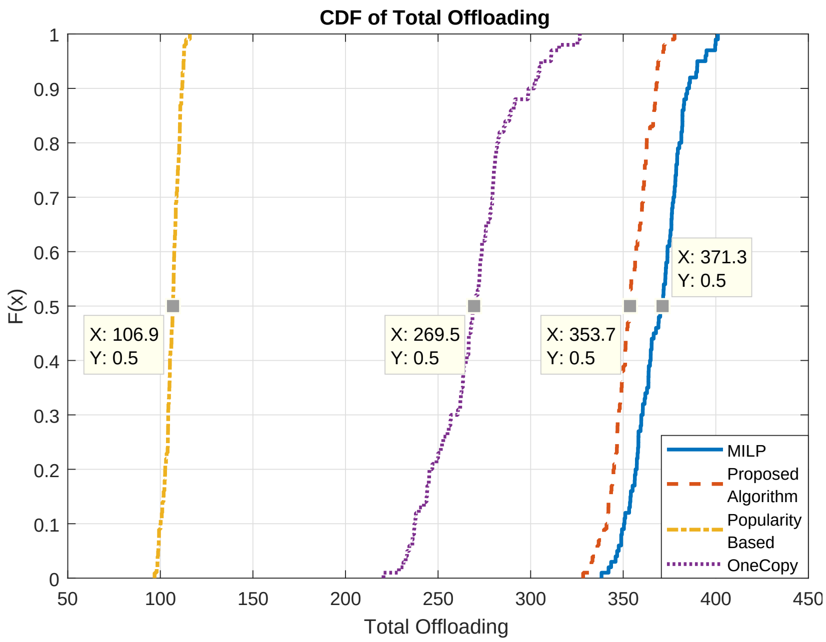

Figure 3: Cumulative Distribution Function for Total Offloading (U=1000, C=30, C_c=4, α=0.6)

notes the 5th percentile. The results in the Table indicate that for all cache sizes and number of helpers, the proposed algorithm almost always perform within 10% of the optimal. Our algorithm especially approaches the optimal for lower number of helpers and a larger cache capacity.

6. Conclusions

In this work we consider the problem of optimal cache placement using D2D communications in a cellular system. We propose a cache placement scheme, where the helper nodes are first clustered and then the caches in a cluster are collectively utilized in order to provide higher hit probability and content diversity. The results clearly show that our algorithm is very successful and provides close-to-optimal offloading uniformly for all values of simulation parameters. The proposed algorithm has two main advantages. First of all it is robust, meaning that it performs close to optimal for various values of number of users,

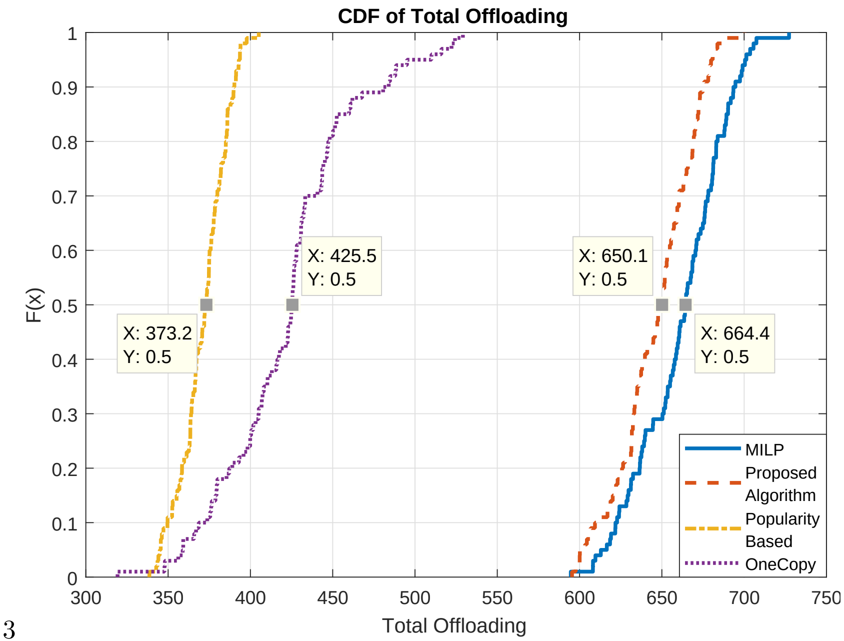

Figure 4: Cumulative Distribution Function for Total Offloading (U=1000, C=30, C_c=4, α=1.5)

helper, Zipf skewness parameter and cache sizes. Secondly it is amenable to distributed implementation. There are various distributed clustering algorithms in the literature, which can be applied in order to form helper cliques.

7. Extensions and Future Work

There are number of issues that can be considered as future work,

7.1. Jointly Optimal Cache Placement and Resource Allocation for Helpers

In this work we assumed that D2D transmissions do not cause interference to cellular transmissions. This can be possible either by using an orthogonal frequency resources (overlay) or using a different technology (e.g. WiFi). In future we plan to consider a D2D underlay, where D2D transmissions interfere with the cellular transmissions [14].

Table 2: Proposed Algorithm vs. MILP for U=1000, C=40, α=1

		Number of Helpers H				
		40	45	50	55	60
Cache Size C_c	2	max: 99.39%	max: 98.13%	max: 96.39%	max: 97.62%	max: 95.82%
		min: 90.55%	min: 89.32%	min: 89.26%	min: 88.78%	min: 87.43%
	4	max: 99.77%	max: 99.66%	max: 99.19%	max: 98.77%	max: 98.42%
		min: 93.68%	min: 93.17%	min: 92.98%	min: 92.36%	min: 91.63%
	6	max: 99.66%	max: 99.58%	max: 99.60%	max: 99.41%	max: 98.92%
		min: 94.77%	min: 94.01%	min: 94.32%	min: 93.32%	min: 92.65%
	8	max: 99.60%	max: 99.57%	max: 99.39%	max: 96.65%	max: 99.09%
		min: 95.24%	min: 94.81%	min: 94.73%	min: 93.99%	min: 93.16%

7.2. Estimating the Content Popularities

In this work we assume that the base station perfectly knows the content popularities and feeds it back to the helpers. However, our proposed algorithm is also able to work in case of estimated probabilities. Moreover, content popularity can also be measured by the helpers. Helpers in a cluster can collect content requests and share their statistics with their cluster head. Cluster head then perform content placement and tell the other cluster members about which content to cache.

Online stochastic learning can also be used in order to implement and online and adaptive caching mechanism. This method has been recently used as a channel and power allocation mechanism in D2D transmissions [27]. Online stochastic learning can be implemented as follows: The helper first randomly caches a content, then measures the request rate and resulting offloading. The probability of caching of a content increases in the next stage depending on its request rate. After some iterations, caching probability of a content approaches 1, which ends the iterations. Although such algorithms require a high number of iterations to converge, they can be preferred for a distributed mechanism, which requires no intervention from the base station.

In real applications content popularity can also be geographically varying. In such a case, estimating the popularity centrally by the base station will be suboptimal. In this case estimation of popularity separately at each cluster is a good choice. Since the helpers in a cluster are geographically close, content popularity in their region should be similar. Our proposed algorithm is also applicable in case of geographically varying content popularity.

7.3. Distributed Clustering

With its current form our proposed clustering algorithm for helpers is a centralized algorithm. However it can be modified as a distributed algorithm. In [22] an architecture is proposed that uses WiFi-Direct jointly with LTE for D2D communications. In their architecture, cellular users build a WiFi-Direct cluster and the clusterhead serves as a bridge between LTE and WiFi Direct networks [28]. In [28] an architecture is proposed, where a device can be in more than one group as a client or group owner. This way a helper clique can create a separate group for content placement. Each helper can also create their own group with neighboring ordinary users. Besides, an ordinary user can be in all groups of neighboring helpers. When a user requests a content, the closest helper that has the content serves the user. Since WiFi-direct uses a separate and unlicensed frequency band, this would result in significant reduction in LTE bandwidth demand.

7.4. Time Varying User Location and Content Popularity Profiles

In this work we consider a single-shot scenario, where clustering and cache placement is performed for a given set of user locations and content popularity profile. In reality , location of mobile users and helpers would change frequently, which requires cluster maintenance and cache content replacement. Developing cache placement methods for mobile users and time varying content profiles is a subject of future work.

Acknowledgement

This research is supported by the Turkish Scientific and Technological Research Council, Research Project Award No. 215E123.

[1] I. F. Akyildiz, S. Nie, S.-C. Lin, M. Chandrasekaran, 5g roadmap: 10 key enabling technologies, Computer Networks 106 (2016) 17–48.

[2] I. O&M, Application of d2d in 5g networks, ZTE White Paper.

[3] E. Bastug, M. Bennis, M. Debbah, Living on the edge: The role of proactive caching in 5g wireless networks, IEEE Communications Magazine 52 (8) (2014) 82–89. doi:10.1109/MCOM.2014.6871674.

[4] N. Golrezaei, A. G. Dimakis, A. F. Molisch, Wireless device-to-device communications with distributed caching, in: 2012 IEEE International Symposium on Information Theory Proceedings, 2012, pp. 2781–2785. doi:10.1109/ISIT.2012.6284029.

[5] K. Shanmugam, N. Golrezaei, A. G. Dimakis, A. F. Molisch, G. Caire, Femtocaching: Wireless content delivery through distributed caching helpers, IEEE Transactions on Information Theory 59 (12) (2013) 8402–8413. doi:10.1109/TIT.2013.2281606.

[6] N. Golrezaei, K. Shanmugam, A. G. Dimakis, A. F. Molisch, G. Caire, Wireless video content delivery through coded distributed caching, in: 2012 IEEE International Conference on Communications (ICC), 2012, pp. 2467–2472. doi:10.1109/ICC.2012.6364526.

[7] M. Ji, G. Caire, A. F. Molisch, The throughput-outage tradeoff of wireless one-hop caching networks, IEEE Transactions on Information Theory 61 (12) (2015) 6833–6859. doi:10.1109/TIT.2015.2490226.

[8] J. Jiang, S. Zhang, B. Li, B. Li, Maximized cellular traffic offloading via device-to-device content sharing, IEEE Journal on Selected Areas in Communications 34 (1) (2016) 82–91. doi:10.1109/JSAC.2015.2452493.

[9] D. Malak, M. Al-Shalash, J. G. Andrews, A distributed auction policy for user association in device-to-device caching networks, in: Personal, Indoor, and Mobile Radio Communications (PIMRC), 2017 IEEE 28th Annual International Symposium on, IEEE, 2017, pp. 1–6.

[10] R. Lan, W. Wang, A. Huang, H. Shan, Device-to-device offloading with proactive caching in mobile cellular networks, in: 2015 IEEE Global Communications Conference (GLOBECOM), 2015, pp. 1–6. doi:10.1109/GLOCOM.2015.7417337.

[11] F. Shen, K. Hamidouche, E. Bastug, M. Debbah, A stackelberg game for incentive proactive caching mechanisms in wireless networks, in: 2016 IEEE Global Communications Conference (GLOBECOM), 2016, pp. 1–6. doi:10.1109/GLOCOM.2016.7841550.

[12] Z. Chen, Y. Liu, B. Zhou, M. Tao, Caching incentive design in wireless d2d networks: A stackelberg game approach, in: 2016 IEEE International Conference on Communications (ICC), 2016, pp. 1–6. doi:10.1109/ICC.2016.7511284.

[13] Z. Zheng, L. Song, Z. Han, G. Y. Li, H. V. Poor, A stackelberg game approach to proactive caching in large-scale mobile edge networks, IEEE Transactions on Wireless Communications 17 (8) (2018) 5198–5211. doi:10.1109/TWC.2018.2839111.

[14] A. C. Kazez, T. Girici, Interference-aware distributed device-to-device caching, in: 2017 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom), 2017, pp. 1–5. doi:10.1109/BlackSeaCom.2017.8277676.

[15] R. Amer, M. M. Butt, M. Bennis, N. Marchetti, Delay analysis for wireless d2d caching with inter-cluster cooperation, in: GLOBECOM 2017 - 2017 IEEE Global Communications Conference, 2017, pp. 1–7. doi:10.1109/GLOCOM.2017.8254693.

[16] H. Hsu, K. C. Chen, A resource allocation perspective on caching to achieve low latency, IEEE Communications Letters 20 (1) (2016) 145–148. doi:10.1109/LCOMM.2015.2499193.

[17] B. Chen, C. Yang, Z. Xiong, Optimal caching and scheduling for cache-enabled d2d communications, IEEE Communications Letters 21 (5) (2017) 1155–1158.

[18] D. Malak, M. Al-Shalash, J. G. Andrews, Spatially correlated content caching for device-to-device communications, IEEE Transactions on Wireless Communications 17 (1) (2018) 56–70.

[19] A. Asadi, V. Mancuso, Network-assisted outband d2d-clustering in 5g cellular networks: theory and practice, IEEE Transactions on Mobile Computing 16 (8) (2017) 2246–2259.

[20] M. Afshang, H. S. Dhillon, P. H. J. Chong, Modeling and performance analysis of clustered device-to-device networks, IEEE Transactions on Wireless Communications 15 (7) (2016) 4957–4972.

[21] M. Afshang, H. S. Dhillon, P. H. J. Chong, Fundamentals of cluster-centric content placement in device-to-device networks, in: Globecom Workshops (GC Wkshps), 2015 IEEE, IEEE, 2015, pp. 1–6.

[22] A. Asadi, V. Mancuso, Wifi direct and lte d2d in action, in: Wireless Days (WD), 2013 IFIP, IEEE, 2013, pp. 1–8.

[23] A. A. Abbasi, M. Younis, A survey on clustering algorithms for wireless sensor networks, Computer communications 30 (14-15) (2007) 2826–2841.

[24] I. M. Bomze, M. Budinich, P. M. Pardalos, M. Pelillo, The maximum clique problem, in: Handbook of combinatorial optimization, Springer, 1999, pp. 1–74.

[25] A. Dharwadker, The clique algorithm (2006).

[26] K. Biswas, V. Muthukkumarasamy, E. Sithirasenan, Maximal clique based clustering scheme for wireless sensor networks, in: Intelligent Sensors, Sensor Networks and Information Processing, 2013 IEEE Eighth International Conference on, IEEE, 2013, pp. 237–241.

[27] S. Maghsudi, S. Stańczak, Channel selection for network-assisted d2d communication via no-regret bandit learning with calibrated forecasting, IEEE Transactions on Wireless Communications 14 (3) (2015) 1309–1322.

[28] C. Casetti, C. F. Chiasserini, L. C. Pelle, C. Del Valle, Y. Duan, P. Giaccone, Content-centric routing in wi-fi direct multi-group networks, in: World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2015 IEEE 16th International Symposium on a, IEEE, 2015, pp. 1–9.