Caching Policy for Cache-Enabled D2D Communications by Learning User Preference

Binqiang Chen, *Student Member, IEEE*, and Chenyang Yang, *Senior Member, IEEE*

***Abstract*— Prior works in a designing caching policy do not distinguish content popularity with user preference. In this paper, we illustrate the caching gain by exploiting individual user behavior in sending requests. After showing the connection between the two concepts, we provide a model for synthesizing user preference from content popularity. We then optimize the caching policy with the knowledge of user preference and activity level to maximize the offloading probability for cache-enabled device-to-device communications, and develop a low-complexity algorithm to find the solution. In order to learn user preference, we model the user request behavior resorting to probabilistic latent semantic analysis, and learn the model parameters by the expectation maximization algorithm. By analyzing a Movielens data set, we find that the user preferences are less similar, and the activity level and topic preference of each user change slowly over time. Based on this observation, we introduce a prior knowledge-based learning algorithm for user preference, which can shorten the learning time. Simulation results show a remarkable performance gain of the caching policy with user preference over existing policy with content popularity, both with realistic data set and synthetic data validated by the real data set.**

***Index Terms*— User preference, content popularity, caching policy, D2D, machine learning, data analysis.**

I. Introduction

CACHING at the wireless edge can improve network throughput and energy efficiency as well as user experience dramatically [2]–[6]. Owing to the small storage size of each node at the wireless edge, say base station (BS) [2], [4] or user device [3], caching in a proactive manner is critical to achieve the performance gain, where future user demand statistics is exploited [2], [7]. By caching at BSs, backhaul traffic can be offloaded and backhaul cost can be reduced. By caching at users, wireless traffic can be further offloaded from peak time to off-peak time [8]. To boost the cache hit rate by precaching contents at each user that has very limited cache size, cache-enabled device-to-device (D2D) communications and coded-multicast strategy are proposed [2], [5], [8].

To heap the proactive caching gain, various caching policies have been optimized with diverse objectives for different networks. Most existing works assume known content popularity, defined as the probability distribution that every content in a catalog is requested by all users. For example, the policies for caching at BSs were optimized to minimize average download delay in [9] and to maximize coverage probability in [10]. Coded caching policy was optimized to maximize the average fractional offloaded traffic and average ergodic rate of small-cell networks in [11]. The policies for caching at users were optimized to maximize the offloading gain in [12] and [13] and to minimize the average delay of users with different group popularity in [14], all for cache-enabled D2D communications. In these works, every user is assumed to request files according to content popularity. However, in practice a user actually sends requests according to its own preference, which may not be identical to content popularity.

To implement above-mentioned proactive caching policies, content popularity needs to be predicted [15]. Popularity prediction has been investigated for diverse applications such as advertisement, where content popularity is defined as the accumulated number of requests every content in a catalog received or the request arrival rate for every content. Numerous methods have been proposed [16]. By using these prediction methods, the content popularity defined with probability in [2]–[6], [9]–[14] can be obtained as a ratio of the number of requests for each file to the number of all requests. In cellular networks, the number of users in a cell is much less than that in a region covered by a content server, and a mobile user may send requests to more than one BS. Since popularity depends on the group of users who send requests, the local popularity in a cell may differ from the global popularity observed at a server. Designing proactive caching policy at wireless edge with global popularity prediction leads to low cache hit rate [7].

In [17], local popularity was learned at small BS as the request arrival rate by a multi-arm bandit algorithm, and the predicted popularity was used for caching policy optimization. The prediction is based on the cumulative growth method [16] and under the assumption that only the requests for already cached files can be observed, hence the learning rate is slow. In existing cellular networks, however, the requested contents from users cannot be observed at BSs, and hence the local popularity is unable to be learned at the BS. In [18],

Manuscript received January 3, 2018, revised May 28, 2018, and July 22, 2018; accepted July 27, 2018. Date of publication August 6, 2018; date of current version December 14, 2018. This work was supported by the National Natural Science Foundation of China (NSFC) under Grants 61731002, 61671036, and 61429101. This paper was presented in part at the IEEE VTC Spring 2017 [1]. The associate editor coordinating the review of this paper and approving it for publication was N. Tran. *(Corresponding author: Chenyang Yang.)*

The authors are with the School of Electronics and Information Engineering, Beihang University, Beijing 100191, China (e-mail: chenbq@buaa.edu.cn; cyyang@buaa.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCOMM.2018.2863364

the content popularity was predicted with a real dataset measured at the mobile core network. By converting the number of requests received at each BS into rating, the local popularity was predicted by a collaborative filtering technique, matrix factorization [19], with which the files with largest ratings are cached.

A. Motivation and Contributions

A large body of priori works for caching at the wireless edge assume that all users send their requests according to content popularity, and do not differentiate content popularity from user preference. The following facts, which are widely recognized in the communities studying recommendation problem and analyzing user behavior with real data, are largely overlooked in the community of studying caching at the wireless edge: (i) as a demand statistic of multiple users, content popularity is not the same as the preference of each individual user [20], (ii) only a small portion of users are active in creating traffic [21]. In practice, user preferences are heterogeneous, although they may exhibit similarity to a certain extent. The caching policy designed under unrealistic assumptions inevitably yields performance loss. While caching based on local popularity can partially address the user preference heterogeneity issue [17], [18], in dense cellular networks with overlapped coverage, local popularity in each cell depends on the associated users that further depends on the caching policy.

In this paper, we investigate the gain of optimizing caching policy by learning user preference and activity level over content popularity. To this end, we take cache-enabled D2D communications as an example system and offloading probability as an example objective. Because there are different definitions in the domains of computer science and wireless communications, we first define user preference and activity level as well as content popularity to be used in this paper. To show the gain of caching with user preference and analyze where the gain comes from, we provide a probabilistic model to synthesize user preference from content popularity by introducing similarity among user preferences. We then formulate an optimization problem with known user preference and activity level to maximize offloading probability. Since the problem is NP-hard, a local optimal algorithm is proposed to reduce the complexity, which achieves at least 1/2 optimality. In order to learn user preference and activity level, we model user request behavior resorting to probabilistic latent semantic analysis (pLSA) originally proposed for natural language processing [22], whose model parameters can be learned by using approximate inference methods such as expectation maximization (EM) [23]. With the help of pLSA model to decompose the user behavior into different components and inspired by an observation obtained from analyzing a real dataset (i.e., activity level and topic preference change slowly over time), we provide a prior knowledge based algorithm, which can quickly learn user preference.

The major contributions of this paper are summarized as follows:

- We illustrate the caching gain of exploiting user preference and activity level by optimizing a caching policy for D2D communications, and predict the behavior of each individual user by estimating model parameters of pLSA. We introduce a prior knowledge based algorithm to learn user preference, which shows the potential of transfer learning.
- We characterize the connection between content popularity and user preference, provide a probabilistic model for synthesizing user preference from content popularity, and validate the method by the MovieLens 1M dataset [24]. We analyze the statistics of individual user behavior in sending reuqests by the real dataset, considering that caching gain highly depends on practical user behavior.
- Simulation results with both synthesized data and MovieLens dataset show remarkable performance gain of the caching policy with user preference over that with local content popularity, no matter the user demands are assumed known or learned from historical requests.

B. Related Works

By assuming user preferences as Zipf distributions with different ranks, caching policies were optimized to minimize the average delay of cache-enabled D2D communications in [25] and to maximize the cache hit rate of mobile social networks in [26]. Both works do not validate the assumption for user preference, do not show the gain over caching with popularity, and assume that all users are with identical activity level. Until now, there exists no method to synthesize user preference validated by real dataset, and the gain of caching with user preference is unknown.

There are few works that consider the relation between content popularity and user preference. In [27], local popularity is computed as a weighted average of preferences for the users associated with each BS, where the weight is the number of requests sent by each user and the user preference was assumed as uniformly distributed. Then, the most popular files at each BS were cached. Differing from this early work, we characterize the connection between the collective and the individual user request behavior in a probabilistic framework, and illustrate the gain of exploiting user preference over popularity by optimizing a caching policy.

These priori works assume known user preference [25]–[27]. To facilitate proactive caching, user preference needs to be predicted, which is a key task in recommendation problem. Collaborative filtering is the most commonly used technique to predict user preference [19], which can be mainly classified into memory based method including user-based and item-based approaches, and model based method that is based on models with parameters estimated from historical records [28]. Typical model based methods employ matrix factorization, latent Dirichlet allocation [29], and pLSA [30] as models [19]. For recommendation problem, user preference is defined as the rating that a user gives for a file, such as $0 \sim 5$ or simply "like" and "dislike". Collaborative filtering methods predict the ratings for unrated contents of each user, which however cannot be used in optimization for wireless caching, where various metrics are

in statistical sense [2]–[6], [9]–[14]. For caching problem, user preference can be defined as request probability, but there is no widely-accepted approach to translate the rating into the request probability. In this paper, we resort to pLSA to model and predict user preference, which is originally developed for classification in automatic indexing [22] and then is applied to predict ratings in [30].

The rest of the paper is organized as follows. Section II provides the relation between content popularity and user preference, and a model to synthesize user preference. Section III optimizes the caching policy with known user preference. Section IV presents the learning algorithms. Section V analyzes the statistics of user demands from and validates the synthetic model by a MovieLens dataset. Section VI provides simulation results. Section VII concludes the paper.

II. Content Popularity and User Preference

In this section, we first define content popularity, user preference and activity level to be used in the following, show their connection, and then provide a probabilistic model with a free parameter to synthesize user preference.

A. Definition and Relationship

Consider a content library $\mathcal{F} = \{\mathrm{f}_1, \mathrm{f}_2, \ldots, \mathrm{f}_F\}$ consisting of F files that K users in an area may request, where f_f denotes the fth file.

Content popularity is defined as the probability distribution that each file in the library is requested by all users, denoted as $\mathbf{p} = [p_1, p_2, \ldots, p_F]$, where $p_f \triangleq P(\mathrm{f}_f)$ is the probability that f_f is requested, $\sum_{f=1}^{F} p_f = 1$, $p_f \in [0,1]$, and $1 \le f \le F$. If the content popularity is computed from the users from a sub-area of the considered whole area, then $\mathbf{p}$ is called local popularity.

User preference is defined as the conditional probability distribution that a user requests a file given that the user sends a request, denoted as $\mathbf{q}_k = [q_{1|k}, q_{2|k}, \ldots, q_{F|k}]$ for the kth user (denoted as u_k)), where $q_{f|k} \triangleq P(\mathrm{f}_f|\mathrm{u}_k)$ is the conditional probability that the kth user requests the fth file when the user sends a file request, $\sum_{f=1}^{F} q_{f|k} = 1$, $q_{f|k} \in [0,1]$, $1 \le f \le F$ and $1 \le k \le K$. We use a $K \times F$ matrix $\mathbf{Q} = (q_{f|k})^{K \times F}$ to denote the preferences of all users.

Activity level is defined as the probability that a request is sent by a user, denoted as $w_k \triangleq P(\mathrm{u}_k)$ for the kth user, which reflects how active the user is, where $\sum_{k=1}^{K} w_k = 1$ and $w_k \in [0,1]$. Then, the vector $\mathbf{w} = [w_1, w_2, \ldots, w_K]$ denotes the activity level distribution of the K users.

Content popularity $\mathbf{p}$ reflects the collective request behavior of a group of users, while $\mathbf{q}_k$ and w_k characterize the individual request behavior of the kth user. To show their connection, we consider a $K \times F$ user-content matrix $\mathbf{N} = (n_{k,f})^{K \times F}$ [19], where $n_{k,f}$ represents the number of requests sent by u_k for f_f during a period. Denote $N = \sum_{k=1}^{K} \sum_{f=1}^{F} n_{k,f}$ as the overall number of requests sent by all the K users for all the F files, $n_f = \sum_{k=1}^{K} n_{k,f}$ as the total number of requests sent by all users for f_f (i.e., the sum of all elements in the fth column of $\mathbf{N}$), and $n_k = \sum_{f=1}^{F} n_{k,f}$ as the total number of requests sent by u_k for all files (i.e., the sum of all elements in the kth row of $\mathbf{N}$). Considering that n_f, n_k, and $n_{k,f}$ are respectively following multinomial distributions with F, K, and F parameters, it is not hard to show that n_f/N, n_k/N, and $n_{k,f}/n_k$ are respectively the maximum likelihood estimate of p_f, w_k and $q_{f|k}$. From their definitions, we have $\sum_{k=1}^{K} \underbrace{n_k/N}_{w_k} \underbrace{n_{k,f}/n_k}_{q_{f|k}} = \sum_{k=1}^{K} \frac{n_{k,f}}{N} = \underbrace{n_f/N}_{p_f}$, and hence each element of $\mathbf{p}$ can be expressed as the average of user preferences weighted by their activity levels,

$$p_f = \sum_{k=1}^{K} w_k q_{f|k}, \quad 1 \le f \le F. \tag{1}$$

In practice, $\mathbf{q}_k \neq \mathbf{q}_m$, and hence $q_{f|k} \neq p_f$, despite that users may have similar preferences, say for popular contents. We can use cosine similarity to reflect the similarity of preferences between two users, which is frequently used in collaborative filtering [19] and defined as

$$\mathrm{sim}(\mathbf{q}_k, \mathbf{q}_m) = \frac{\sum_{f=1}^{F} q_{f|m} q_{f|k}}{\sqrt{\sum_{f=1}^{F} q_{f|m}^2 \sum_{f=1}^{F} q_{f|k}^2}}. \tag{2}$$

To show the similarity among K users, we can define average cosine similarity as

$$\mathbb{E}[\mathrm{sim}(\mathbf{q}_k, \mathbf{q}_m)] = \frac{2}{K(K-1)} \sum_{k,m} \frac{\sum_{f=1}^{F} q_{f|k} q_{f|m}}{\sqrt{\sum_{f=1}^{F} q_{f|k}^2 \sum_{f=1}^{F} q_{f|m}^2}}. \tag{3}$$

B. Modeling and Synthesizing User Preference

Content popularity can be modeled as a Zipf distribution according to the analyses for many real datasets [18], [31], [32]. The probability that the fth file is requested by all users is $p_f = f^{-\beta} / \sum_{j=1}^{F} j^{-\beta}, 1 \le f \le F$, where the files are indexed in descending order of popularity.

User preference model obtained from real datasets is unavailable in the literature so far. Inspired by the method in [7] to synthesize local popularity of a cell from that of a core network, we represent users and files in a shared one-dimensional latent space, which bears the same spirit as the latent factor model widely applied in collaborative filtering [29], [33]. To connect with content popularity, we model user preference as the following generative process:

- u_k is associated with a feature value X_k, which is randomly selected from $[0,1]$.
- f_f is associated with a feature value Y_f, which is again chosen uniformly from $[0,1]$.
- The joint probability that the fth file is requested by the kth user is given by

$$P(\mathrm{u}_k, \mathrm{f}_f) = w_k q_{f|k} = p_f \frac{g(X_k, Y_f)}{\sum_{k'=1}^{K} g(X_{k'}, Y_f)}, \tag{4}$$

and then u_k's activity level is $w_k = \sum_{\mathrm{f}_f \in \mathcal{F}} P(\mathrm{u}_k, \mathrm{f}_f)$ and its preference is $q_{f|k} = P(\mathrm{u}_k, \mathrm{f}_f)/w_k$, where $g(X_k, Y_f) \in [0,1]$ is a kernel function used to control the correlation between the kth user and the fth file.

When $g(X_k, Y_f) = 0$, the fth file will never be requested by the kth user. When $g(X_k, Y_f) = 1$, the file is a preferred file of the user.

Intuitively, the value of X_k can be interpreted as the likelihood that the kth user prefers a topic, and the value of Y_f can be interpreted as the likelihood that the fth file belongs to a topic. Various kernel function can be applied, e.g., Gaussian, logarithmic and power kernels. To control the average similarity among the user preferences by introducing a parameter α in kernel function, we choose power kernel with expression $g(X_k, Y_f) = (1 - |X_k - Y_f|)^{(\frac{1}{\alpha^3} - 1)} \in [0, 1]$ $(0 < \alpha \leq 1)$, which exhibits a nearly linear relation between $\mathbb{E}[\text{sim}(\mathbf{q}_k, \mathbf{q}_m)]$ and α in a wide range.

Remark 1: When $\alpha = 1$, $g(X_k, Y_f) = 1$ for $\forall k, f$. From (4) we can see that all user preferences are equal to the content popularity, and from (3) we can obtain $\mathbb{E}[\text{sim}(\mathbf{q}_k, \mathbf{q}_m)] = 1$. When $\alpha \to 0$, $g(X_k, Y_f) \to 0$ for $X_k \neq Y_f$, and $g(X_k, Y_f) = 1$ only for $X_k = Y_f$. Because X_k and Y_f are uniformly chosen from $[0, 1]$, i.e., $\mathbb{P}(X_k = Y_f) \to 0$, we have $g(X_k, Y_f)g(X_{k'}, Y_f) \to 0, k \neq k'$. Then, according to (4) and (3), $\mathbb{E}[\text{sim}(\mathbf{q}_k, \mathbf{q}_m)] \to 0$, i.e., no user has the same preference. When α is small, $g(X_k, Y_f)$ is low, which implies that u_k is interested in a small number of files. Because each user randomly chooses feature values, the interested file sets among different users are less overlapped, and hence the preference similarity is low.

This model will be validated later by a real dataset. It can be used for synthesizing data of request of each user for any given content popularity with flexibly controlled user preference similarity, which differs from pLSA that can be used for predicting the individual behavior.

III. Caching Policy Optimization: An Illustration

In this section, we illustrate how to optimize the caching policy with known request behavior of each individual user. For comparison, we also provide the corresponding caching policy optimization problem with known content popularity, whose solution reflects the existing policy in literature. To focus on the performance gain brought by distinguishing user preference from content popularity, we consider a simple objective: the offloading gain of D2D communications. To reduce the time complexity in finding the solution, we provide a local optimal algorithm.

A. System Model

Multiple BSs in the area are connected to core network via backhaul to serve the K uniformly distributed users, which constitutes a set of users $\mathcal{U} = \{\mathrm{u}_1, \mathrm{u}_2, \ldots, \mathrm{u}_K\}$ that request the files in content library $\mathcal{F}$. Assume that each file is with same file size, but the results are applicable for general case with different sizes [7] by dividing each file into chunks of approximately equal size.[1] Each single-antenna user has a local cache to store M files, and can act as a helper to share

[1]We can also formulate another optimization problem with different file sizes, which can be shown as a knapsack problem.

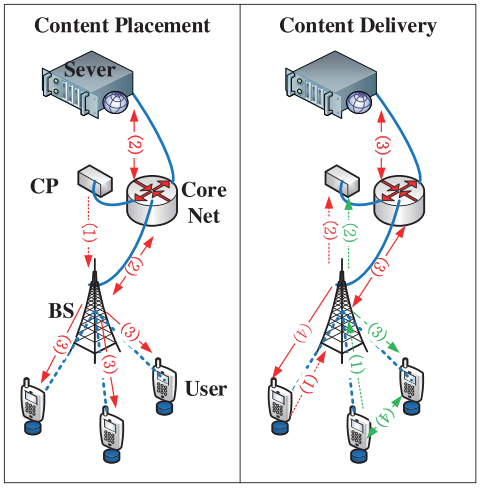


Fig. 1. Illustration of D2D communication systems with proactive caching, where the green arrows represent the procedure for cache-hit requests and the red arrows are for cache-miss requests.

files via D2D link. To provide high rate transmission with low energy cost at each user device, we consider a user-centric D2D communication protocol as in [13]. A helper can serve as a D2D transmitter and send its cached files to a user only if their distance is smaller than a collaboration distance r_c, which reflects the coverage of the helper. Each BS is aware of the cached files at and the locations of the users, and coordinates the D2D communications.

Proactive caching consists of content delivery and content placement phases, as shown in Fig. 1. Assume that a central processor (CP) can identify user requests and record the requests history of users. The CP needs to be deployed in mobile core networks, such that the requested contents of users can be observed and the requests can be recorded. To determine where to deploy the CP, coverage area and computational cost also need to be considered.

The procedure for content delivery is as follows. (1) Each user initiates requests according to its own preference. If a user can find its requested file in local cache (i.e., fetching locally), it directly retrieves the file with zero delay. If not, the user sends the request to BS. (2) The BS reports the request to the CP, which identifies the requested file and informs the file index to the BS, and records the requests of every user. (3) If the BS finds that the file is cached in the local caches of helpers adjacent to the user, it informs the request to the closest helper, and assists to establish a D2D link between the user and the closest helper. Otherwise, the BS fetches the file via backhaul. (4) For a cache-hit request, the user fetches the file via the established D2D link. For a cache-miss request, the BS transmits the file to the user. For simplicity, both fetching locally and via D2D link are called fetching via D2D links.

The procedure for content placement is as follows. (1) The CP learns the user preferences $\mathbf{Q}$ and activity levels $\mathbf{w}$ from historical requests records $\mathbf{N}$, and then optimizes the caching policy and informs the cached files of the users to the BSs. (2) A BS fetches the files from the server. (3) The BS refreshes

the caches at users via unicast or multicast during off-peak time,[2] say every several hours or every day, noticing that traffic load varies on the timescale of hours as measured by real cellular data [21].

B. Caching Policy Optimization With Individual Request Behavior

We consider deterministic caching policy,[3] denoted as a vector $\mathbf{c}_k = [c_{k,1}, c_{k,2}, \ldots, c_{k,F}]$ for the kth user, where $c_{k,f} = 1$ if f_f is cached at u_k, $c_{k,f} = 0$ otherwise, and $\sum_{f=1}^{F} c_{k,f} \leq M$. Denote the caching policy for all users as $\mathbf{C} = (c_{k,f})^{K\times F}$.

We optimize the caching policy to maximize *offloading probability*, defined as the probability that a user can fetch the requested file via D2D links. It reflects the average fraction of requests able to be offloaded and hence the offloading gain introduced by cache-enabled D2D communications. The resulting caching policy is applicable to any transmission scheme in content delivery phase, but is with different performance when different interference management schemes are employed, as shown via simulations later.

When optimizing caching policy in the content placement phase, it is hard to know where a mobile user will be located in the content delivery phase. Therefore, it is hard to know when and how long the users will contact. Fortunately, data analysis shows that users always periodically reappear at the same location with high probability [34]. Consequently, it is reasonable to assume that the contact probability is known *a priori* [26]. Let $\mathbf{A} = (a_{i,j})^{K\times K}$ represent the contact probability among users, where $a_{i,j} \in [0,1]$ is the probability that the distance between the ith user and the jth user is less than r_c. When all users do not move, $a_{i,j} = 0$ or 1.

In practice, adjacent helpers may have overlapped coverage. Since different helpers serve different groups of users, which depend not only on r_c but also on the cached files at the adjacent helpers, the "local popularity" observed at a helper differs from that observed at another helper and relies on the caching policy. As a result, the caching policy can not be designed based on the "local popularity".

Denote $p^{\mathrm{d}}_{k,f}(\mathbf{A},\mathbf{C})$ as the probability that the kth user can fetch the fth file via D2D links given contact probability $\mathbf{A}$ and caching policy $\mathbf{C}$. The complementary probability of $p^{\mathrm{d}}_{k,f}(\mathbf{A},\mathbf{C})$ is the probability that the fth file is not cached at any users in proximity to the kth user, which can be derived as $\prod_{m=1}^{K}(1-a_{k,m}c_{m,f})$. Then, we can obtain the offloading probability as

$$\begin{aligned} &p_{\mathrm{off}}(\mathbf{Q},\mathbf{w},\mathbf{A},\mathbf{C}) \\ &\quad= \sum_{k=1}^{K} w_k \sum_{f=1}^{F} q_{f|k} p^{\mathrm{d}}_{k,f}(\mathbf{A},\mathbf{C}) \\ &\quad= \sum_{k=1}^{K} w_k \sum_{f=1}^{F} q_{f|k} \left(1 - \prod_{m=1}^{K}(1-a_{k,m}c_{m,f})\right). \end{aligned} \tag{5}$$

With known user preference and activity level, the caching policy can be optimized to maximize the offloading probability by solving the following problem,

$$\mathbf{P1}: \max_{c_{m,f}} \; p_{\mathrm{off}}(\mathbf{Q},\mathbf{w},\mathbf{A},\mathbf{C}) \tag{6a}$$

$$s.t. \; \sum_{f=1}^{F} c_{m,f} \leq M, c_{m,f} \in \{0,1\}, \tag{6b}$$

$$1 \leq m \leq K, \quad 1 \leq f \leq F. \tag{6c}$$

Remark 2: If all users are with equal activity level and equal preference, then (5) becomes

$$\begin{aligned} p_{\mathrm{off}}(\mathbf{Q},\mathbf{w},\mathbf{A},\mathbf{C}) &= \frac{1}{K}\sum_{f=1}^{F} p_f \sum_{k=1}^{K} p^{\mathrm{d}}_{k,f}(\mathbf{A},\mathbf{C}) \\ &\triangleq p^{\mathrm{pop}}_{\mathrm{off}}(\mathbf{p},\mathbf{A},\mathbf{C}). \end{aligned} \tag{7}$$

Remark 3: If the collaboration distance $r_c \to \infty$, then $a_{k,m} = 1$, and (5) becomes

$$\begin{aligned} &p_{\mathrm{off}}(\mathbf{Q},\mathbf{w},\mathbf{A},\mathbf{C}) \\ &\quad= \sum_{f=1}^{F} \left(1 - \prod_{m=1}^{K}(1-c_{m,f})\right) \sum_{k=1}^{K} w_k q_{f|k} \\ &\quad= \sum_{f=1}^{F} \left(1 - \prod_{m=1}^{K}(1-c_{m,f})\right) p_f. \end{aligned}$$

It is easy to show that $p_{\mathrm{off}}(\mathbf{Q},\mathbf{w},\mathbf{A},\mathbf{C}) = p^{\mathrm{pop}}_{\mathrm{off}}(\mathbf{p},\mathbf{A},\mathbf{C})$ in this extreme case where D2D links can be established between any two users in the area even with heterogeneous user preferences.

If the assumptions in the two remarks hold, then the "local popularity" observed at every helper will be identical, which is equal to the content popularity of the area with the K users. In practice, the assumptions are not true, hence exploiting user preference for caching is necessary.

With known content popularity, the caching policy is optimized by maximizing $p^{\mathrm{pop}}_{\mathrm{off}}(\mathbf{p},\mathbf{A},\mathbf{C})$ in (7) under constraint (6c), called problem **P2**,[4] which is actually a special case of **P1**.

By setting the contact probability $a_{i,j}$ as 1 or 0 (i.e., all users are static), we can obtain a special case of problem **P2**, which has the same structure as a NP-hard problem formulated in [9]. Since **P2** is a special case of **P1**, problem **P1** is NP-hard. As a consequence, it is impossible to find its global optimal solution within polynomial time. By using similar way of the

[2]Multicast transmission can be applied for predownloading files to users in the content placement phase when some files need to be cached in multiple users according to the caching policy. For example, user 1 caches files A, B and C, and user 2 caches files B, C, and D. Then, files B and C can be placed to the two users via multicast, and files A and D are conveyed via unicast. Precaching at users consumes wireless resource in this phase. The required resource for the preference-based caching policy may be higher than the popularity-based caching policy. We will address this issue via simulations later.

[3]We do not consider probabilistic caching policy, which is designed under the assumption that a group of nodes share the same caching distribution [9]–[14], and hence is not appropriate for a system with heterogeneous user preferences.

[4]Problem **P2** slightly differs from the problems in [12] and [13], where the future user location is exactly known in [12] and completely unknown in [13] when optimizing caching policy, but the contact probability is known in **P2**.

proof in [9], it is not hard to prove that **P1** is equivalent to maximizing a submodular function over matroid constraints. Thus, we can resort to greedy algorithm, which is commonly used to provide a solution achieving at least $\frac{1}{2}$ optimality for such type of problem [35].[5]

The greedy algorithm starts with zero elements for the caching matrix, i.e., $\mathbf{C} = (0)^{K\times F}$. In each step, the value of one element in $\mathbf{C}$ is changed from zero to one with the maximal incremental caching gain defined as

$$\begin{aligned} &v_{\mathbf{C}}(m,f) \\ &\quad = p_{\text{off}}(\mathbf{Q},\mathbf{w},\mathbf{A},\mathbf{C}|_{c_{m,f}=1}) - p_{\text{off}}(\mathbf{Q},\mathbf{w},\mathbf{A},\mathbf{C}) \\ &\quad \overset{(a)}{=} \sum_{k=1}^{K} w_k q_{f|k} \left(p^{\text{d}}_{k,f}\left(\mathbf{A},\mathbf{C}|_{c_{m,f}=1}\right) - p^{\text{d}}_{k,f}\left(\mathbf{A},\mathbf{C}\right)\right), \end{aligned} \quad (8)$$

where (a) follows by substituting (5), $\mathbf{C}$ is the caching matrix at previous step, and $\mathbf{C}|_{c_{m,f}=1}$ is the matrix by letting $c_{m,f}=1$ in $\mathbf{C}$. The algorithm is summarized in Algorithm 1.

Algorithm 1 Greedy Algorithm

Require: $\mathbf{A}$; $\mathbf{w}$; $\mathbf{Q}$; Initialize: Caching matrix $\mathbf{C} = (0)^{K\times F}$; Files not cached at the mth user $\bar{\mathcal{C}}_m \leftarrow \{\mathrm{f}_1, \mathrm{f}_2, \ldots, \mathrm{f}_F\}$; Users with residual storage space $\mathcal{U}_0 \leftarrow \{\mathrm{u}_1, \mathrm{u}_2, \ldots, \mathrm{u}_K\}$;

1: **while** $\mathcal{U}_0 \neq \emptyset$ **do**
2: $[m^*, f^*] = \arg\max_{\mathrm{u}_m\in\mathcal{U}_0, \mathrm{f}_f\in\bar{\mathcal{C}}_m} v_{\mathbf{C}}(m,f)$; $\mathbf{C} = \mathbf{C}|_{c_{m^*,f^*}=1}$; $\bar{\mathcal{C}}_{m^*} \leftarrow \bar{\mathcal{C}}_{m^*} \setminus \mathrm{f}_{f^*}$;
3: **if** $|\bar{\mathcal{C}}_{m^*}| = F - M$ **then**
4: $\mathcal{U}_0 \leftarrow \mathcal{U}_0 \setminus \mathrm{u}_{m^*}$;
5: **end if**
6: **end while**
7: $\mathbf{C}^* = \mathbf{C}$;

Ensure: Caching matrix $\mathbf{C}^*$.

The loops in step 1 of Algorithm 1 take KM iterations, because there are totally KM files that are possible to be cached at all users. The step 2 for finding the element in $\mathbf{C}$ that introduces the highest incremental caching gain takes at most KF iterations. For each time of computing $v_{\mathbf{C}}(m,f)$ in (8), the time complexity is $O(K^2)$, and thus computing all $v_{\mathbf{C}}(m,f)$ is $O(K^3F)$. Hence the total time complexity for Algorithm 1 is $O(KM(KF+K^3F)) = O(K^2FM(K^2+1))$, which is high especially when the numbers of users K and files F are large.

C. A Low Complexity Algorithm With 1/2 Optimality Guarantee

Since the greedy algorithm is with high time complexity, finding a low-complexity algorithm is worthwhile for practice use. In what follows, we propose an alternating optimization algorithm, which improves the offloading gain at every iteration and converges to a local optimal solution.

[5]The best algorithm with polynomial time complexity for such problem can achieve $(1-\frac{1}{e})$ optimality guarantee, which is based on continuous greedy process and pipage rounding techniques [36]. However, when $K = 100$ and $F = 3000$ in the considered setting as detailed later, its complexity is $O((FK)^8) = O(6.5\times 10^{43})$, which is too complex for our problem.

To be specific, we fix the caching policy at users $\mathbf{c}_m (m \neq k', 1 \leq m \leq K)$ and optimize $\mathbf{c}_{k'}$. Then, from problem **P1** we obtain the optimization problem with respect to $\mathbf{c}_{k'}$ as

$$\begin{aligned} \mathbf{P1}': \max_{c_{k',f}} \; & f_{\text{off}}(\mathbf{c}_{k'}) = \sum_{k=1}^{K} w_k \sum_{f=1}^{F} q_{f|k}\Big(1 - \prod_{m\neq k'}^{K} \\ & (1 - a_{k,m}c_{m,f})(1 - a_{k,k'}c_{k',f})\Big) \\ s.t. \; & \sum_{f=1}^{F} c_{k',f} \leq M, c_{k',f} \in \{0,1\}, \\ & 1 \leq f \leq F. \end{aligned} \quad (9)$$

Proposition 1: $\mathbf{P1}'$ can be solved with polynomial time complexity $O(F(K^2+M))$.

Proof: See Appendix A. ∎

Based on the proof of Proposition 1, we propose an algorithm to iteratively solve problem $\mathbf{P1}'$ by changing k' from 1 to K until convergence. The algorithm starts with a given initial value of $\mathbf{C}$. In every iteration, by fixing $\mathbf{c}_m (m \neq k', 1 \leq m \leq K)$, it respectively computes the offloading gain introduced by caching the fth file at the k'th user

$$b_{k',f} = \sum_{k=1}^{K} w_k q_{f|k} a_{k,k'} \prod_{m=1,m\neq k'}^{K} (1 - a_{k,m}c_{m,f}), \quad 1 \leq f \leq F, 1 \leq k' \leq K. \quad (10)$$

Then, the algorithm finds the file indices with the maximal M values of $b_{k',f}$ to constitute a set $\mathcal{I}_{k'}$, and obtain $\mathbf{c}^*_{k'}$ as

$$c^*_{k',f} = \begin{cases} 1, & f \in \mathcal{I}_{k'} \\ 0, & f \notin \mathcal{I}_{k'}. \end{cases} \quad (11)$$

Algorithm 2 A Low Complexity Algorithm

Require: $\mathbf{A}$; $\mathbf{w}$; $\mathbf{Q}$; Initialize: Random caching $\mathbf{c}^{(0)}_m (1 \leq m \leq K)$, $t \leftarrow 1$;

1: **repeat**
2: **for** $k' = 1, 2, \ldots, K$ **do**
3: Based on $\mathbf{c}^{(t-1)}_m (m \neq k')$, compute $b_{k',f}$ by (10), constitute $\mathcal{I}_{k'}$ and obtain $\mathbf{c}^*_{k'}$ by (11).
4: $\mathbf{c}^{(t)}_{k'} = \mathbf{c}^*_{k'}$;
5: **end for**
6: **until** We obtain the converged result $(\mathbf{c}^{(t)}_m = \mathbf{c}^{(t-1)}_m, 1 \leq m \leq K)$

Ensure: Caching matrix $\mathbf{c}^{(t)}_m$.

The detailed algorithm is presented in Algorithm 2. The loops in step 2 take K iterations. Step 3 is with complexity $O(F(K^2+M))$ according to Proposition 1. Hence the total time complexity of Algorithm 2 is $O(t_{A2}KF(K^2+M))$, where t_{A2} is the number of iterations for step 1.

Proposition 2: Algorithm 2 monotonically increases the objective function of problem **P1** and finally converges to achieve at least $1/2$ optimality.

Proof: See Appendix B. ∎

It is noteworthy that Algorithm 1 and Algorithm 2 can also solve **P2** by letting $q_{f|k} = p_f, \forall k, f$ in $\mathbf{Q}$. Solutions for **P1**

and **P2** obtained with Algorithm 1 are called **S1** $-$ $A1$ and **S2** $-$ $A1$, and solutions using Algorithm 2 for **P1** and **P2** are called **S1** $-$ $A2$ and **S2** $-$ $A2$, respectively.

IV. Learning User Preference and Activity Level

In this section, we first use pLSA to model content request behavior of an individual user. We then learn the model parameters by maximizing likelihood function, either without the pLSA model for comparison or with the model using the EM algorithm, which is efficient for maximal likelihood (ML) parameter estimation with latent variables [23]. Finally, we present a prior knowledge based algorithm to learn user preference.

A. Modeling Individual User Behavior in Requesting Contents

To characterize the request behavior of a user, pLSA associates each request with a topic, which may be unobservable but can be intuitively interpreted as comedy, adventure, etc.

By introducing latent topic set $\mathcal{Z} = \{\mathrm{z}_1, \mathrm{z}_2, \ldots, \mathrm{z}_Z\}$ with cardinality $|\mathcal{Z}| = Z$, pLSA associates each topic $\mathrm{z}_j \in \mathcal{Z}$ with each possible user request, i.e., $\mathrm{u}_k \in \mathcal{U}$ requests $\mathrm{f}_f \in \mathcal{F}$. Specifically, the request of each user can be modeled with three model parameters using the following steps:

- A request is sent by u_k with probability $P(\mathrm{u}_k)$ (i.e., *activity level*),
- u_k chooses a topic z_j with probability $P(\mathrm{z}_j|\mathrm{u}_k)$ (i.e., *topic preference*, $\sum_{j=1}^{Z} P(\mathrm{z}_j|\mathrm{u}_k) = 1$),
- u_k requests f_f in topic z_j with probability $P(\mathrm{f}_f|\mathrm{z}_j)$, $\sum_{f=1}^{F} P(\mathrm{f}_f|\mathrm{z}_j) = 1$, where *conditional independence* assumption is used. In particular, conditioned on a request being sent by u_k who chooses topic z_j, u_k chooses f_f with probability $P(\mathrm{f}_f|\mathrm{z}_j, \mathrm{u}_k) = P(\mathrm{f}_f|\mathrm{z}_j)$, i.e., $P(\mathrm{f}_f|\mathrm{u}_k) = \sum_{\mathrm{z}_j \in \mathcal{Z}} P(\mathrm{f}_f|\mathrm{z}_j)P(\mathrm{z}_j|\mathrm{u}_k)$. In other words, no matter which user sends a request and selects topic z_j, the user will request f_f with probability $P(\mathrm{f}_f|\mathrm{z}_j)$.

Then, the joint probability that u_k requests f_f can be expressed as

$$\begin{aligned} P(\mathrm{u}_k, \mathrm{f}_f) &= P(\mathrm{u}_k)P(\mathrm{f}_f|\mathrm{u}_k) \\ &= P(\mathrm{u}_k) \sum_{\mathrm{z}_j \in \mathcal{Z}} P(\mathrm{f}_f|\mathrm{z}_j)P(\mathrm{z}_j|\mathrm{u}_k). \end{aligned} \tag{12}$$

B. Learning Individual User Behavior in Requesting Contents

According to the ML principle, we can learn $P(\mathrm{u}_k), P(\mathrm{f}_f|\mathrm{z}_j)$ and $P(\mathrm{z}_j|\mathrm{u}_k)$ with the observed number of requests $n_{k,f}$ by maximizing the following log-likelihood function [37]

$$\begin{aligned} \mathcal{L} &= \sum_i \log P(\mathrm{u}_{i_\mathrm{u}}, \mathrm{f}_{i_\mathrm{f}}) = \underbrace{\sum_{\mathrm{u}_k \in \mathcal{U}} \sum_{\mathrm{f}_f \in \mathcal{F}} n_{k,f} \log P(\mathrm{u}_k, \mathrm{f}_f)}_{(a)} \\ &= \underbrace{\sum_{\mathrm{u}_k \in \mathcal{U}} \sum_{\mathrm{f}_f \in \mathcal{F}} n_{k,f} \log P(\mathrm{u}_k) \sum_{\mathrm{z}_j \in \mathcal{Z}} P(\mathrm{f}_f|\mathrm{z}_j)P(\mathrm{z}_j|\mathrm{u}_k)}_{(b)}, \end{aligned} \tag{13}$$

where the ith sample corresponding to the event that the i_uth user requests the i_fth file, and in (b) the pLSA model is applied.

1) ML Algorithm Without pLSA Model: By maximizing the log likelihood function in (a) of (13) without the pLSA model, it is not hard to obtain that

$$\hat{P}(\mathrm{u}_k, \mathrm{f}_f) = \frac{n_{k,f}}{\sum_{k'=1}^{K} \sum_{f=1}^{F} n_{k',f}}. \tag{14}$$

Then, the activity level and user preference can be learned as $\hat{w}_k = \hat{P}(\mathrm{u}_k) = \sum_{f=1}^{F} \hat{P}(\mathrm{u}_k, \mathrm{f}_f)$ and $\hat{q}_{f|k} = \frac{\hat{P}(\mathrm{u}_k, \mathrm{f}_f)}{\hat{P}(\mathrm{u}_k)} = \frac{\hat{P}(\mathrm{u}_k, \mathrm{f}_f)}{\sum_{f=1}^{F} \hat{P}(\mathrm{u}_k, \mathrm{f}_f)}$, respectively. This algorithm is actually a simple frequency-count prediction, which can serve as a baseline for learning activity level and user preference.

Remark 4: If we directly predict w_k and $q_{f|k}$ using (14), the number of parameters to estimate is KF. By using the pLSA as in (b) of (13), the number of parameters is reduced from KF to $K + KZ + ZF = Z(K + F) + K$, where K parameters are for learning activity level, KZ parameters are for topic preference, and ZF parameters are for $P(\mathrm{f}_f|\mathrm{z}_j)$. With less number of parameters to estimate, a learning algorithm can converge more quickly.

2) ML Algorithm With pLSA Model: To maximize the log-likelihood function in (b) of (13), we first rewrite the function as

$$\begin{aligned} \mathcal{L} = &\underbrace{\sum_{\mathrm{u}_k \in \mathcal{U}} n_k \log P(\mathrm{u}_k)}_{(a)} \\ &+ \underbrace{\sum_{\mathrm{u}_k \in \mathcal{U}} \sum_{\mathrm{f}_f \in \mathcal{F}} n_{k,f} \log \sum_{\mathrm{z}_j \in \mathcal{Z}} P(\mathrm{f}_f|\mathrm{z}_j)P(\mathrm{z}_j|\mathrm{u}_k)}_{(b)}, \end{aligned} \tag{15}$$

where $n_k = \sum_{\mathrm{f}_f \in \mathcal{F}} n_{k,f}$. It is not hard to see that the terms in (a) and (b) can be independently maximized. The activity level of u_k can be learned by maximizing term (a) in (15) as

$$\hat{w}_k = \hat{P}(\mathrm{u}_k) = \frac{n_k}{\sum_{k'=1}^{K} \sum_{f=1}^{F} n_{k',f}}, \tag{16}$$

which is the same as that obtained from (14). The other two model parameters $P(\mathrm{f}_f|\mathrm{z}_j)$ and $P(\mathrm{z}_j|\mathrm{u}_k)$ can be learned by maximizing term (b) in (15) using the EM algorithm as follows [37].

Starting from randomly generated initial values for the model parameters $P(\mathrm{z}_j|\mathrm{u}_k)$ and $P(\mathrm{f}_f|\mathrm{z}_j)$, $1 \le j \le Z$, $1 \le f \le F$ and $1 \le k \le K$, the EM algorithm alternates two steps: expectation (E) step and maximization (M) step.

In the E-step, the posterior probability is computed for latent variable z_j with current estimation of the parameters as

$$\hat{P}(\mathrm{z}_j|\mathrm{u}_k, \mathrm{f}_f) = \frac{\hat{P}(\mathrm{z}_j|\mathrm{u}_k)\hat{P}(\mathrm{f}_f|\mathrm{z}_j)}{\sum_{\mathrm{z}_{j'} \in \mathcal{Z}} \hat{P}(\mathrm{z}_{j'}|\mathrm{u}_k)\hat{P}(\mathrm{f}_f|\mathrm{z}_{j'})}, \tag{17}$$

which is the probability that f_f requested by u_k belongs to topic z_j.

In the M-step, given $\hat{P}(\mathrm{z}_j|\mathrm{u}_k,\mathrm{f}_f)$ computed by the previous E-step, the two parameters are updated as

$$\hat{P}(\mathrm{f}_f|\mathrm{z}_j)=\frac{\sum_{\mathrm{u}_k\in\mathcal{U}} n_{k,f}\hat{P}(\mathrm{z}_j|\mathrm{u}_k,\mathrm{f}_f)}{\sum_{\mathrm{u}_k\in\mathcal{U}}\sum_{\mathrm{f}_{f'}\in\mathcal{F}} n_{k,f'}\hat{P}(\mathrm{z}_j|\mathrm{u}_k,\mathrm{f}_{f'})},$$
$$\hat{P}(\mathrm{z}_j|\mathrm{u}_k)=\frac{\sum_{\mathrm{f}_f\in\mathcal{F}} n_{k,f}\hat{P}(\mathrm{z}_j|\mathrm{u}_k,\mathrm{f}_f)}{n_k}. \quad (18)$$

By alternating (17) with (18), the EM algorithm converges to a local maximum of log-likelihood function. Then, the preference of the kth user for the fth file can be learned as

$$\hat{q}_{f|k}=\hat{P}(\mathrm{f}_f|\mathrm{u}_k)=\sum_{\mathrm{z}_j\in\mathcal{Z}}\hat{P}(\mathrm{f}_f|\mathrm{z}_j)\hat{P}(\mathrm{z}_j|\mathrm{u}_k). \quad (19)$$

3) Prior Knowledge Based Algorithm to Learn User Preference: Video files in real world website always have topic information, e.g., movies are labeled with *comedy*, *drama* and so on.

Intuitively, the topic preference and activity level of a user change slowly over time, and hence can be regarded as invariant. This will be validated later by real dataset. Thanks to the pLSA model, such intuition naturally yields a *prior knowledge based algorithm* to learn user preference by exploiting the activity level and topic preference of a user learned previously during a much longer time than learning user preference, with the help of the topic information. This algorithm can be regarded as a parameter-transfer approach [38]. While the activity level $P(\mathrm{u}_k)$ and topic preference $P(\mathrm{z}_j|\mathrm{u}_k)$ can never be learned perfectly, we assume that they are known in order to show the potential of such transfer learning. Then, the user preference can be learned by only estimating $P(\mathrm{f}_f|\mathrm{z}_j)$, which can be obtained similarly as in (18),

$$\hat{P}(\mathrm{f}_f|\mathrm{z}_j)=\begin{cases}\dfrac{\sum_{\mathrm{u}_k\in\mathcal{U}} n_{k,f}\hat{P}(\mathrm{z}_j|\mathrm{u}_k,\mathrm{f}_f)}{\sum_{\mathrm{u}_k\in\mathcal{U}}\sum_{\mathrm{f}_{f'}\in\mathcal{F}} n_{k,f'}\hat{P}(\mathrm{z}_j|\mathrm{u}_k,\mathrm{f}_{f'})}, & \mathrm{f}_f\in\mathcal{F}_j\\ 0, & \mathrm{f}_f\notin\mathcal{F}_j,\end{cases} \quad (20)$$

where $\mathcal{F}_j$ is the set of files associated with topic $\mathrm{z}_j(1\le j\le Z)$, which is available on the video website. For instance, the movie *Forrest Gump* is associated with topics *comedy, romance* and *war* on the MovieLens.

The algorithm is presented in Algorithm 3. Step 2 takes KFZ times computation of posterior probability by (17), where each computation is with complexity $O(Z-1)$ and thus totally at most $O(KFZ(Z-1))$. Step 3 computes (20) with ZF times, each is with complexity $O(K(F+1))$. It is not hard to see that step 4 is with complexity $O(KFZ)$. Hence the total time complexity for Algorithm 3 is $O(t_{A3}KFZ(Z+F+1))$, where t_{A3} is the number of iterations for step 1.

V. User Request Behavior Analysis With MovieLens Dataset

The gain from caching highly depends on the user behavior in requesting contents. In this section, we use a real dataset to analyze topic preference and user preference, and validate the intuition in Section IV-B.3 as well as the user preference model provided in Section II.

Algorithm 3 Learning User Preference With Prior Knowledge

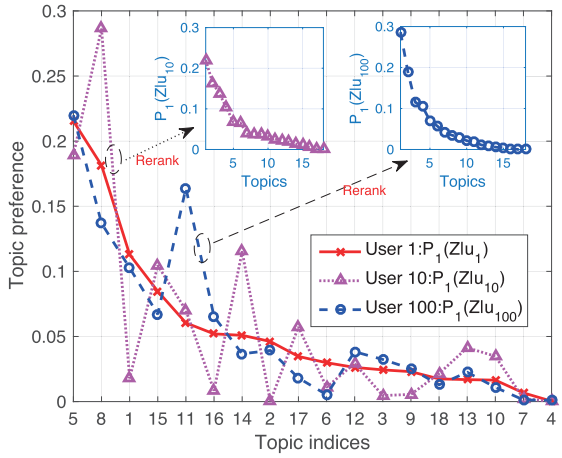
Require: $\mathbf{N}$; Z; $\mathcal{F}_j, 1\le j\le Z$; $\hat{P}(\mathrm{z}_j|\mathrm{u}_k)$; Stop condition $0<\epsilon<1$;
Initialize: $\hat{P}^{(0)}(\mathrm{f}_f|\mathrm{z}_j)$; Step $i\leftarrow 1$; Difference $\Delta\leftarrow\infty$; Log likelihood $\mathcal{L}(0)\leftarrow 0$;
1: **while** $\Delta>\epsilon$ **do**
2: Using $\hat{P}(\mathrm{z}_j|\mathrm{u}_k)$ and $\hat{P}^{(i-1)}(\mathrm{f}_f|\mathrm{z}_j)$ to compute $\hat{P}^{(i)}(\mathrm{z}_j|\mathrm{u}_k,\mathrm{f}_f)$ by (17);
3: Using $\hat{P}^{(i)}(\mathrm{z}_j|\mathrm{u}_k,\mathrm{f}_f)$ and $\mathcal{F}_j$ to compute $\hat{P}^{(i)}(\mathrm{f}_f|\mathrm{z}_j)$ by (20);
4: Compute log likelihood $\mathcal{L}(i)$ with $\hat{P}(\mathrm{z}_j|\mathrm{u}_k)$ and $\hat{P}^{(i)}(\mathrm{f}_f|\mathrm{z}_j)$ using term (b) in (15);
5: $\Delta=|\mathcal{L}(i)-\mathcal{L}(i-1)|$; $i\leftarrow i+1$;
6: **end while**
7: $\hat{q}_{f|k}\leftarrow\sum_{\mathrm{z}_j\in\mathcal{Z}}\hat{P}(\mathrm{f}_f|\mathrm{z}_j)\hat{P}(\mathrm{z}_j|\mathrm{u}_k)$ to compute $\hat{\mathbf{Q}}$;
Ensure: $\hat{\mathbf{Q}}$.

A. Statistical Results of User Demands

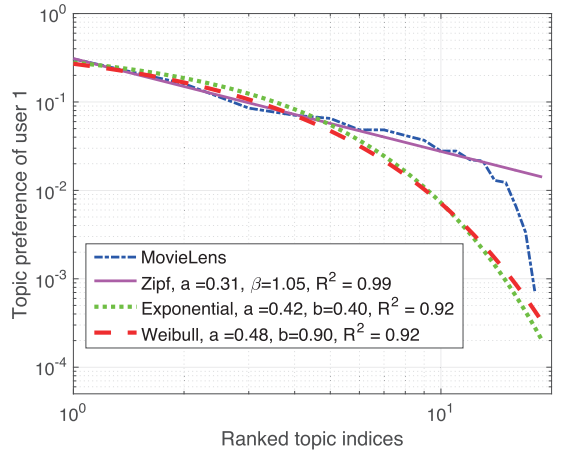
We use the *MovieLens 1M Dataset* [24] to reflect the request behavior, which contains 1000209 ratings for 3952 movies provided by 6026 users from the year of 2000 to 2003. Each sample of the dataset consists of user identity (ID), movie ID, rating and timestamp. Because users typically give ratings only after watching, we translate the rating record into the request record, i.e., when a user gives rating for a movie, we set the movie as requested by once by the user. Except for the ratings, MovieLens also provides topic information of movies. Every movie is associated with one, two or more topics from 18 topics, which include the genre of *action, adventure, animation, children's, comedy, etc.* and are denoted as $\mathrm{z}_1,\ldots,\mathrm{z}_{18}$. For instance, *Forrest Gump* is associated with topics *comedy* (z_5), *romance* (z_{14}) and *war* (z_{17}). From the topic information, we know that if the fth movie is not associated with the jth topic, users who select the jth topic will not request the fth file, i.e., we can set $P(\mathrm{f}_f|\mathrm{z}_j)=0$ in (20).

To analyze temporal evolution of user behavior, we sort all the 3952 movies according to their released date in ascendant order and then divide them into two subsets $\mathcal{F}_1$ and $\mathcal{F}_2$, where the file request matrices on $\mathcal{F}_1$ and $\mathcal{F}_2$ are $\mathbf{N}_1\in\mathbb{R}^{6040\times 1976}$ and $\mathbf{N}_2\in\mathbb{R}^{6040\times 1976}$, respectively. $\mathbf{N}_1$ and $\mathbf{N}_2$ can reflect user behavior on the previously released file subset $\mathcal{F}_1$ and the subsequently released file subset $\mathcal{F}_2$. Specifically, we analyze the following statistical results:

- *Topic preference:* Denote the topic preference of the kth user estimated on subsets $\mathcal{F}_1$ and $\mathcal{F}_2$ as $\mathbf{p}_1(\mathcal{Z}|\mathrm{u}_k)=[P_1(\mathrm{z}_1|\mathrm{u}_k),\ldots,\mathrm{P}_1(\mathrm{z}_Z|\mathrm{u}_k)]$ and $\mathbf{p}_2(\mathcal{Z}|\mathrm{u}_k)=[P_2(\mathrm{z}_1|\mathrm{u}_k),\ldots,\mathrm{P}_2(\mathrm{z}_Z|\mathrm{u}_k)]$, respectively. $P_1(\mathrm{z}_j|\mathrm{u}_k)$, and $P_2(\mathrm{z}_j|\mathrm{u}_k)$ are computed using (18) by EM algorithm with $\mathbf{N}_1$ and $\mathbf{N}_2$, respectively. To reflect the temporal dynamic of topic preference for the kth user, we evaluate the cosine similarity on the two subsets, $\mathrm{sim}(\mathbf{p}_1(\mathcal{Z}|\mathrm{u}_k),\mathbf{p}_2(\mathcal{Z}|\mathrm{u}_k))$.
- *User preference:* $q_{f|k}=\sum_{\mathrm{z}_j\in\mathcal{Z}}P(\mathrm{f}_f|\mathrm{z}_j)P(\mathrm{z}_j|\mathrm{u}_k)$ is obtained by EM algorithm on $\mathbf{N}_1$. The result obtained from $\mathbf{N}_2$ or $\mathbf{N}$ is similar, and hence is omitted for conciseness.

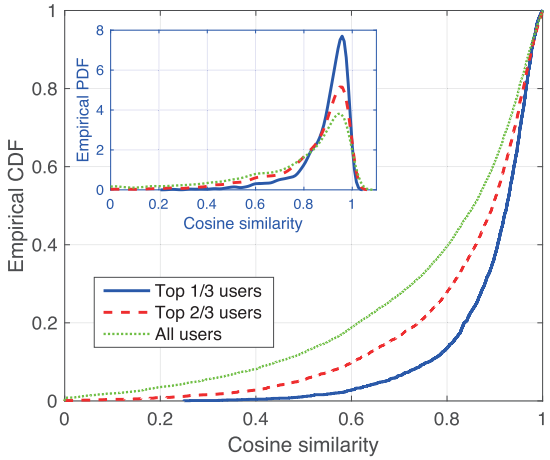


(a) Topic preferences of different users

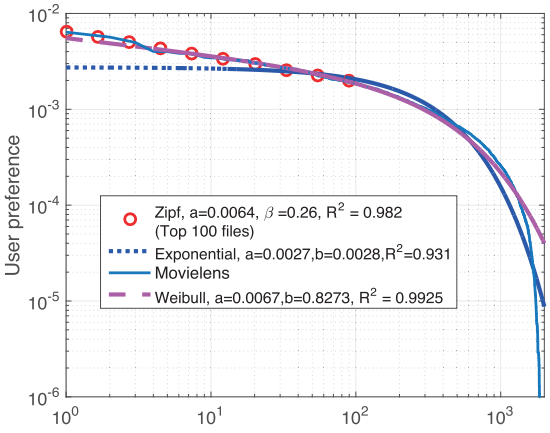


(b) Topic preference of the 1st user in descending order

Fig. 2. Topic preferences of different users and fitted distributions.



(a) Empirical CDF and PDF of cosine similarity between $\mathbf{p}_1(\mathcal{Z}|\mathrm{u}_k)$ and $\mathbf{p}_2(\mathcal{Z}|\mathrm{u}_k)$.



(b) User preference of the 1st user, where the files are ranked in descending order.

Fig. 3. Temporal dynamics of topic preferences and user preference of the 1st user.

- *Activity level:* $P_1(\mathrm{u}_k)$ and $P_2(\mathrm{u}_k)$ are computed using (16) with $\mathbf{N}_1$ and $\mathbf{N}_2$, respectively.

In Fig. 2(a), we show the topic preferences of the 1st, 10th and 100th users obtained from $\mathcal{F}_1$ (the results obtained from $\mathcal{F}_2$ are similar). The labels of x-axis are ranked in descending order according to $\mathbf{p}_1(\mathcal{Z}|\mathrm{u}_1)$. The topic preferences of the 10th and 100th users with re-ranked x-axis according to $\mathbf{p}_1(\mathcal{Z}|\mathrm{u}_{10})$ and $\mathbf{p}_1(\mathcal{Z}|\mathrm{u}_{100})$ are also provided in the inner-figures. As expected, topic preferences of different users differ. For example, the most favorite topic is *comedy* for the 1st and 100th user and *drama* for the 10th user. The topic preference of each user is skewed, which indicates that each user has strong preferences towards specific topics. In fact, we find that the topic preferences of all users in the dataset are skewed.

In Fig. 2(b), we show the topic preference of the 1st user and the fitted distributions in a log-log coordinate, where "Zipf", "Exponential" and "Weibull" are with functions $f(x) = \mathrm{a}x^{-\beta}$, $f(x) = \mathrm{a}e^{-\mathrm{b}x}$, and $f(x) = \mathrm{ab}x^{\mathrm{b}-1}e^{-\mathrm{a}x^{\mathrm{b}}}$, respectively. To evaluate the goodness of fit, we use the coefficient of determination (also called R-square) in linear regression, i.e., $R^2 = 1 - \frac{\sum_{i=1}^{S}(y_i - f(x_i))^2}{\sum_{i=1}^{S}(y_i - \bar{y})^2} \leq 1$, where S is the number of samples, (x_i, y_i) is the ith sample, and $\bar{y} = \frac{\sum_{i=1}^{S} y_i}{S}$ [39]. A large value of R^2 indicates a good fitting result. The parameters a, b and c for each function and R^2 are listed in the legends. We can see that the best fitted distribution is a Zipf distribution with parameter $\beta = 1.05$. We also fit the distributions for other users, and observe that the best fitted distributions differ for users, where Zipf distribution is the best for only 1425 users with parameter β uniformly distributed in $[0.5, 3]$. Yet for the most favorite several topics, Zipf distribution is always the best.

In Fig. 3(a), we show the empirical cumulative distribution function (CDF) and probability density function (PDF) of the cosine similarity between *topic preferences* over time of all users. We can see that 60% of all users have cosine similarity larger than 0.8, and almost 90% users among the top $1/3$ active users have cosine similarity larger than 0.8 (i.e., their topic preferences change slowly in the three years). Considering that the statistical results for active users with more requests are with high confidence level, this result indicates that topic

preferences can be approximated as invariant over time.[6] We also analyze the activity levels of users, and find that the distribution of activity levels is skewed as observed in [41]. Besides, the distribution of activity levels from the two subsets of data are similar, where the cosine similarity is 0.87, which indicates that the activity level of a user changes slightly over time. This validates the intuition in Section IV-B.3. Due to the space limitation, the results are not shown.

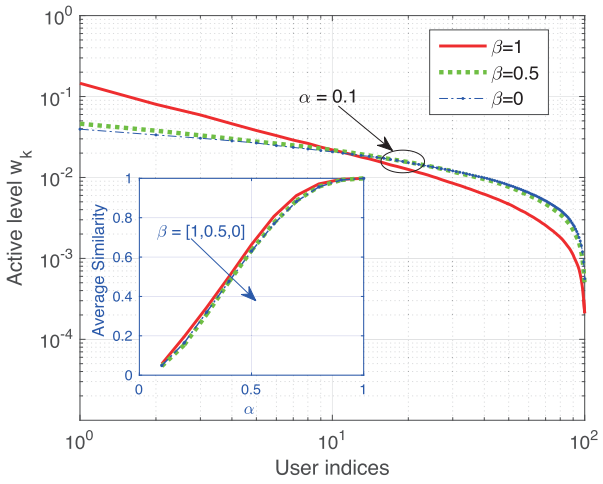
In Fig. 3(b), we show user preference of the 1st user and the fitted distributions. We find that the preferences of the 1st user (as well as other users) for the top 100 favorite files are close to Zipf distribution with parameters of β varying in $[0.2, 0.8]$, and the preferences for less favorite files have a truncated tail. This is because a user almost does not request the files belonging to its unfavorable topics. In summary, the users differ in the file set that they may request, and the skewness and ranking of their preferences. This is not consistent with the assumption in [25] and [26] that all user preferences follow Zipf distributions with same parameter but with different ranks. Besides, we observe that the average cosine similarity of preferences among different users on dataset $\mathbf{N}_1$ or $\mathbf{N}_2$ is $\mathbb{E}_{k,m}[\text{sim}(\mathbf{q}_k, \mathbf{q}_m)] \approx 0.4$.[7] This is mainly because the interested file sets of different users are less overlapped, recalling that the topic preferences of users differ.

Since the caching gain highly depends on the library size F [6], which is related to the number of users K, we also analyze the average number of files requested by given number of randomly chosen users from the dataset. The results show that the $F - K$ relation can be fitted well with power function $f(x) = \text{a}x^{\text{b}} + \text{c}$, where F increases with K first quickly and then slowly. For the *MovieLens* dataset, when $K = 100$, $F = 3000$.

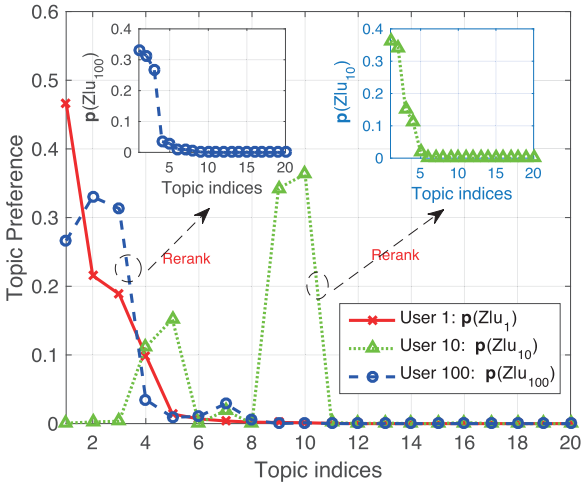
B. Validating Synthetic User Preference Model

Now, we validate the user preference model by comparing the results obtained from data synthesized by the generative process in Section II and those from the *MovieLens* dataset.

In Fig. 4(a), we first show the impact of parameter α in the kernel function. The inner-figure indicates that the synthetic user preference model can capture different levels of similarity among user preferences by adjusting α, while the Zip parameter β has negligible impact on the average cosine similarity. This seems counter-intuitive, since a more skewed popularity distribution seems to imply highly correlated user preferences. However, such an intuition comes from the implicit assumption that the users send their requests with equal probability (i.e., with identical activity level), which is not true in reality. From the figure we can observe that even when $\beta = 1$, α can be as small as 0.1. This is because few users are very active in sending file requests and have skewed user preference, who have large impact on content popularity according to (1). We can see that the distributions of user activity level are skewed, which agree well with the results obtained from the *MovieLens* dataset. In Fig. 4(b), we show the topic preference of the 1st, 10th and 100th users. The labels of x-axis are ranked according to $\mathbf{p}(\mathcal{Z}|\text{u}_1)$, as in Fig. 2(a). We can see that the topic preferences of the users are skewed and with different distributions, which are consistent with the results in Fig. 2(a) obtained from the *MovieLens* dataset.



(a) Activity levels of users in a log-log coordinate



(b) Topic preferences of users, $\alpha = 0.36$ and $\beta = 0.6$.

Fig. 4. Activity level and topic preferences obtained from the requests generated by the synthetic method, $K = 100$, $F = 3000$.

VI. Simulation Results

In this section, we demonstrate the caching gain of exploiting user preference over content popularity. To provide ground truth of the request behavior for evaluating the learning performance, we use simulation results obtained from the synthetic data using the model in section V-B. To validate the gain from real data, we also consider the MovieLens dataset.

We consider a square area with side length 500 m, where $K = 100$ users are uniformly located. The collaboration distance $r_c = 30$ m. The file catalog size $F = 3000$, and each user can cache $M = 5$ files (i.e., 1.67 ‰ of all files).

[6] In recommendation problem, it has been shown that user preference varies over time due to the dynamic of file catalog and the user's exploration for new items [40]. However, the topic preference variation has never been analyzed in the literature.

[7] We also analyze a real video dataset of Youku in a university campus. The result shows that $\mathbb{E}_{k,m}[\text{sim}(\mathbf{q}_k, \mathbf{q}_m)] \approx 0.28$.

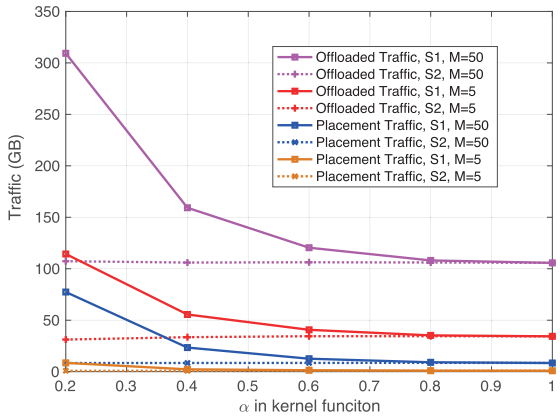
$\alpha = 0.36$ in the kernel function of the synthetic model, which corresponds to the average cosine similarity 0.4 of the MovieLens dataset. The parameter of Zipf distribution is $\beta = 0.6$, which is a typical value for a small area, and is slightly smaller than that is observed at the Web proxy [31]. We divide time into periods, each consisting of a peak time and an off-peak time. The cached files at each user are updated in off-peak time. This setup is used in the sequel unless otherwise specified.

A. Impact of Key Parameters and Interference Management Schemes

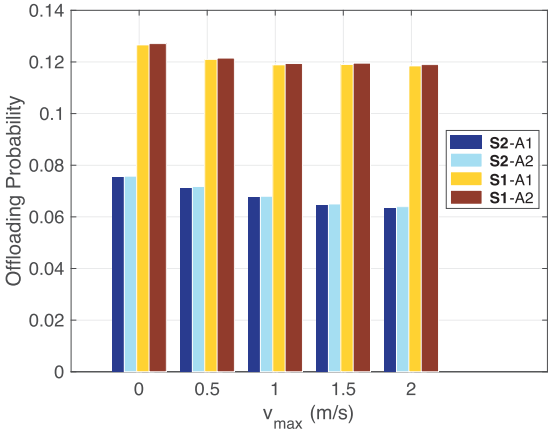
In this subsection, both user preference and content popularity are perfect. By using the caching policy exploiting individual user behavior in cache-enabled D2D communications, more wireless traffic in content delivery phase can be offloaded, but more wireless resources are also consumed in content placement phase, which may counteract the offloading gain. To address this concern, we first compare the offloaded traffic during content delivery and the traffic load generated by pre-caching the files to users during content placement with the proposed solution (i.e., **S1**) and the policy based on content popularity (i.e., **S2**, which represents existing caching policy). Then, we analyze the impact of user mobility, user preference similarity α, collaboration distance r_c, cache size M, and Zipf parameter β on the offloading probability. Considering that the gain from caching will reduce in interference environment [6], we also compare the performance of **S1** and **S2** when different interference management schemes are applied in content delivery phase.

We consider a widely used mobility model, *random walk model*, where a user moves from its current location to a new location with a randomly chosen direction and speed [42]. To compute the contact probability matrix, we consider that each user moves 100 seconds before changing direction and speed in two hours during peak time. The users are initially uniformly distributed, and the speed and direction of each user are uniformly chosen from [0, $v_{\max}$] m/s and [0, 2π], respectively. By computing the duration that the kth and the mth user can establish D2D links, $t^d_{k,m}$, in the period of $T_p = 2$ hours, we can obtain the contact probability $a_{k,m} = \frac{t^d_{k,m}}{T_p}$. By increasing $v_{\max}$, users may move with higher speed, and when $v_{max} = 0$, all users keep fixed.

In Fig. 5(a), we show the "offloaded traffic" (in GBytes, computed as the total number of requests served by D2D links multiplied by the file size), and the "placement traffic" load (in GBytes, computed as the overall amount of data transmitted by BS for placing the contents to the caches of all users, which depends on the total number of files to be placed and the file size). For content placement, both **S1** and the popularity-based caching policy (i.e., **S2**) can use unicast and multicast. It is worthy to note that the cached files according to **S2** are also different for users. Due to the coverage overlap of the helpers, adjacent users tend to cache different files even with **S2** in order to maximize the offloading probability. The content delivery phase is set as 10 hours, during which the request arrival rate in the area is 0.4 requests/s. Each file is with size of 30 MBytes (typical for YouTube videos). The results show



(a) Offloaded/generated traffic in content delivery/placement

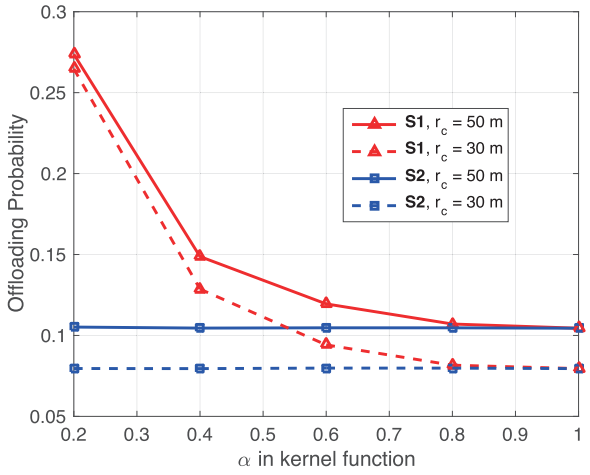


(b) Impact of user mobility, $\alpha = 0.36$

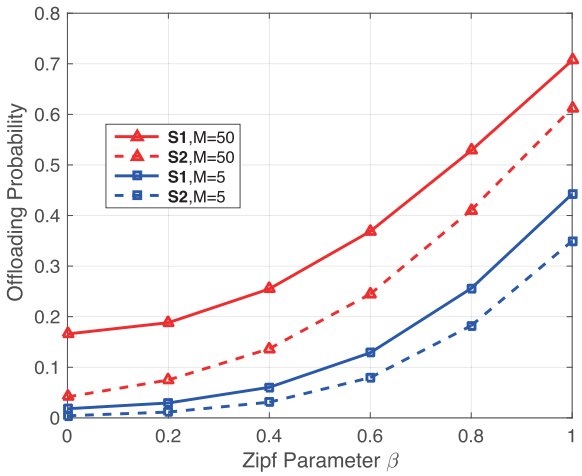
Fig. 5. Impact of the traffic in content placement and user mobility, **S1** is the caching policy with user preference.

that **S1** indeed generates more placement traffic than **S2** when user preferences are less similar (and also when the content popularity is less skewed, not shown due to space limitation) owing to the reduced multicast opportunity, and when each user caches more files. Nonetheless, **S1** generates much less traffic in content placement phase than the offloaded traffic in content delivery phase.

In Fig. 5(b), we show the impact of user mobility. $A1$ and $A2$ in the legend respectively represent the greedy algorithm and local optimal algorithm, which almost perform the same. The offloading probabilities decrease slightly with the growth of $v_{\max}$, as explained as follows. Owing to the mobility model, the average number of users that a user can establish D2D links with at any time does not change with $v_{\max}$. Then, the total effective cache size seen by the user does not change with $v_{\max}$. On the other hand, every user can contact with more users in the period with higher $v_{\max}$. Then, the caching policy needs to consider the preferences of more users, which reduces the cache hit ratio due to heterogeneous user preferences. Since the impact of mobility is not significant, we only consider $v_{max} = 0$ in the sequel. To obtain the results of $A2$ in Fig. 5, three iterations of step 1 (i.e., $t_{A2} = 3$) is necessary for convergence. According to

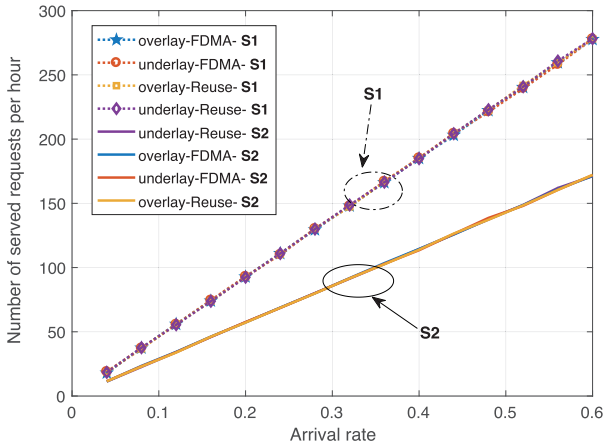


(a) Offloading probability vs. α, $\beta = 0.6$ and $M = 5$.

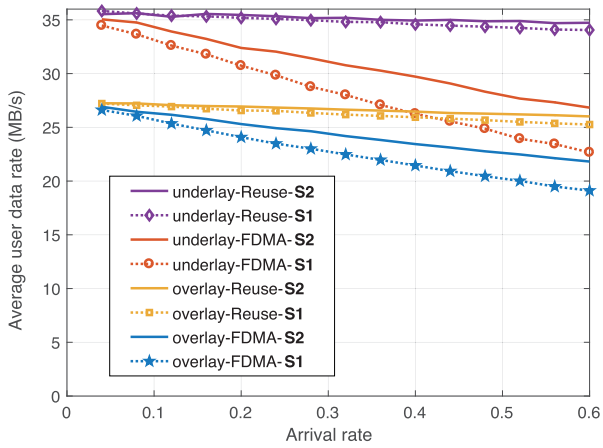


(b) Offloading probability vs. β, $\alpha = 0.36$ and $r_c = 30$ m.

Fig. 6. The impact of α, β, r_c and M, **S1** and **S2** are caching policies with user preference and content popularity.



(a) Offloaded number of requests in each hour



(b) Average rate of each user

Fig. 7. Impact of interference on system performance and user performance, $\alpha = 0.36$, $\beta = 0.6$, **S1** is the proposed solution.

analysis in Section III, the time complexity for $A1$ and $A2$ are respectively $O(K^2FM(K^2+1))$ and $O(3KF(K^2+M))$, and $A2$ will be $\frac{KM(K^2+1)}{3(K^2+M)} \approx 167$ times faster than $A1$ when $K = 100$ and $M = 5$. Since the proposed local optimal algorithm can achieve the same performance and is faster than the greedy algorithm, we only use $A2$ to obtain the caching policy in the following.

In Fig. 6(a), we show the impact of α and r_c. We can see that the offloading gain of **S1** over **S2** is high when α is small. This suggests that optimizing caching policy according to user preferences is critical when the user preferences are less correlated. As expected, when $\alpha \to 1$, the performance of the two policies coincide. The offloading gain is high for large collaboration distance, but the gain by using **S1** reduces as indicated in Remark 3. This is because with the growth of r_c, the number of users whose preferences a helper should consider for optimizing caching policy increases (when $r_c \to \infty$, the number of users equals to K).

In Fig. 6(b), we show the impact of β and M. As expected, with the growth of the value of M or β, the offloading probabilities increase for both **S1** and **S2**. This is because with a given value of α, the preference of every user becomes more skewed when β increases.

In Fig. 7, we show the impact of interference on the performance gain of the proposed solution. We consider both overlay and underlay modes [43], and in each mode the D2D links share the bandwidth either with full reuse or with frequency division multi-access (FDMA). The transmit power of each user is 23 dBm. The path loss model of D2D links is $37.6+36.8\log r$, where r is the distance between a transmitter and a receiver. In overlay mode, all active D2D links transmit over 20 MHz bandwidth with full reuse or share the 20 MHz bandwidth with FDMA, and the noise power is -100 dBm. In underlay mode, the D2D links share 40 MHz bandwidth among each other together with uplink transmission of cellular users [43] (also with full reuse or FDMA), and the noise power is -97 dBm. We consider five uniformly-located cellular users, which corresponds to the arrival rate of at least 0.5 requests/s if each user requests a file of 30 MB and the file is completely conveyed within 10s. To demonstrate the system performance, we evaluate the number of requests whose files can be completely conveyed in an hour. To demonstrate the

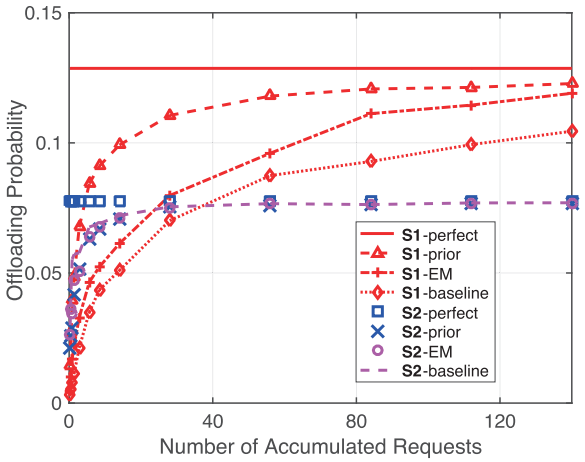
performance of each user, we evaluate the data rate per user averaged over large and small scale channels (also averaged over interference in underlay mode or in overlay mode with frequency reuse among D2D links). The simulation results show that the system performance of **S1** is superior to **S2** consistently in all scenarios, and the gain is large when the traffic load is heavy. When the D2D links reuse the bandwidth in both underlay and overlay modes, the user performance of **S1** is very close to **S2**. When using FDMA, **S1** becomes inferior. This is because each D2D link will be allocated less bandwidth if more D2D links transmit simultaneously.

B. Offloading Gain With Learned User Preference

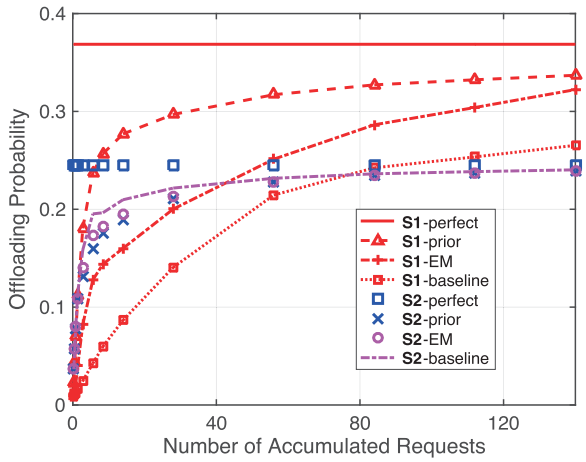
To show the gain of caching with user preference over content popularity, and the gain from learning with the pLSA model and the priori knowledge, we compare the following schemes:

1) "**S1-perfect**": The *proposed* caching policy with perfect user preference and activity level.
2) "**S2-perfect**": The *existing* caching policy optimized with perfect content popularity, which is the solution of problem **P2** (slightly different from the policies in [12] and [13]).
3) "**S1-EM**": The *proposed* caching policy with $\hat{\mathbf{w}}$ and $\hat{\mathbf{Q}}$ learned by the EM algorithm.
4) "**S2-EM**": The *existing* caching policy with learned local popularity of the K users, which is computed with (1) from the learned user preference by EM algorithm.
5) "**S1-prior**": The *proposed* caching policy with $\hat{\mathbf{Q}}$ learned by Algorithm 3.
6) "**S2-prior**": The *existing* caching policy with learned local popularity of the K users, which is computed from the learned user preference by Algorithm 3 as $\hat{p}_f = \sum_{\mathrm{u}_k \in \mathcal{U}} \hat{P}(\mathrm{u}_k, \mathrm{f}_f)$.
7) "**S1-baseline**": The *proposed* caching policy with learned user preference, which is obtained by the ML algorithm without pLSA model.
8) "**S2-baseline**": The *existing* caching policy with learned local popularity of the K users, which is obtained by using the traditional frequency-count popularity prediction method in [16]. Such a popularity learning method is the same as the method used in [17].

In Fig. 8, we show the offloading probability achieved by these schemes during the learning procedure with the synthetic data. To compare with the results to be obtained with realistic dataset in the sequel, we set the x-axis as the accumulated number of requests. It is shown that using pLSA and even the priori information do not help accelerate convergence of local popularity, because the simple frequency-count method already converges rapidly. Compared to the proposed caching policy with learned user preference (**S1**-EM, **S1**-prior and **S1**-baseline), **S2** with learned local popularity (**S2**-EM, **S2**-prior and **S2**-baseline) converge to **S2** with perfect content popularity (**S2**-perfect) more quickly. This is because the number of requests for each file from each user is much less than that from all the users in the area. Nonetheless, the proposed caching policy with learned user preference can



(a) $M = 5$.



(b) $M = 50$.

Fig. 8. Convergence performance on synthetic dataset, $\alpha = 0.36$, $r_c = 30$ m, $Z = 20$ for pLSA.

quickly achieve higher offloading probability than **S2** with learned (and even perfect) content popularity. The proposed caching policy with pLSA (both **S1**-EM and **S1**-prior) is superior to the baseline (**S1**-baseline), especially when the cache size at each user M is large. This is because some unpopular files will be cached with large M. For the unpopular files, the number of accumulated requests is less and user preference learning is more difficult. Besides, we can see that by exploiting prior knowledge of user activity level and topic preference, **S1**-prior converges much faster than **S1**-EM.

In Fig. 9, we show the offloading gain with the MovieLens dataset. Because **S2**-EM and **S2**-prior perform closely to **S2**-baseline, we only simulate **S2**-baseline here. We randomly choose 100 users from the dataset (which include both active and inactive users) and the most popular 3000 files. The timestamps of user requests are shuffled as in [18] to ensure the training set has the same user demand statistics as the test set. Compared with Fig. 8(a), we can see that the offloading probabilities for all methods on the realistic dataset are less than those with the synthetic data. This is because a user requests each movie at most once in the realistic dataset

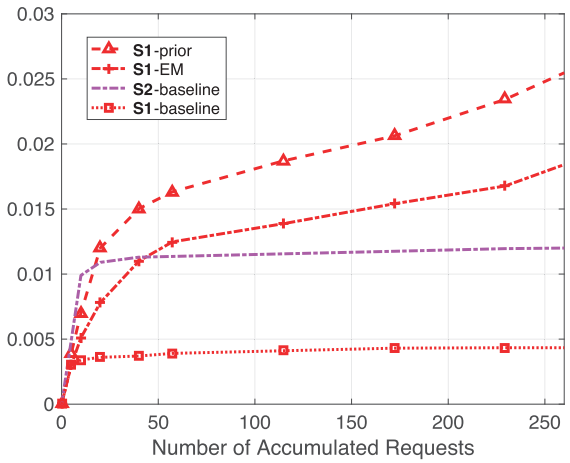


Fig. 9. Offloading probability on MovieLens dataset, $r_c = 30$ m, $M = 5$.

translated from rating data, while a user may request a file more than one time in the synthetic data. In practice, a user may request a file more than once, e.g., the file is a favorite song of the user or an educational video. Nonetheless, we can see that the proposed caching policy with predicted user preference still achieves much higher offloading gain than existing scheme. Because the prior knowledge of topic preferences is learned from realistic dataset rather than known as in Fig. 8(a), the performance gain of **S1**-prior over **S1**-EM is lower here, which however is still remarkable. Besides, we can see that **S1**-baseline always performs the worst, because user preference for unvisited files is hard to predict by using the ML algorithm without pLSA model when each user only requests a file at most once. However, we can still predict user preference for the unvisited files with both **S2**-EM and **S2**-prior, owing to the pLSA model.

VII. Conclusions

In this paper, we demonstrated the caching gain by exploiting learned individual user behavior in sending request. We first showed the connection between user preference and content popularity, and provided a probabilistic model to synthesize user preference from content popularity. We then formulated an optimization problem with given user preference and activity level to maximize the offloading probability for cache-enabled D2D communications, and proposed a low-complexity algorithm to solve the problem, which can achieve at least 1/2 optimality. Next, we modeled the user request behavior by pLSA, based on which the EM algorithm was used to learn the user preference and activity level. We analyzed statistics of user behavior in requesting contents and validated the synthetical model by a Movielens dataset. We find that: (i) the preferences for the most favorable files of each user can be modeled as Zipf distribution, but with different skewness and over different file sets, (ii) the user preferences are less similar, and (iii) the activity level and topic preference of each user change slowly over time, say in the time scale of year. Based on the 3rd observation from the real dataset, we introduced a prior knowledge based algorithm to exploit the activity level and topic preference previously learned, which shows the potential of transfer learning. Simulation results showed that using pLSA can quickly learn the individual user behavior, and the prior knowledge based algorithm converges even faster. Compared to existing caching policy using content popularity, the offloading performance can be remarkably improved by the caching policy using user preferences, both on the synthetic data with parameters fitted from real dataset and on the MovieLens dataset.

Appendix A
Proof of Proposition 1

The objective function of problem **P1′** can be further derived as

$$\begin{aligned} & f_{\text{off}}(\mathbf{c}_{k'}) \\ &= \sum_{k=1}^{K} w_k \sum_{f=1}^{F} q_{f|k} \\ &\quad \times \left(1 - \prod_{m \neq k'}^{K} (1 - a_{k,m} c_{m,f})(1 - a_{k,k'} c_{k',f})\right) \\ &= \underbrace{1 - \sum_{f=1}^{F} \sum_{k=1}^{K} w_k q_{f|k} \prod_{m \neq k'}^{K} (1 - a_{k,m} c_{m,f})}_{(a)} \\ &\quad + \sum_{f=1}^{F} c_{k',f} \underbrace{\left(\sum_{k=1}^{K} w_k q_{f|k} a_{k,k'} \prod_{m \neq k'}^{K} (1 - a_{k,m} c_{m,f}) \right)}_{(b)}, \end{aligned} \tag{A.1}$$

where both terms in (a) and (b) are not related to $c_{k',f}$. Then, solving the problem in (9) is equivalent to solving the following problem

$$\begin{aligned} \mathbf{P1'} \quad & \max_{c_{k',f}} \sum_{f=1}^{F} c_{k',f} \left(\sum_{k=1}^{K} w_k q_{f|k} a_{k,k'} \right. \\ & \qquad \left. \times \prod_{m=1, m \neq k'}^{K} (1 - a_{k,m} c_{m,f}) \right) \overset{(a)}{=} \sum_{f=1}^{F} c_{k',f} b_{k',f} \\ & s.t. \sum_{f=1}^{F} c_{k',f} \leq M, c_{k',f} \in \{0,1\}, \\ & \qquad 1 \leq f \leq F, \end{aligned} \tag{A.2}$$

where (a) is obtained by letting $b_{k',f} = \sum_{k=1}^{K} w_k q_{f|k} a_{k,k'} \prod_{m=1, m \neq k'}^{K} (1 - a_{k,m} c_{m,f})$. By finding file indices of the maximal M values of $b_{k',f} (1 \leq f \leq F)$ to constitute the set $\mathcal{I}_{k'}$, it is not hard to show that the optimal caching policy $c^*_{k',f}$ can be obtained as (11).

To obtain $c^*_{k',f}$, we need to compute $b_{k',f}$ with time complexity $O(K^2 F)$ and then choose the maximal M values of $b_{k',f}$ with complexity $O(FM)$. Finally, we can prove that the optimal solution of problem (9) can be obtained with complexity $O(K^2 F + FM) = O(F(K^2 + M))$.

APPENDIX B
PROOF OF PROPOSITION 2

Denote the caching policy at the k'th user after the $(t-1)$th iteration as $\mathbf{c}_{k'}^{(t-1)}$. In the tth iteration, the offloading probability before step 3 of Algorithm 2 is $f_{\text{off}}(\mathbf{c}_{k'}^{(t-1)})$ as in (A.1). After that step, $\mathbf{c}_{k'}^{(t)}$ is computed for the k'th user and the corresponding offloading probability is $f_{\text{off}}(\mathbf{c}_{k'}^{(t)})$. By subtracting $f_{\text{off}}(\mathbf{c}_{k'}^{(t-1)})$ from $f_{\text{off}}(\mathbf{c}_{k'}^{(t)})$, we can obtain

$$\begin{aligned}
&f_{\text{off}}(\mathbf{c}_{k'}^{(t)}) - f_{\text{off}}(\mathbf{c}_{k'}^{(t-1)}) \\
&= \sum_{k=1}^{K} w_k \sum_{f=1}^{F} q_{f|k} \Bigg(1 - \prod_{m=1, m\neq k'}^{K} (1 - a_{k,m}c_{m,f}) \\
&\quad \times (1 - a_{k,k'}c_{k',f}^{(t)})\Bigg) - \sum_{k=1}^{K} w_k \sum_{f=1}^{F} q_{f|k} \Bigg(1 \\
&\quad - \prod_{m=1, m\neq k'}^{K} (1 - a_{k,m}c_{m,f})(1 - a_{k,k'}c_{k',f}^{(t-1)})\Bigg) \\
&\overset{(a)}{=} \sum_{f=1}^{F} c_{k',f}^{(t)} b_{k',f} - \sum_{f=1}^{F} c_{k',f}^{(t-1)} b_{k',f} \\
&\overset{(b)}{=} (\max_{c_{k',f}} \sum_{f=1}^{F} c_{k',f} b_{k',f}) - \sum_{f=1}^{F} c_{k',f}^{(t-1)} b_{k',f} \geq 0,
\end{aligned}$$

where (a) is obtained by substituting (10) and (A.1), and (b) is obtained from (A.2). Thus, the offloading gain is monotonically improved until convergence.

To prove the optimality guarantee of the local optimal algorithm, we first convert the offloading probability into a function of a set instead of a matrix (i.e., $\mathbf{C}$). Denoting f_f^k as an action that caching the fth file at the kth user. Recall that $c_{k,f} = 1$ represents the kth user caching the fth file. Then, the caching policy for the kth user, $\mathbf{c}_k = [c_{k,1}, c_{k,2}, \ldots, c_{k,F}]$, can be re-expressed as a set $\mathcal{C}_k = \{\mathrm{f}_f^k | c_{k,f} = 1\}$, i.e., caching which files at the kth user. Let $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_K\}$, then problem **P1** is equivalent to the following problem,

$$\begin{aligned}
&\max_{\mathcal{C}} \ f(\mathcal{C}) \\
&\qquad = \sum_{k=1}^{K} w_k \sum_{f=1}^{F} q_{f|k} \left(1 - \prod_{\mathrm{f}_f^m \in \mathcal{C}} (1 - a_{k,m})\right) \\
&s.t. \ |\mathcal{C}_k| \leq M, 1 \leq k \leq K.
\end{aligned} \tag{B.1}$$

By defining a set $\mathcal{S} = \{\mathrm{f}_1^1, \mathrm{f}_2^1, \ldots, \mathrm{f}_F^1, \ldots, \mathrm{f}_1^K, \mathrm{f}_2^K, \ldots, \mathrm{f}_F^K\}$, we can see that $\mathcal{C} \subseteq \mathcal{S}$ and $f(\mathcal{C}) : 2^{\mathcal{S}} \rightarrow R$ is a discrete set function on subsets of $\mathcal{S}$. Let $\mathcal{A}, \mathcal{B} \subseteq \mathcal{S}$, $\mathcal{A} \subseteq \mathcal{B}$, and $\mathrm{f}_{k'}^{f'} \in \mathcal{S} \setminus \mathcal{B}$.

Denote the global optimal caching policy as $\mathcal{C}^* = \{\mathcal{C}_1^*, \mathcal{C}_2^*, \ldots, \mathcal{C}_K^*\}$, a local optimal caching policy obtained by Algorithm 2 as $\mathcal{C}^L = \{\mathcal{C}_1^L, \mathcal{C}_2^L, \ldots, \mathcal{C}_K^L\}$, and caching policy at users except the kth user as $\mathcal{C}_{\bar{k}}^L = \{\mathcal{C}_1^L, \mathcal{C}_2^L, ., \mathcal{C}_{k-1}^L, \mathcal{C}_{k+1}^L, .., \mathcal{C}_K^L\}$. Then, we can obtain

$$\begin{aligned}
f(\mathcal{C}^*) - f(\mathcal{C}^L) &\overset{(a)}{\leq} f(\mathcal{C}^* \cup \mathcal{C}^L) - f(\mathcal{C}^L) \\
&\triangleq f_{\mathcal{C}^L}(\mathcal{C}^*) \overset{(b)}{\leq} \sum_{k=1}^{K} f_{\mathcal{C}^L}(\mathcal{C}_k^*) \\
&\overset{(c)}{\leq} \sum_{k=1}^{K} f_{\mathcal{C}_{\bar{k}}^L}(\mathcal{C}_k^*) \overset{(d)}{\leq} \sum_{k=1}^{K} f_{\mathcal{C}_{\bar{k}}^L}(\mathcal{C}_k^L) \\
&\overset{(e)}{\leq} f(\mathcal{C}_1^L) + \sum_{k=2}^{K} f_{\cup_{i=1}^{k-1}\mathcal{C}_i^L}(\mathcal{C}_k^L) = f(\mathcal{C}^L),
\end{aligned} \tag{B.2}$$

where (a) is obtained because offloading probability is a monotone increasing function, i.e., $f(\mathcal{C}^*) \leq f(\mathcal{C}^* \cup \mathcal{C}^L)$, (b) is obtained by the property that for set $\mathcal{A}, \mathcal{B}, \mathcal{C} \subseteq \mathcal{S}$, we have $f_{\mathcal{A}}(\mathcal{B} \cup \mathcal{C}) \leq f_{\mathcal{A}}(\mathcal{B}) + f_{\mathcal{A}}(\mathcal{C})$, (c) and (e) are obtained by the property that for set $\mathcal{C} \subseteq \mathcal{A} \subseteq \mathcal{S}$ and $\mathcal{B} \subseteq \mathcal{S}$, $f_{\mathcal{A}}(\mathcal{B}) \leq f_{\mathcal{C}}(\mathcal{B})$, and (d) is obtained because for any caching policy at the kth user denoted as $\mathcal{C}_k^a$, we have $f_{\mathcal{C}_{\bar{k}}^L}(\mathcal{C}_k^L) - f_{\mathcal{C}_{\bar{k}}^L}(\mathcal{C}_k^a) = f(\mathcal{C}_k^L \cup \mathcal{C}_{\bar{k}}^L) - f(\mathcal{C}_k^a \cup \mathcal{C}_{\bar{k}}^L) \geq 0$ considering $\mathcal{C}^L$ is the local optimum of Algorithm 2. Thus, we have $f(\mathcal{C}^L) \geq \frac{1}{2} f(\mathcal{C}^*)$, and Proposition 2 follows.

REFERENCES

[1] B. Chen and C. Yang, "Caching policy optimization for D2D communications by learning user preference," in *Proc. 8th IEEE VTC*, Jun. 2017, pp. 1–6.

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

[3] K. Wang, Z. Chen, and H. Liu, "Push-based wireless converged networks for massive multimedia content delivery," *IEEE Trans. Wireless Commun.*, vol. 13, no. 5, pp. 2894–2905, May 2014.

[4] D. Liu, B. Chen, C. Yang, and A. F. Molisch, "Caching at the wireless edge: Design aspects, challenges, and future directions," *IEEE Commun. Mag.*, vol. 54, no. 9, pp. 22–28, Sep. 2016.

[5] N. Golrezaei, P. Mansourifard, A. F. Molisch, and A. G. Dimakis, "Base-station assisted device-to-device communications for high-throughput wireless video networks," *IEEE Trans. Wireless Commun.*, vol. 13, no. 7, pp. 3665–3676, Jul. 2014.

[6] D. Liu and C. Yang, "Energy efficiency of downlink networks with caching at base stations," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 4, pp. 907–922, Apr. 2016.

[7] M. Leconte, G. Paschos, L. Gkatzikis, M. Draief, S. Vassilaras, and S. Chouvardas, "Placing dynamic content in caches with small population," in *Proc. IEEE INFOCOM*, Apr. 2016, pp. 1–9.

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

[9] N. Golrezaei, K. Shanmugam, A. G. Dimakis, A. F. Molisch, and G. Caire, "FemtoCaching: Wireless video content delivery through distributed caching helpers," in *Proc. IEEE INFOCOM*, Mar. 2012, pp. 1107–1115.

[10] B. Blaszczyszyn and A. Giovanidis, "Optimal geographic caching in cellular networks," in *Proc. IEEE ICC*, Jun. 2015, pp. 3358–3363.

[11] X. Xu and M. Tao, "Modeling, analysis, and optimization of coded caching in small-cell networks," *IEEE Trans. Commun.*, vol. 65, no. 8, pp. 3415–3428, Aug. 2017.

[12] M. Ji, G. Caire, and A. F. Molisch, "Wireless device-to-device caching networks: Basic principles and system performance," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 1, pp. 176–189, Jan. 2016.

[13] B. Chen, C. Yang, and A. F. Molisch, "Cache-enabled device-to-device communications: Offloading gain and energy cost," *IEEE Trans. Wireless Commun.*, vol. 16, no. 7, pp. 4519–4536, Jul. 2017.

[14] Y. Guo, L. Duan, and R. Zhang, "Cooperative local caching under heterogeneous file preferences," *IEEE Trans. Commun.*, vol. 65, no. 1, pp. 444–457, Jan. 2017.

[15] A. Gharaibeh, A. Khreishah, B. Ji, and M. Ayyash, "A provably efficient online collaborative caching algorithm for multicell-coordinated systems," *IEEE Trans. Mobile Comput.*, vol. 15, no. 10, pp. 1863–1876, Aug. 2016.

[16] A. Tatar, M. D. de Amorim, S. Fdida, and P. Antoniadis, "A survey on predicting the popularity of Web content," *J. Internet Services Appl.*, vol. 5, no. 1, pp. 1–20, 2014.

[17] P. Blasco and D. Gündüz, "Learning-based optimization of cache content in a small cell base station," in *Proc. IEEE ICC*, Jun. 2014, pp. 1897–1903.
[18] E. Baştuğ *et al.*, "Big data meets telcos: A proactive caching perspective," *J. Commun. Netw.*, vol. 17, no. 6, pp. 549–557, Dec. 2015.
[19] M. D. Ekstrand, J. T. Riedl, and J. A. Konstan, "Collaborative filtering recommender systems," *Found. Trends Hum.-Comput. Interact.*, vol. 4, no. 2, pp. 81–173, Feb. 2010.
[20] D. Jannach, P. Resnick, A. Tuzhilin, and M. Zanker, "Recommender systems–beyond matrix completion," *Commun. ACM*, vol. 59, no. 11, pp. 94–102, 2016.
[21] U. Paul, A. P. Subramanian, M. M. Buddhikot, and S. R. Das, "Understanding traffic dynamics in cellular data networks," in *Proc. IEEE INFOCOM*, Apr. 2011, pp. 882–890.
[22] T. Hofmann, "Probabilistic latent semantic analysis," in *Proc. 15th Conf. Uncertainty Artif. Intell.*, 1999, pp. 289–296.
[23] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. Roy. Statist. Soc. B (Methodol.)*, vol. 39, no. 1, pp. 1–38, 1977.
[24] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *ACM Trans. Interact. Intell. Syst.*, vol. 5, no. 4, p. 19, Jan. 2016.
[25] X. Zhang, Y. Wang, R. Sun, and D. Wang, "Clustered device-to-device caching based on file preferences," in *Proc. 27th IEEE PIMRC*, Sep. 2016, pp. 1–6.
[26] Y. Wu, S. Yao, Y. Yang, Z. Hu, and C.-X. Wang, "Semigradient-based cooperative caching algorithm for mobile social networks," in *Proc. IEEE GLOBECOM*, Dec. 2016, pp. 1–6.
[27] E. Baştuğ, J.-L. Guénégo, and M. Debbah, "Proactive small cell networks," in *Proc. IEEE ICT*, May 2013, pp. 1–5.
[28] Y. Shi, M. Larson, and A. Hanjalic, "Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges," *ACM Comput. Surv.*, vol. 47, no. 1, p. 3, 2014.
[29] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, Mar. 2003.
[30] T. Hofmann, "Latent semantic models for collaborative filtering," *ACM Trans. Inf. Syst.*, vol. 22, no. 1, pp. 89–115, Jan. 2004.
[31] P. Gill, M. Arlitt, Z. Li, and A. Mahanti, "Youtube traffic characterization: A view from the edge," in *Proc. ACM SIGCOMM*, 2007, pp. 15–28.
[32] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, and S. Moon, "I tube, you tube, everybody tubes: Analyzing the world's largest user generated content video system," in *Proc. 7th ACM SIGCOMM*, 2007, pp. 1–14.
[33] C. Wang and D. M. Blei, "Collaborative topic modeling for recommending scientific articles," in *Proc. 17th ACM SIGKDD*, 2011, pp. 448–456.
[34] W.-J. Hsu, T. Spyropoulos, K. Psounis, and A. Helmy, "Modeling time-variant user mobility in wireless mobile networks," in *Proc. IEEE INFOCOM*, May 2007, pp. 758–766.
[35] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions—I," *Math. Program.*, vol. 14, no. 1, pp. 265–294, 1978.
[36] G. Calinescu, C. Chekuri, M. Pál, and J. Vondrák, "Maximizing a monotone submodular function subject to a matroid constraint," *SIAM J. Comput.*, vol. 40, no. 6, pp. 1740–1766, 2011.
[37] T. Hofmann, "Unsupervised learning by probabilistic latent semantic analysis," *Mach. Learn.*, vol. 42, no. 1, pp. 177–196, Jan. 2001.
[38] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.
[39] K. S. Trivedi, *Probability and Statistics With Reliability, Queueing, and Computer Science Applications*. Hoboken, NJ, USA: Wiley, 2002.
[40] D. Rafailidis and A. Nanopoulos, "Modeling users preference dynamics and side information in recommender systems," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 46, no. 6, pp. 782–792, Jun. 2016.
[41] F. Benevenuto, A. Pereira, T. Rodrigues, V. Almeida, J. Almeida, and M. Gonçalves, "Characterization and analysis of user profiles in online video sharing systems," *J. Inf. Data Manage.*, vol. 1, no. 2, p. 261, 2010.
[42] T. Camp, J. Boleng, and V. Davies, "A survey of mobility models for ad hoc network research," *Wireless Commun. Mobile Comput.*, vol. 2, no. 5, pp. 483–502, Sep. 2002.
[43] H. Min, J. Lee, S. Park, and D. Hong, "Capacity enhancement using an interference limited area for device-to-device uplink underlaying cellular networks," *IEEE Trans. Wireless Commun.*, vol. 10, no. 12, pp. 3995–4000, Dec. 2011.

Binqiang Chen (S'14) received the B.S. and Ph.D. degrees in electronics engineering from the School of Electronics and Information Engineering, Beihang University, in 2012 and 2018, respectively. His research interests include interference management, device-to-device communication, content-centric networks, and artificial intelligence in wireless communication. He has served as a Technical Program Committee Member for the IEEE VTC Spring 2017.

Chenyang Yang (M'99–SM'08) received the Ph.D. degree in electrical engineering from Beihang University in 1997. She has been a Full Professor with the School of Electronics and Information Engineering, Beihang University, since 1999. Her recent research interests include wireless edge caching, mobile artificial intelligence, and URLLC. She was supported by the 1st Teaching and Research Award Program for Outstanding Young Teachers of Higher Education Institutions by the Ministry of Education of China from 1999 to 2004. She was the Chair of the IEEE Communications Society Beijing Chapter from 2008 to 2012. She has served as a TPC member for numerous IEEE conferences and an Associate Editor or a Guest Editor for several IEEE journals.