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

Binqiang Chen and Chenyang Yang

Abstract

Prior works in designing caching policy do not distinguish content popularity with user preference. In this paper, we optimize caching policy for cache-enabled device-to-device (D2D) communications by exploiting individual user behavior in sending requests for contents. We first show the connection between content popularity and user preference. We then optimize the caching policy with the knowledge of user preference and active level to maximize the offloading probability, 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 expectation maximization algorithm. By analyzing a MovieLens dataset, we find that the user preferences are less similar. The dataset also shows that the active 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 indicates the potential of a hierarchical learning strategy in accelerating convergence rate. Simulation results show remarkable performance gain of the caching policy with learned user preference over existing policy with learned content popularity, which are obtained with the synthetic data validated by the MovieLens dataset.

Index Terms

User preference, Content popularity, Caching policy, D2D, Machine learning.

I. INTRODUCTION

Caching at the wireless edge has become a trend for content delivery [2-4], which can improve network throughput and energy efficiency as well as user experience dramatically [5, 6].

Owing to the limited storage size at each base station (BS) or user, optimizing caching policy in a proactive manner is critical to achieve the performance gain of caching at wireless edge, where
user demand statistics is exploited \cite{3, 7}. Assuming known content popularity, caching policy at BSs was optimized to minimize average download delay in \cite{8} and to maximize coverage probability in \cite{9}. Coded caching policy was optimized to maximize average fractional offloaded traffic and average ergodic rate for small-cell networks in \cite{10}. Caching policy was optimized to maximize offloading gain of cache-enabled D2D networks in \cite{11}. In these works, every user is assumed to request files according to \textit{content popularity}, i.e., the probability that each content in a catalog is requested by all users. Noticing that content popularity may not be identical among social groups, caching policies at the user groups with different group popularity were optimized in \cite{12} to minimize the average delay for cache-enabled D2D networks.

To implement proactive caching, content popularity needs to be predicted \cite{13}. Predicting content popularity has long been an active research field, targeting to diverse applications such as advertisement, where content popularity is defined as the accumulated number of requests each file received in the literature. Numerous methods have been proposed and can be classified into three categories \cite{14}: (i) Cumulative growth method, which predicts the growth rate by assuming that popularity increases with a constant rate. (ii) Time series analysis, which models the number of requests for a file at different moment as time series, such as autoregressive moving average model. (iii) Employing evolution trends, which is based on the observation that the evolution of popularity over time can be modeled by several temporal patterns. After classifying a file into a certain pattern and learning the pattern parameters, the popularity of the file can be predicted. By using these prediction methods, the content popularity defined with probability in \cite{2, 3, 5, 6, 8-12} can be obtained as a ratio of the number of requests for each file to the number of all requests \cite{15}. However, the number of users in a cell is much less than that in a region covered by a content server, and the requests of a mobile user may not only send to a single BS. As a result, the popularity learning at the BS is slower than the learning at the server due to the low request arrival rate in a cell \cite{7} \cite{15}. On the other hand, the local popularity observed at the BS may be very different from the global popularity observed at the server. Hence, using the popularity learned at the server for designing proactive caching policy for a BS may lead to low cache hit rate \cite{7}. By assuming that only the requests for already cached files can be observed, local popularity was predicted at a small BS by cumulative growth method as the number of observed requests for each file divided by the observation time in \cite{16}. Then, the popularity of each file is adjusted with a perturbation term, which is learned by applying multi-arm bandit (MAB) algorithm to trade off between exploitation (continue to
cache popular files) and exploration (caching the not-cached-files to predict their popularity more accurately). Finally, the predicted popularity with perturbation term was used for caching policy optimization. While interesting, the learning of this algorithm is slow. In [17], the content popularity was predicted with a real dataset measured from cellular network. By converting the number of requests received at each BS into rating, local content popularity at each cell was predicted by a widely-used collaborative filtering (CF) technique, singular value decomposition (SVD) [18], with which the files with largest ratings are cached at the BS.

In the literature of wireless caching, the following facts are largely overlooked: (i) as a demand statistic of large number of users, content popularity can not reflect the preference of each individual user, (ii) only a small potion of users are active in requesting contents. Most priori works [2, 3, 5–13, 15–17, 19] do not differentiate user preference and content popularity. The first fact has been noticed very recently. By assuming user preferences as Zipf distributions with different ranks, caching policy for D2D communications was optimized to minimize the average delay in [20] and to maximize the average cache hit ratio in [21]. However, both works do not validate the assumption for user preference, and assume that all users have same active level.

To facilitate proactive caching, user preference needs to be predicted. Predicting user preference is a key task for recommendation problem, which has been extensively investigated and the most commonly used technique is CF [18]. In the literature for designing recommendation systems, user preference is defined as the rating that a user gives for a file, such as 0 ~ 5 or simply “like” and “dislike”. CF methods can be mainly classified into two categories [22]: (i) Memory based method, which includes user-based and item-based approaches. User-based approach predicts a user’s rating on an item by the weighted average of the ratings that similar users have previously given to the item, and item-based approach is similar. (ii) Model based method, which is based on models with parameters estimated from historical records. Typical model based methods employs SVD, matrix factorization, and probabilistic latent semantic analysis (pLSA) as models [18]. Most CF methods predict the ratings for unrated contents of each user, which however cannot be directly used for wireless caching. To optimize caching policy in wireless edge, where various metrics are in statistical sense in terms of file requests [2, 3, 5, 6, 8–12, 15], user preference needs to be defined in probabilistic form. Nonetheless, there is no widely-accepted way to map the rating into probability, and there is even no clear definition for user preference. Fortunately, pLSA can be applied to model and predict user preference, which is originally developed for automatic indexing and information retrieval [23].
and then is employed in [24] to predict ratings as a probabilistic model in CF.

In this paper, we study optimal caching policy by learning user preference and active level, by taking cache-enabled D2D communications as an example. D2D links can be established for users in proximity, and the number of users that each user can share files with is rather limited. Consequently, optimizing caching policy by exploiting the preferences of adjacent users is more reasonable than that using content popularity, which reflects the collective behavior of large number of users. We start from the connection between content popularity and user preference, and define user preference and active level. We then formulate an optimization problem with given user preference and active level to maximize the offloading probability. Since the problem is NP-hard, a local optimal algorithm is proposed to reduce the complexity. In order to learn user preference and active level, we model user request behavior resorting to pLSA, and estimate the model parameters by maximizing likelihood function using expectation maximization (EM) algorithm [25]. By analyzing a MovieLens dataset, we find that the active level and topic preference of individual user change slowly over time. Based on this observation, we provide a prior knowledge based algorithm to predict preference. Simulation results show that the prior knowledge based algorithm can quickly learn user preference. Compared to simply regarding content popularity as user preference as in priori works, the offloading gain can be remarkably increased by the proposed caching policy with the learned user preference.

The major contributions of this paper are summarized as follows:

- We characterize the connection between content popularity and user preference, provide a method to synthesize user preference from content popularity, and validate the method by a real data set. We analyze the statistics of active level and topic preference of each user, and the user preference by a MovieLens dataset.

- We optimize the caching policy with user preference and active level, and predict the behavior of each individual user by estimating model parameters of pLSA. With the help of pLSA model to decompose the user behavior into different components and the observation that active level and topic preference change slowly over time, we introduce a prior knowledge based algorithm to learn user preference, which shows the potential of a hierarchical learning strategy in fast convergence. By using the synthesized user demands with parameters analyzed from the real dataset, simulation results show remarkable performance gain of the caching policy with user preference over that with local popularity, no matter the user demands are assumed known or learned from the historical requests.
The rest of the paper is organized as follows. Section II provides the relation between content popularity and user preference. Section III introduces the system model. Section IV optimizes the caching policy with known user preference and active level. Section V presents the pLSA model and learning algorithms. Section VI analyzes the statistics of user demands from a MovieLens dataset. Section VII provides simulation results. Section VIII concludes the paper.

II. CONTENT POPULARITY AND USER PREFERENCE

Consider a content library $\mathcal{F} = \{f_1, f_2, ..., f_F\}$ consisting of $F$ files that $K$ users in an area may request. Each file is with the same file size, but the analysis can be easily extended to the general case with different file sizes by dividing each file into chunks of equal size.

Content popularity is defined as the probability distribution that each file in the library is requested by all users in the area, denoted as $p = [p_1, p_2, ..., p_F]$, where $p_f = P(f_f)$ is the probability that the $f$th file is requested, $\sum_{f=1}^{F} p_f = 1$, $p_f \in [0, 1]$, $1 \leq f \leq F$. If the area consists of one or several cellular cells, then $p$ is local content popularity, which is usually different from the global content popularity observed at a video server. Content popularity can reflect the collective behavior of users in a community.

User preference is defined as the conditional probability distribution that a user requests a file given that the user sends a request, denoted as $q_k = [q_1|k, q_2|k, ..., q_F|k]$ for the $k$th user, where $q_{f|k} = P(f_f|u_k)$ is the probability that the $k$th user requests the $f$th file when the user sends a file request, $\sum_{f=1}^{F} q_{f|k} = 1$, $q_{f|k} \in [0, 1]$, $1 \leq f \leq F$ and $1 \leq k \leq K$. We use matrix $Q = (q_{f|k})^{K \times F}$ to denote the preferences of all users, where $(q_{f|k})^{K \times F}$ represents a matrix with $K$ rows and $F$ columns and $q_{f|k}$ as the element in the $k$th row and $f$th column. User preference can reflect the personal behavior of each individual user. Denote the probability that the $k$th user (denoted as $u_k$) sends a request as $w_k = P(u_k)$, which reflects how active the user is in sending file requests, $\sum_{k=1}^{K} w_k = 1$ and $w_k \in [0, 1]$. We use vector $w = [w_1, w_2, ..., w_K]$ to denote the active levels of the users. Then, $Q$ and $w$ can characterize the file request behavior of every individual user.

Computing content popularity from user preference: Content popularity for the area with $K$ users can be expressed as the weighted average of user preferences\footnote{In \cite{25}, the local popularity of each cell is obtained by taking a weighted average of the preferences of users associated with each BS, where the weight is the number of requests sent by each user.}, i.e.,

$$
\forall 1 \leq f \leq F:
$$

$$
p_f = \sum_{k=1}^{K} w_k q_{f|k}.
$$

\footnote{In \cite{25}, the local popularity of each cell is obtained by taking a weighted average of the preferences of users associated with each BS, where the weight is the number of requests sent by each user.}
Synthesizing user preference from content popularity: Content popularity can be modeled as a Zipf distribution according to many measured datasets \([17, 27, 28]\). The probability that the \(f\)th file (denoted as \(f\)) in the library is requested by the users in an area is

\[
p_f = \frac{f^{-\beta}}{\sum_{j=1}^{F} j^{-\beta}}, \quad 1 \leq f \leq F,
\]

where the files in the library are indexed in descending order of popularity, and the content popularity distribution is more skewed with larger value of \(\beta\) \([8]\).

User preference model obtained from measured dataset, unfortunately, is unavailable in the literature so far. To show the performance gain of caching with the knowledge of user preference over content popularity, we synthesize the user preference from the content popularity. Our method is inspired by the method in \([7]\) to synthesize local content popularity of a cell from that of a core network, and is with the detailed steps as follows:

1. \(u_k\) is associated with a feature value \(X_k\), which is a random variable uniformly selected from \([0, 1]\).
2. \(f_f\) is associated with a feature value \(Y_f\), which is again chosen uniformly from \([0, 1]\).
3. The correlation between the \(k\)th user and the \(f\)th file is controlled by a kernel function as \(g(X_k, Y_f) = (1 - |X_k - Y_f|)\frac{1}{\pi\alpha^2} \in [0, 1]\), where \(0 < \alpha \leq 1\). When \(g(X_k, Y_f) = 0\), the \(f\)th file will not be requested by the \(k\)th user. When \(g(X_k, Y_f) = 1\), the \(f\)th file is a preferred file of the \(k\)th user.

Then, the joint probability that the \(f\)th file is requested by the \(k\)th user is given by

\[
P(u_k, f_f) = w_k q_{f|k} = p_f \frac{g(X_k, Y_f)}{\sum_{k'=1}^{K} g(X_{k'}, Y_f)},
\]

from which we can obtain \(w_k = \sum_{f_f \in F} P(u_k, f_f)\) and \(q_{f|k} = P(u_k, f_f) / w_k\), respectively.

The main differences with \([7]\) lie in that: (i) we use (3) to compute the joint probability instead of the number of requests for the \(f\)th file received at a BS, and (ii) we introduce a parameter \(\alpha\) into the kernel function to capture the similarity of preferences among users rather than simply using \(g(X_k, Y_f) = (1 - 2|X_k - Y_f|)^4\) as in \([7]\).

To understand the role of parameter \(\alpha\), we consider cosine similarity frequently used in CF to reflect the similarity of preferences between two users \([18]\), which is defined as

\[
sim(q_k, q_m) = \frac{\sum_{f=1}^{F} q_{f|m} q_{f|k}}{\sqrt{\sum_{f=1}^{F} q_{f|m}^2} \sqrt{\sum_{f=1}^{F} q_{f|k}^2}}.
\]

\(2\)This model will be validated in Section \([7]\) by a real dataset.
To show the similarity among user preferences, we can define average cosine similarity as

\[
\mathbb{E}_{k,m}[\text{sim}(q_k, q_m)] = \frac{2}{K(K - 1)} \sum_{k,m} \frac{\sum_{f=1}^{F} q_f k q_f m}{\sqrt{\sum_{f=1}^{F} q_f^2 k \sum_{f=1}^{F} q_f^2 m}}.
\] (5)

**Remark 1:** When \( \alpha = 1 \), \( g(X_k, Y_f) = 1 \) for \( \forall k, f \). Then, all user preferences are identical and equal to the content popularity, and \( \mathbb{E}_{k,m}\{\text{sim}(q_k, q_m)\} = 1 \). 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 \( X_k = Y_f \). This indicates that \( g(X_k, Y_f)g(X_{k'}, Y_f) \to 0, k \neq k' \). Then, no user has the same preference, and \( \mathbb{E}_{k,m}\{\text{sim}(q_k, q_m)\} \to 0 \). We can see that \( \alpha \) in the kernel function can reflect the average cosine similarity among user preferences.

### III. System Model

Multiple BSs in the area are connected to core network via backhaul to serve the \( K \) uniformly distributed users, which constitutes a user set \( \mathcal{U} = \{u_1, u_2, ..., u_K\} \). Each single-antenna user has a local cache to store \( M \) files, and can act as a helper to share files via D2D link.

To provide high rate transmission with low energy cost, we consider a user-centric D2D communication protocol as in [29]. 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 \( r_c \). Each BS is aware of the cached files at users and coordinates the D2D communications.

Proactive caching consists of content delivery phase and content placement phase.

In content delivery phase, each user requests files according to its own preference. If a user can find its requested file in local cache, it directly retrieves the file with zero delay. If not, but the user can find the file in caches of other users with distance smaller than \( r_c \), the user establishes a D2D link with the closest user cached the file to fetch it. Otherwise, the user is accessed with the BS to fetch the file via backhaul. Because fetching locally can be regraded as fetching via D2D with extremely high data rate, both fetching locally and via D2D links are called fetching via D2D links in the sequel.

Denote the file requests matrix after a period as \( N = (n_{k,f})^{K \times F} \), where \( n_{k,f} \geq 0 \) is the cumulative number of requests from \( u_k \in \mathcal{U} \) to \( f_f \in \mathcal{F} \) in the period. Assume that a central processor (CP) at the core network can record the requests history of all users.

In content placement phase, the CP learns the user preferences \( Q \) and active levels \( w \) from the requests history \( N \), and then optimizes the caching policy for users and informs the cached
files of the users to the BSs. We consider deterministic caching policy\(^3\) denoted as a vector \(c_k = [c_{k,1}, c_{k,2}, \ldots, c_{k,F}]\) for the \(k\)th user, where \(c_{k,f} = 1\) if \(f\) 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 \(C = (c_{k,f})^{K \times F}\). After being informed about the files to be cached at the users in its cell, a BS fetches the files from the core network and refreshes the caches of the users during the off-peak time, where the energy cost at the BS for placing the files can be minimized by using the method in [30].

IV. CACHING POLICY OPTIMIZATION

In this section, we optimize the caching policy with known file request behavior of each individual user. For comparison, we also present the corresponding caching policy optimization problem with known content popularity. As an illustration for the caching gain brought by distinguishing user preference from content popularity, we use offloading gain as a performance metric. To reduce time complexity, we provide a local optimal algorithm.

A. Caching Policy Optimization with Individual and Collective Request Behavior

We use offloading probability to reflect the offloading gain introduced by cache-enabled D2D communications, defined as the probability that a user can fetch the requested file via D2D link.

When optimizing the 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 [31]. Consequently, it is reasonable to assume that the contact probability is known \textit{a priori} [21]. Let \(A = (a_{i,j})^{K \times K}\) represent the contact probability among users, where \(a_{i,j} \in [0, 1]\) is the probability that the distance between the \(i\)th user and the \(j\)th user is less than \(r_c\). When all users do not move, \(a_{i,j} \in \{0, 1\}\).

Denote \(p_{k,f}^d(A, C)\) as the probability that the \(k\)th user can fetch the \(f\)th file via D2D links given contact probability \(A\) and caching policy \(C\). The complementary probability of \(p_{k,f}^d(A, C)\) is the probability that the \(f\)th file is not cached at any users in proximity to the \(k\)th 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

\[
\begin{align*}
    p_{\text{off}}(Q, w, A, C) &= \sum_{k=1}^{K} w_k \sum_{f=1}^{F} q_{f|k} p_{k,f}^d(A, C) = \sum_{k=1}^{K} w_k \sum_{f=1}^{F} q_{f|k} \left(1 - \prod_{m=1}^{K} (1 - a_{k,m}c_{m,f})\right).
\end{align*}
\]

\(^3\)We do not consider probabilistic caching policy. Such policy is designed under the assumption that a group of nodes can share the same caching distribution [34][2], which is not appropriate for a system with heterogeneous user preferences.
Remark 2: Prior works assume known content popularity $p$, and implicitly assume that all users send requests with equal probability of $p$. Then, the offloading probability is

$$p_{off}^{pop}(p, A, C) = \frac{1}{K} \sum_{f=1}^{F} p_f \sum_{k=1}^{K} p_{k,f}^{d}(A, C).$$

Remark 3: When the collaboration distance $r_c \to \infty$, $a_{k,m} = 1$. Then, (6) becomes

$$p_{off}(Q, w, A, C) = \sum_{f=1}^{F} \left( 1 - \prod_{m=1}^{K} (1 - c_{m,f}) \right) \sum_{k=1}^{K} w_{kq_{f|k}} = \sum_{f=1}^{F} \left( 1 - \prod_{m=1}^{K} (1 - c_{m,f}) \right) p_f.$$  

In this extreme case, where D2D links can be established between any two users in the area, it is easy to show that $p_{off}(Q, w, A, C) = p_{off}^{pop}(p, A, C)$. This suggests that if (i) the area is one cell, (ii) the local popularity of the cell is known, and (iii) all users in the cell can establish D2D links, then the offloading probability achieved by optimizing caching policy with individual user behavior is identical to that with collective user behavior.

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

$$\textbf{P1} : \max_{c_{m,f}} \quad p_{off}(Q, w, A, C)$$

$$\text{s.t.} \quad \sum_{f=1}^{F} c_{m,f} \leq M, c_{m,f} \in \{0, 1\}, 1 \leq m \leq K, 1 \leq f \leq F. \quad (7)$$

With known content popularity, the counterpart caching policy can be optimized by solving the following problem,

$$\textbf{P2} : \max_{c_{m,f}} \quad p_{off}^{pop}(p, A, C)$$

$$\text{s.t.} \quad \sum_{f=1}^{F} c_{m,f} \leq M, c_{m,f} \in \{0, 1\}, 1 \leq m \leq K, 1 \leq f \leq F. \quad (8)$$

**Proposition 1:** Solving problems \textbf{P1} and \textbf{P2} are NP-hard.

**Proof:** See Appendix A.

Since problem \textbf{P1} is NP-hard, it is impossible to find its global optimal solution within polynomial time. In the sequel, we show that problem \textbf{P1} belongs to the same type of problem as in [8]. Thus, we can resort to greedy algorithm, which is commonly used to provide a solution achieving at least $\frac{1}{2}$ of the optimal value for such type of problem [32].

**Proposition 2:** \textbf{P1} is equivalent to maximizing a submodular function over matroid constraints.

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

$$v_C(m, f) = p_{\text{off}}(Q, w, A, C|_{c_{m,f}=1}) - p_{\text{off}}(Q, w, A, C)$$

$$= \sum_{k=1}^{K} w_k q_{f|k} \left( p_{d}^{d} (A, C|_{c_{m,f}=1}) - p_{d}^{d} (A, C) \right),$$

where (a) follows by substituting (6), $C$ is the caching matrix at previous step, and $C|_{c_{m,f}=1}$ is the matrix by letting $c_{m,f} = 1$ in $C$. The algorithm is summarized in Algorithm 1.

**Algorithm 1 Greedy Algorithm**

**Input:** $A; w; Q$;

- Initialize: Caching matrix $C = (0)^{K \times F}$; Files not cached at the $m$th user $\tilde{C}_m \leftarrow \{f_1, f_2, ..., f_F\}$;
- Users with residual storage space $U_0 \leftarrow \{u_1, u_2, ..., u_K\}$;

1: for $i = 1, 2, ..., K \times M$ do
2: \[$m^*, f^*\] = arg\ max_{m_n \in U_0, f_f \in \tilde{C}_m} v_C(m, f)$;
3: $C = C|_{c_{m^*,f^*}=1}$;
4: $\tilde{C}_{m^*} \leftarrow \tilde{C}_{m^*} \setminus f_{f^*}$;
5: if $|\tilde{C}_{m^*}| = F - M$ then
6: $U_0 \leftarrow U_0 \setminus u_{m^*}$;
7: end if
8: end for
9: $C^* = C$;

**Output:** Caching matrix $C^*$.

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 $C$ that introduces the highest incremental caching gain takes at most $KF$ iterations. For each time of computing $v_C(m, f)$ in (9), the time complexity is $O(K^2)$, and thus computing all $v_C(m, f)$ is $O(K^3 F)$. Hence the total time complexity for Algorithm 1 is $O(KM(KF + K^3 F)) = O(K^2 FM(K^2 + 1))$, which is high especially when the numbers of users $K$ and files $F$ are large.

**Remark 4:** The algorithm is also applicable for solving problem $P2$ by letting $q_{f|k} = p_f, \forall k, f$ in $Q$. Solutions for $P1$ and $P2$ obtained with Algorithm 1 are called $S1 - A1$ and $S2 - A1$.  

July 27, 2017 DRAFT
B. A Low Complexity Algorithm

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.

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

$$\mathbf{P1'}: \max_{c_{k'},f} f_{\text{off}}(c_{k'}) = \sum_{k=1}^{K} w_k \sum_{f=1}^{F} q_{f|k} \left(1 - \prod_{m=1, m \neq k'}^{K} (1 - a_{k,m}c_{m,f})(1 - a_{k,k'}c_{k',f})\right)$$

s.t. $\sum_{f=1}^{F} c_{k',f} \leq M, c_{k',f} \in \{0, 1\}, 1 \leq f \leq F$. (10)

**Proposition 3:** $\mathbf{P1'}$ can be solved with polynomial time complexity $O(F(K^2 + M))$.

*Proof:* See Appendix C. □

Based on the proof of Proposition 3, 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 $C$. In every iteration, by fixing $c_m (m \neq k', 1 \leq m \leq K)$, it respectively computes the offloading gain introduced by caching the $f$th file at the $k'$th user

$$b_{k',f} = \sum_{k=1}^{K} w_k q_{f|k} a_{k,k'} \prod_{m=1, m \neq k'}^{K} (1 - a_{k,m}c_{m,f}), 1 \leq f \leq F, 1 \leq k' \leq K.$$ (11)

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 $c^*_{k'}$ as

$$c^*_{k',f} = \begin{cases} 1, & f \in \mathcal{I}_{k'} \\ 0, & f \notin \mathcal{I}_{k'} \end{cases}. \quad (12)$$

The detailed algorithm is presented in Algorithm 2.

**Proposition 4:** Algorithm 2 monotonically increases the objective function $p_{\text{off}}(Q, w, A, C)$ of problem $\mathbf{P1}$ and finally converges.

*Proof:* See Appendix D. □

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

It is noteworthy that Algorithm 2 can also solve $\mathbf{P2}$ by letting $q_{f|k} = p_f, \forall k, f$ in Q. Solutions based on Algorithm 2 for $\mathbf{P1}$ and $\mathbf{P2}$ are respectively called $\mathbf{S1} - A2$ and $\mathbf{S2} - A2$. 

July 27, 2017 DRAFT
Algorithm 2 A Low Complexity Algorithm

Input: \( A; w; Q; \)

Initialize: Random caching \( c_m^{(0)} \) \((1 \leq m \leq K)\), \( 1 \rightarrow t \);

1: repeat
2: for \( k' = 1, 2, ..., K \) do
3: Based on \( c_m^{(t-1)} \) \((m \neq k')\), compute \( b_{k',f} \) by (11), constitute \( I_{k'} \) and obtain \( c_{k'}^* \) by (12).
4: \( c_m^{(t)} = c_{k'}^*; \)
5: end for
6: until We obtain the converged result \( (c_m^{(t)} = c_m^{(t-1)}, 1 \leq m \leq K) \)

Output: Caching matrix \( c_m^{(t)} \).

V. LEARNING USER PREFERENCE AND ACTIVE LEVEL

In this section, we predict user preference and active level. We first use pLSA to model content request behavior of an individual user, then learn the parameters in the model by maximizing likelihood function with EM algorithm. Finally, we present a prior knowledge based algorithm to learn user preference, which suggests a hierarchical learning strategy.

A. Modeling User Behavior in Requesting Content

PLSA was originally developed for classification in automatic indexing and information retrieve problem [23] and then applied to predict rating of a user [24], a key task in recommendation problem. To characterize the request behavior of a user, the model introduces a topic for each file request, where the topic maybe unobservable, but can be intuitively understood as comedy, adventure, etc.

By introducing latent topic set \( Z = \{z_1, z_2, ..., z_Z\} \) with cardinality \( |Z| = Z \), pLSA associates each topic \( z_j \in Z \) with each possible user request, i.e., \( u_k \in \mathcal{U} \) requests \( f_f \in \mathcal{F} \). Specifically, the request of each user can be modeled as the following steps with three model parameters:

- \( u_k \) sends a request with probability \( P(u_k) \) (i.e., active level),
- \( u_k \) chooses a topic \( z_j \) with probability \( P(z_j|u_k) \) (i.e., topic preference, \( \sum_{j=1}^{Z} P(z_j|u_k) = 1 \)),
- \( u_k \) requests \( f_f \) in topic \( z_j \) with probability \( P(f_f|z_j), \sum_{f=1}^{F} P(f_f|z_j) = 1 \), where conditional independence assumption is used. In particular, conditioned on \( u_k \) sending a request and choosing topic \( z_j \), \( u_k \) chooses \( f_f \) with probability \( P(f_f|z_j, u_k) = P(f_f|z_j) \), i.e., \( P(f_f|u_k) = \sum_{z_j \in Z} P(f_f|z_j)P(z_j|u_k) \).
Then, the joint probability that \( u_k \) requests \( f_f \) can be expressed as

\[
P(u_k, f_f) = P(u_k)P(f_f|u_k) = P(u_k) \sum_{z_j \in Z} P(f_f|z_j)P(z_j|u_k).
\]  

(13)

B. Learning User Behavior in Requesting Content

According to maximal likelihood (ML) principle, we can learn \( P(u_k) \), \( P(f_f|z_j) \) and \( P(z_j|u_k) \) with requests history \( n_{k,f} \) by maximizing the following log-likelihood function [33]

\[
\mathcal{L} = \sum_{i} \log P(u_{i_u}, f_{i_f}) = \sum_{u_k \in U} \sum_{f_f \in F} n_{k,f} \log P(u_k, f_f) = \sum_{u_k \in U} \sum_{f_f \in F} n_{k,f} \log P(u_k) \sum_{z_j \in Z} P(f_f|z_j)P(z_j|u_k),
\]

(14)

where the \( i \)th sample corresponding to the event that the \( i_u \)th user requests the \( i_f \)th file.

1) ML algorithm without pLSA model: By maximizing the log likelihood function in (a) of (14) without the pLSA model, it is not hard to obtain that

\[
\hat{P}(u_k, f_f) = \frac{n_{k,f}}{\sum_{k'=1}^{K} \sum_{f'=1}^{F} n_{k',f'}},
\]

(15)

which is nothing but a simple frequency-count prediction often used in literature [14], and can serve as a baseline for learning user preference.

Remark 5: If we directly estimate \( P(u_k, f_f) \) using (15), the number of parameters to estimate is \( KF \). By using the pLSA as in (b) of (14), the number of parameters is reduced from \( KF \) to \( K + KZ +ZF = Z(K + F) + K \), where \( K \) parameters are for learning active level, \( KZ \) parameters are for topic preference, and \( ZF \) parameters are for \( P(f_f|z_j) \). With less number of parameters to estimate, a learning algorithm can converge more quickly.

2) EM algorithm: EM algorithm is efficient for ML parameter estimation with latent variables [25]. To apply the algorithm, we rewrite the log-likelihood function in (b) of (14) as

\[
\mathcal{L} = \sum_{u_k \in U} n_k \log P(u_k) + \sum_{u_k \in U} \sum_{f_f \in F} n_{k,f} \log \sum_{z_j \in Z} P(f_f|z_j)P(z_j|u_k),
\]

(16)

where \( n_k = \sum_{f_f \in F} n_{k,f} \). It is not hard to see the terms in (a) and (b) can be independently maximized. The active level can be learned by maximizing term (a) in (16) as

\[
\hat{P}(u_k) = \frac{n_k}{\sum_{k'=1}^{K} \sum_{f'=1}^{F} n_{k',f'}},
\]

(17)

The other two model parameters \( P(f_f|z_j) \) and \( P(z_j|u_k) \) can be learned by maximizing term (b) in (16) using the EM algorithm as follows [33].
Starting from randomly generated initial values for the model parameters $P(z_j|u_k)$ and $P(f_f|z_j)$, $1 \leq j \leq Z$, $1 \leq f \leq F$ and $1 \leq k \leq K$, the EM algorithm alternates two steps: expectation (E) step and maximization (M) step. In the E-step, the posterior probabilities are computed for latent variable $z_j$ with current estimation of the parameters as

$$\hat{P}(z_j|u_k, f_f) = \frac{\hat{P}(z_j|u_k) \hat{P}(f_f|z_j)}{\sum_{z_j' \in Z} \hat{P}(z_j'|u_k) \hat{P}(f_f|z_j')}$$

which is the probability that $f_f$ requested by $u_k$ belongs to topic $z_j$. In the M-step, given $\hat{P}(z_j|u_k, f_f)$ computed by previous E-step, the parameters are updated as

$$\hat{P}(f_f|z_j) = \frac{\sum_{u_k \in U} n_{k,f} \hat{P}(z_j|u_k, f_f)}{\sum_{u_k \in U} \sum_{f_f' \in \mathcal{F}} n_{k,f'} \hat{P}(z_j|u_k, f_f')}$$

$$\hat{P}(z_j|u_k) = \frac{\sum_{f_f \in \mathcal{F}} n_{k,f} \hat{P}(z_j|u_k, f_f)}{n_k}$$

By alternating E-step (18) with M-step (19a) and (19b), the EM algorithm converges to a local maximum of log-likelihood function. Then, the active level of $u_k$ and the preference of $u_k$ for $f_f$ can be learned as $\hat{w}_k = \hat{P}(u_k)$ and $\hat{q}_{f|k} = \hat{P}(f_f|u_k) = \sum_{z_j \in Z} \hat{P}(f_f|z_j) \hat{P}(z_j|u_k)$, respectively.

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 active 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 active level and topic preference of a user learned previously (say at a video server that can observe much more requests from the user during a much longer time than at the CP), with the help of the topic information. While the active level $P(u_k)$ and topic preference $P(z_j|u_k)$ can never be learned perfectly, we assume that they are perfectly known in order to show the potential of the hierarchical learning strategy. Then, the user preference can be learned by only estimating $P(f_f|z_j)$, which can be obtained as in (19a),

$$\hat{P}(f_f|z_j) = \begin{cases} \frac{\sum_{u_k \in U} n_{k,f} \hat{P}(z_j|u_k, f_f)}{\sum_{u_k \in U} \sum_{f_f' \in \mathcal{F}} n_{k,f'} \hat{P}(z_j|u_k, f_f')}, & f_f \in \mathcal{F}_j, \\ 0, & f_f \notin \mathcal{F}_j \end{cases}$$

where $\mathcal{F}_j$ is the set of files associated with topic $z_j (1 \leq j \leq Z)$, 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 detailed algorithm is presented in Algorithm 3.
Algorithm 3 Learning user preference with prior knowledge.

Input: $N; Z; F_j, 1 \leq j \leq Z; \hat{P}(z_j|u_k)$; Stop condition $0 < \epsilon < 1$;

Initialize: $\hat{P}^{(0)}(f_j|z_j)$; Step $i \leftarrow 1$; Difference $\Delta \leftarrow \infty$; Log likelihood $L(0) \leftarrow 0$;

1: while $\Delta > \epsilon$ do
2: Using $\hat{P}(z_j|u_k)$ and $\hat{P}^{(i-1)}(f_j|z_j)$ to compute $\hat{P}^{(i)}(z_j|u_k, f_j)$ by (18);
3: Using $\hat{P}^{(i)}(z_j|u_k, f_j)$ and $F_j$ to compute $\hat{P}^{(i)}(f_j|z_j)$ by (20);
4: Compute log likelihood $L(i)$ with $\hat{P}(z_j|u_k)$ and $\hat{P}^{(i)}(f_j|z_j)$ using term (b) in (16);
5: $\Delta = |L(i) - L(i-1)|$; $i \leftarrow i + 1$;
6: end while
7: $\hat{q}_{f|k} \leftarrow \sum_{z_j \in Z} \hat{P}(f_j|z_j)\hat{P}(z_j|u_k)$ to compute $\hat{Q}$;

Output: $\hat{Q}$.

VI. USER REQUEST BEHAVIOR ANALYSIS WITH Movielens DATASET

The gain from caching highly depends on the user behavior in requesting contents, both collectively and individually. In this section, we first use a real dataset to analyze the connection between file catalog size and number of users in a region, as well as the active level, topic preference of each user and user preference, and validate the intuition in Section V-B3. Then, we validate the method provided in Section II to synthesize user preference from content popularity.

A. Statistical Results of User Demands

We use the MovieLens 1M Dataset [34] to analyze the statistics of user demands for movie topics, where MovieLens is a website that recommends movies for its users operated by GroupLens lab at the University of Minnesota. This dataset contains 1000209 ratings for 3952 movies provided by 6026 MovieLens users from the year of 2000 to 2003. Each sample of the dataset consists of user identity (ID), movie ID, rating and timestamp. A rating record in the dataset can be translated into a request record, because users typically give ratings to a movie only after watching it. 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 action, adventure, animation, children’s, comedy, crime, documentary, drama, fantasy, film-noir, horror, musical, mystery, romance, science fiction (sci-fi), thriller, war and western genre and are denoted as $z_1, z_2, \ldots, z_{18}$. For instance, Forrest Gump is associated with topics comedy ($z_5$), romance
(z_{14}) and war (z_{17}). With the topic information provided by MovieLens, we can see that if the $f$th movie is not associated with the $j$th topic, users who select $j$th topic will not choose to request the $f$th file, i.e., we can set $P(f_j|z_j) = 0$ 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. $\mathbf{N}_1$ can reflect user behavior on previously released file subset $\mathcal{F}_1$, and $\mathbf{N}_2$ can reflect user behavior on subsequently released file subset $\mathcal{F}_2$.

Specifically, we analyze the following statistical results:

- **File catalog size:** To reflect the randomness of the users who send requests in content delivery phase, it is the average number of files requested by a fixed number of randomly chosen users, which is obtained from $\mathbf{N} = [\mathbf{N}_1 \mathbf{N}_2]$ and the average is taken over users.

- **Active level:** $P_1(u_k)$ and $P_2(u_k)$ are learned using (17) with $\mathbf{N}_1$ and $\mathbf{N}_2$, respectively.

- **Topic preference:** Denote $\mathbf{p}_1(\mathcal{Z}|u_k) = [P_1(z_1|u_k), P_1(z_2|u_k), ..., P_1(z_{1976}|u_k)]$ and $\mathbf{p}_2(\mathcal{Z}|u_k) = [P_2(z_1|u_k), P_2(z_2|u_k), ..., P_2(z_{1976}|u_k)]$ as topic preference of the $k$th user estimated on subsets $\mathcal{F}_1$ and $\mathcal{F}_2$, respectively. $P_1(z_j|u_k)$, $P_2(z_j|u_k)$ are learned using (19b) by EM algorithm with $\mathbf{N}_1$ and $\mathbf{N}_2$, respectively. To reflect the temporal dynamic of topic preference for the $k$th user, we use the metric of cosine similarity in (4) to evaluate the similarity level as $\text{sim}(\mathbf{p}_1(\mathcal{Z}|u_k), \mathbf{p}_2(\mathcal{Z}|u_k))$.

- **User preference:** $q_{f|k} = \sum_{z_j \in \mathcal{Z}} P(f_j|z_j)P(z_j|u_k)$ 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.

In Fig. (1a), we show the relation between file catalog size and the number of users obtained from dataset (with legend “MovieLens”) and the corresponding fitted curves. The curve with legend “Log” fits the relation with the function $f(x) = \log(bx)$, and the curve with legend “Power” is with $f(x) = a x^b + c$. To evaluate the goodness of fit, we use the coefficient of determination (also called R-square) in linear regression, i.e., $R^2 = 1 - \frac{\sum_{i=1}^{S}(y_i - f(x_i))^2}{\sum_{i=1}^{S}(y_i - \bar{y})^2}$, where $S$ is the number of samples of data, $(x_i, y_i)$ is the $i$th sample, and $\bar{y} = \frac{\sum_{i=1}^{S} y_i}{S}$ [35]. $R^2 \leq 1$, and the large value of $R^2$ indicates 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 catalog size first increases quickly and then slowly, where “Power” function fits better than “Log” function. When the number of users is small (e.g., in a small cell), the file catalog size is small, which implies that the cache hit ratio
Fig. 1. File catalog size and user active level in log-log coordinates.

could be high with limited cache size. However, with limited number of requests due to a few associated users, the popularity is hard to predict rapidly at the small BS. When the number of users is large (e.g., in a macro cell with mobile users), the file catalog size increases slowly, and both fitted curves are close to the measured file catalog size. In [2], the authors suggest to use “Log” function to compute the file catalog size without validation of measured dataset. Here, the result shows that the “Log” function is reasonable when the number of users in a considered area for optimizing caching policy is large, say $K \geq 100$, at least for MovieLens dataset.

In Fig. 1(b), we show the active levels of users, where the user indices are ranked in descending order according to $P_1(u_k)$. We can see that the distribution of active levels is skewed, which indicates that majority requests are generated by a small number of users. Besides, the distribution of active levels from the two subsets of data are similar, where the cosine similarity is 0.87. This validates that the active level of a user changes slightly over time. We also show the corresponding fitted distributions, where “Weibull” is with function $f(x) = abx^{b-1}e^{-ax^{b}}$, “Exponential” is with function $f(x) = ae^{-bx}$, and “Zipf” is for Zipf distribution with fitting function $f(x) = ax^{-\beta}$ (Zipf only fits the most active 1000 users). We can see that the curve with measured data is linear on a log-log scale for active users, but the tail (after the 1000th user) decreases quickly. The truncate tail may come from the rating behavior of users for watched movies on MovieLens website. Some users rarely give ratings, and some users do not continuously give ratings after several initial ratings. As a result, the observed active levels of these users are very low. From
the values of $R^2$, we can find that “Weibull” is the best fitted distribution for user active level. Nonetheless, the distribution of the most active 1000 users is very close to Zipf distribution. This is fortunate, since these active users generate majority of traffic load and hence are of practical interests, and Zipf distribution is easy for analytical analysis.

In Fig. 2(a), we show the topic preferences of the 1st, 10th and 100th users obtained from $\mathcal{F}_1$, i.e., $p_1(Z|u_1)$, $p_1(Z|u_{10})$ and $p_1(Z|u_{100})$. The results obtained from $\mathcal{F}_2$ are similar and are not shown. The labels of x-axis are ranked in descending order according to $p_1(Z|u_1)$. The topic preferences of the 10th and 100th users with re-ranked x-axis according to $p_1(Z|u_{10})$ and $p_1(Z|u_{100})$ are also provided in the sub-figures. We can see that topic preference of each user is skewed, which indicates that each user has strong preferences towards specific topics. In fact, the topic preferences of all users in the dataset are skewed, which is not shown for consciousness. We can also see that 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.

In Fig. 2(b), we show the topic preference of the 1st user and the fitted distributions in a log-log coordinate. We can see that the best fitted distribution is a Zipf distribution with parameter $\beta = 1.05$. We have also fitted distributions of topic preferences for other users, but the curves are not provided. We observe that the best fitted distributions differ for users, where Zipf distribution is the best of 1425 users, Weibull distribution is the best for 1899 users, and Exponential distribution is the best for the remaining 2702 users (but the difference in $R^2$ from Weibull distribution for these users is negligible). For the users whose best fitted distributions are
Zipf distributions, the parameters of $\beta$ differ, which approximately follow a uniform distribution in $[0.5, 3]$. Yet for the most favorite several topics, Zipf distribution is always the best.

In Fig. 3(a), we show the change of topic preference over time of each user, where the users are ranked in the same way as in Fig. 1(b). We can see that the user with small index (i.e., the very active user) tends to have high cosine similarity between topic preferences, while the topic preferences of the less active users change significantly. This may come from the estimation errors, considering that the number of requests from a user with large index is much less.

In Fig. 3(b), 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% 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 degree, this result indicates that topic preferences can be approximated as invariant over time. This validates the intuition in Section V-B3.

In Fig. 4, we show user preference of the 1st user and the fitted distributions. The distribution of user preference for most favorite files (e.g., the top 100 favorite files) is close to Zipf distribution (a straight line in the log-log coordinate). The distribution for less popular files has a truncated

---

4In 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 [36]. However, the topic preference variation has not been analyzed in the literature.
Fig. 4. User preference of the 1st user, where the files are ranked in descending order.

tail. This is because a user almost does not request the files belonging to its unfavourable topics. We have also fitted the distributions for the preferences of other users, but the curves are not shown. We find that Weibull distribution is the best fitted distribution for all users, but the parameters differ and the skewness of curves are different. By fitting the user preferences for the top favorable 100 files of each user with Zipf distribution, we find that the parameters of $\beta$ differ in a range of $[0.2, 0.8]$, which is not consistent to the model that all user preferences are Zipf distributions with same parameter but with different ranks as assumed [20, 21]. Besides, we observe that the average cosine similarity of preferences among different users on dataset $N_1$ is $E_{k,m}[\text{sim}(q_k, q_m)] \approx 0.4$.

B. Validating Synthetic User Preference Model

In the following, we validate the user preference model by comparing the results obtained from data generated by the synthetic method in Section II and those from the MovieLens dataset.

In Fig. 5(a), we first show the impact of parameter $\alpha$ in the kernel function. The subfigure indicates that the synthetic user preference model can capture different levels of similarity among user preferences by adjusting $\alpha$, while the Zip parameter $\beta$ 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

$^5$We also analyze the real video dataset of Youku in a university campus. The result shows that $E_{k,m}[\text{sim}(q_k, q_m)] \approx 0.28$. 

July 27, 2017
Fig. 5. Active level and topic preferences obtained from the requests generated by the synthetic method, $K = 100$, $F = 3000$.

assumption that the users send their requests with equal probability, which is not true in reality, as shown in Fig. 5(a). From the figure we can observe that even when $\beta = 1$, $\alpha$ can be as small as 0.1. This is because few users are very active in sending file requests, who have large impact on content popularity according to (1). We can see that the distributions of user active level are skewed, which agree well with the results in Fig. 1(b) obtained from MovieLens dataset.

In Fig. 5(b), we show the topic preference distribution of the 1st, 10th and 100th users. The labels of x-axis are ranked according to $p(Z|u_i)$, as in Fig. 2(a). We can see that the topic preferences of the users are skewed, and the topic preference distributions of the three users are different, which are consistent with the results in Fig. 2(a) obtained from MovieLens dataset.

VII. SIMULATION RESULTS

In this section, we demonstrate the caching gain by exploiting user preference over that with content popularity, either perfect or predicted. The simulation results are obtained from the data generated from the synthetic user preference model, where the core parameters $F$ and $\alpha$ are set according to statistics obtained from the MovieLens dataset.

We do not use MovieLens data due to the following reasons: (1) The dataset is very sparse, where in average 0.15 requests is generated per user per day, which is much less than that suggested in [37]. This is because users do not often give ratings after watching a movie, and wireless traffic is generated by requests for contents from different providers. (2) The dataset can not provide the ground truth for user preference, hence can not provide an upper-bound for learning performance. (3) With the synthetic model, we can adjust parameters to reflect different levels of similarity among user preferences.
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 is willing to cache $M = 5$ files (i.e., 1.67 % of all files). $\alpha = 0.4$ in the kernel function, which corresponds to average cosine similarity 0.4 in Fig. 5(a) obtained from the MovieLens dataset. The parameter of Zipf distribution is $\beta = 0.6$, which is slightly smaller for a small area than that is observed at the Web proxy as reported in [27]. We divide time into two-hour periods, each consisting of a peak time and a off-peak time. The cached files at each user are updated in off-peak time. The request arrival rate of the users in the area is 0.04 requests per second, which reflects the high traffic load scenario for files with 30 MByte size (typical size of the YouTube videos) in [37]. This setup is used in the sequel unless otherwise specified.

### A. Impact of Key Parameters

In the sequel, we analyze the impact of user mobility, average user preference similarity $\alpha$, collaboration distance $r_c$, caching size $M$ and Zipf parameter $\beta$ on the offloading probability with perfectly known user preference and content popularity.

We consider a widely used mobility model, random walk model, where a user moves from its current location to a new location by randomly choosing a direction and speed to travel [38]. To compute the contact probability matrix, we consider a $T_p = 2$ hours period where each user moves 100 seconds in each period before changing direction and speed. The users are initially uniformly distributed, and the speed and direction of each user are uniformly chosen from $[0, v_{\text{max}}]$ m/s and $[0, 2\pi]$, respectively. By computing the duration that the $k$th and the $m$th user can establish D2D links, $t_{k,m}^d$, in each period of $T_p$ hours, we can obtain the contact probability $a_{k,m} = \frac{t_{k,m}^d}{T_p}$. By increasing $v_{\text{max}}$, users may move with higher speed, and when $v_{\text{max}} = 0$, all users keep fixed.

In Fig. 6, we show the impact of user mobility. $A1$ and $A2$ in the legend respectively represent the greedy algorithm (i.e., Algorithm 1) and local optimal algorithm (i.e., Algorithm 2), which achieve almost the same offloading probability. It is shown that the offloading probabilities decrease slightly with the growth of $v_{\text{max}}$, 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

---

Footnote: Using other values as the period does not affect the learning performance of user preferences. Yet the period can not be too short, since a frequent content placement brings extra traffic load.
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 a period of time (e.g., two hours) with higher $v_{\max}$. Then, the caching policy needs to be optimized by considering 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. 6, three iterations of step 1 (i.e., $t_{A2} = 3$) is necessary for convergence. According to analysis in Section IV, 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. 7(a), we show the impact of $\alpha$ and $r_c$. We can see that offloading gain of $S1$ over $S2$ can be remarkably improved when $\alpha$ 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. We can also see that the offloading gain can be improved by extending 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 to which a helper user can share cached files increases.

In Fig. 7(b), we show the impact of $\beta$ and $M$. As expected, with the growth of the number of files cached at each user $M$, the offloading probabilities increase for both $S1$ and $S2$. With
the increase of $\beta$, the offloading probability increases because more requests are generated for most popular files.

B. Offloading Gain with Learned User Preference

In what follows, we demonstrate the performance gain of the proposed caching policy by exploiting perfectly known and predicted user preferences over that with perfectly known and predicted content popularity. Specifically, we compare the following schemes:

1) “S1-perfect”: The proposed caching policy with perfect user preference and active level, which is the solution of problem $\text{P1}$.

2) “S2-perfect”: The existing caching policy optimized with perfect content popularity, which is the solution of problem $\text{P2}$.

3) “S1-EM”: The proposed caching policy with $\hat{w}$ and $\hat{Q}$ learned by the EM algorithm.

4) “S2-EM”: The existing caching policy with learned local content popularity, which is computed from user preference learned by EM algorithm with (1).

5) “S1-prior”: The proposed caching policy with $\hat{Q}$ learned by Algorithm 3.

6) “S2-prior”: The existing caching policy with learned local content popularity, which is computed from user preference learned by Algorithm 3 as $\hat{p}_f = \sum_{u_k \in \mathcal{U}} \hat{P}(u_k, f)$. 

7) “S1-baseline”: The proposed caching policy with learned user preference, which is obtained from (15) without pLSA model as $\hat{q}_{f|k} = \frac{n_{k,f}}{\sum_{f=1}^{F} n_{k,f}}, \hat{w}_k = \frac{\sum_{f=1}^{F} n_{k,f}}{\sum_{k'=1}^{K} \sum_{f=1}^{F} n_{k',f}}$. 

Fig. 7. The impact of $\alpha$, $\beta$, $r_c$ and $M$, S1 and S2 are caching policies with user preference and content popularity.
8) “S2-baseline”: The existing caching policy with learned local content popularity, which is obtained by using the traditional frequency-count popularity prediction method in [14].

In Fig. 8 we show the offloading probability achieved by these schemes during the learning procedure. It is shown that using pLSA and even the priori information do not help accelerate convergence of local content popularity, the simple frequency-count method converges rapidly (for the case with $M = 50$). Compared to the proposed caching policy with learned user preference (S1-EM, S1-prior and S1-baseline), we can see that S2 with learned local content 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 $K$ users in the area. Nonetheless, the proposed caching policy with learned user preference can 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 in the initial stage of the learning procedure and/or 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 active level and topic preference, S1-prior converges much faster than S1-EM. This suggests that the learning procedure can converge much faster by learning different information at different nodes, according to their time scale in variation.
Consequently, a hierarchical learning strategy deserves further investigation.

VIII. CONCLUSIONS

In this paper, we optimized the caching policy with learned individual user behavior in sending request for cache-enabled D2D communications. We first showed the connection between user preference and content popularity. We then formulated an optimization problem with given user preference and active level to maximize the offloading probability. Since the problem is NP-hard, a low-complexity algorithm was proposed to solve the problem. Next, we modeled the user request behavior by pLSA, based on which the EM algorithm was used to learn the user preference and active level. We used Movielens dataset to analyze several kind of user behavior in requesting contents, both collectively and individually. We find that (i) when the number of users $K$ in an area is large, the file catalog size is a logarithm function of $K$, (ii) the active level of the most active users can be modeled as Zipf distribution, (iii) the user preferences for the most favorable 100 files of each user can be modelled as Zipf distribution but with different skewness parameters, (iv) the user preferences is less similar, and (v) the active level and topic preference of each user change slowly over time, say in the time scale of year. Based on the 5th observation from analyzing the real dataset, we introduced a prior knowledge based algorithm to exploit the active level and topic preference previously learned, which shows the potential of a hierarchical learning strategy. 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 performance can be remarkably improved by the proposed caching policy when the user preferences are less correlated.

APPENDIX A

PROOF OF PROPOSITION

A special case of the objective function of problem P2 when contact probability $a_{i,j} \in \{0, 1\}$ (i.e., all user locations keep fixed) can be obtained from (1) and (8) as

$$P2': \max_{c_{m,f}} \sum_{k=1}^{K} \frac{1}{K} \sum_{f \in F_k} \rho_f$$

subject to

$$|C_m| \leq M, \quad F_k = \bigcup_{u_m \in U_k} C_m, \quad 1 \leq m \leq K, \quad 1 \leq f \leq F,$$

where $U_k \in \mathcal{U}$ is the user set that the $k$th user can establish D2D links with, $u_m \in U_k$ if $a_{k,m} = 1$ (i.e., $U_k = \{u_m | a_{k,m} = 1\}$), $F_k \in \mathcal{F}$ is the file set that the $k$th user can fetch via
D2D links (i.e., the union of the cached contents at the users in \(U_k\), which can be obtained as \(F_k = \bigcup_{u_m \in U_k} C_m = \{ f | \sum_{u_m \in U_k} c_{m,f} > 0 \} = \{ f | \sum_{m=1}^{K} a_{k,m} c_{m,f} > 0 \}\), and \(|C_m|\) is the cardinality of \(C_m\). It is easy to show that problem \(\textbf{P2}'\) has the same structure with the problem formulated in [8], which has been proved as NP-hard [8]. Because \(\textbf{P2}'\) is a special case of \(\textbf{P2}\), and \(\textbf{P2}\) has the same structure with \(\textbf{P1}\), both problems \(\textbf{P1}\) and \(\textbf{P2}\) are NP-hard.

\section*{Appendix B}

\textbf{Proof of Proposition 2}

To prove that the objective function of problem \(\textbf{P1}\) is a submodular function, we first convert it into a function of a set instead of a matrix (i.e., \(C\)).

Denoting \(f^k_f\) as an action that caching the \(f\)th file at the \(k\)th user. Recall that \(c_{k,f} = 1\) represents the \(k\)th user caching the \(f\)th file. Then, the caching policy for the \(k\)th user, \(c_k = [c_{k,1}, c_{k,2}, \ldots, c_{k,F}]\), can be re-expressed as a set \(C_k = \{ f^k_f | c_{k,f} = 1 \}\), i.e., caching which files at the \(k\)th user. Let \(C = \{ C_1, C_2, \ldots, C_K \}\), then problem \(\textbf{P1}\) is equivalent to the following problem,

\[
\max_{C} \quad f_{\text{off}}(C) = \sum_{k=1}^{K} \sum_{f=1}^{F} w_k q_{f,k} \left( 1 - \prod_{m \in C} (1 - a_{k,m}) \right)
\]

\[\text{s.t.} \quad |C_k| \leq M, 1 \leq k \leq K.\]

By defining a set \(S = \{ f^1_1, f^1_2, \ldots, f^K_1, f^K_2, \ldots, f^K_F \}\), we can see that \(C \subseteq S\) and \(f_{\text{off}}(C) : 2^S \to R\) is a discrete set function on subsets of \(S\).

Let \(A, B \subseteq S, A \subseteq B\), and \(f^a_k' \in S \setminus B\). Then, we have

\[
f_{\text{off}}(A \cup f^a_k') - f(A) - \left( f(B \cup f^a_k') - f(B) \right)
\]

\[
= \sum_{k=1}^{K} \sum_{f \in A} w_k q_{f,k} a_{k,k'} \prod_{m \in A} (1 - a_{k,m}) - \sum_{k=1}^{K} \sum_{f \in B} w_k q_{f,k} a_{k,k'} \prod_{m \in B} (1 - a_{k,m})
\]

\[
= \sum_{k=1}^{K} \sum_{f \in A} w_k q_{f,k} a_{k,k'} \left( \prod_{m \in A} (1 - a_{k,m}) - \prod_{m \in B} (1 - a_{k,m}) \right)
\]

\[
= \sum_{k=1}^{K} \sum_{f \in A} w_k q_{f,k} a_{k,k'} \prod_{m \in A} (1 - a_{k,m}) \left( 1 - \prod_{m \in B \setminus A} (1 - a_{k,m}) \right) \geq 0,
\]

where the inequality comes from the fact that \(a_{k,m} \in [0, 1]\).

Thus, the objective function \(f_{\text{off}}(C)\) is a submodular function.

The constraints of the problem in (B.1) can be shown as a matroid constraint as in [8]. Finally, Proposition 2 follows.
APPENDIX C

PROOF OF PROPOSITION 3

The objective function of problem $P1'$ can be further derived as

$$f_{\text{off}}(c_k') = \sum_{k=1}^{K} w_k \sum_{f=1}^{F} q_{f|k} \left( 1 - \prod_{m=1, m \neq k'}^{K} (1 - a_{k,m}c_{m,f})(1 - a_{k,k'}c_{k',f}) \right)$$

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

$$+ \sum_{f=1}^{F} C_{k',f} \left( \sum_{k=1}^{K} w_k q_{f|k}a_{k,k'} \prod_{m=1, m \neq k'}^{K} (1 - a_{k,m}c_{m,f}) \right), \quad (C.1)$$

where both terms in (a) and (b) are not related to $c_{k',f}$. Then, solving the problem in (10) is equivalent to solving the following problem

$$P1' \quad \max_{c_{k',f}} \quad \sum_{f=1}^{F} c_{k',f} \left( \sum_{k=1}^{K} w_k q_{f|k}a_{k,k'} \prod_{m=1, m \neq k'}^{K} (1 - a_{k,m}c_{m,f}) \right) \quad (a) \quad \sum_{f=1}^{F} c_{k',f}b_{k',f}$$

$$\text{s.t.} \quad \sum_{f=1}^{F} c_{k',f} \leq M, c_{k',f} \in \{0,1\}, 1 \leq f \leq F, \quad (C.2)$$

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 $I_{k'}$, it is not hard to show that the optimal caching policy $c^*_{k',f}$ can be obtained as (12).

To obtain $c^*_{k',f}$, we need to compute $b_{k',f}$ with time complexity $O(K^2F)$ 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 (10) can be obtained with complexity $O(K^2F + FM) = O(F(K^2 + M))$.

APPENDIX D

PROOF OF PROPOSITION 4

In step 3 of Algorithm 2, $c^{(t)}_{k'}$ is computed for the $k'$th user by fixing the caching policies at other users, and the corresponding offloading probability is $f_{\text{off}}(c^{(t)}_{k'})$ as in (C.1). It is not hard to show that $f_{\text{off}}(c^{(t)}_{k'}) \geq f_{\text{off}}(c^{(t-1)}_{k'})$ after each iteration of step 3 of Algorithm 2. Then, the algorithm will converge. Finally, Proposition 4 follows.
REFERENCES

[1] B. Chen and C. Yang, “Caching policy optimization for D2D communications by learning user preference,” in Proc. IEEE VTC Spring, 2017.

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

[3] 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, 2014.

[4] X. Wang, M. Chen, T. Taleb, A. Ksentini, and V. Leung, “Cache in the air: exploiting content caching and delivery techniques for 5G systems,” IEEE Commun. Mag., vol. 52, no. 2, pp. 131–139, 2014.

[5] N. Golrezaei, P. Mansourifard, A. Molisch, and A. 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, 2014.

[6] 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, 2016.

[7] M. Leconte, G. Paschos, L. Gkatziikis, M. Draief, S. Vassilaras, and S. Chouvardas, “Placing dynamic content in caches with small population,” in Proc. IEEE INFOCOM, 2016.

[8] 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, 2012.

[9] B. Blaszczyszyn and A. Giovanidis, “Optimal geographic caching in cellular networks,” in Proc. IEEE ICC, 2015.

[10] X. Xu and M. Tao, “Modeling, analysis, and optimization of coded caching in small-cell networks,” IEEE Trans. Commun., early access.

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

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

[13] A. Gharaiheb, A. Khreishah, B. Ji, and M. Ayyash, “A provably efficient online collaborative caching algorithm for multicell-coordinated systems,” IEEE Trans. on Mobile Comput., vol. 15, no. 8, pp. 1863–1876, 2016.

[14] A. Tatar, M. D. de Amorim, S. Fdida, and P. Antoniadis, “A survey on predicting the popularity of web content,” Springer J. Internet Services and Appl., vol. 5, no. 1, pp. 1–20, 2014.

[15] B. Bharath, K. Nagaranda, and H. V. Poor, “A learning-based approach to caching in heterogenous small cell networks,” IEEE Trans. Commun., vol. 64, no. 4, pp. 1674–1686, 2016.

[16] P. Blasco and D. Gunduz, “Learning-based optimization of cache content in a small cell base station,” in Proc. IEEE ICC, 2014.

[17] E. Baştuğ, M. Bennis, E. Zeydan, M. A. Kader, I. A. Karatepe, A. S. Er, and M. Debbah, “Big data meets telcos: A proactive caching perspective,” IEEE J. Commun. Netw., vol. 17, no. 6, pp. 549–557, 2015.

[18] M. D. Ekstrand, J. T. Riedl, and J. A. Konstan, “Collaborative filtering recommender systems,” Foundations and Trends in Human-Computer Interaction, vol. 4, no. 2, pp. 81–173, 2011.

[19] J. Song, M. Sheng, T. Q. S. Quek, C. Xu, and X. Wang, “Learning based content caching and sharing for wireless networks,” IEEE Trans. Commun., early access.

[20] X. Zhang, Y. Wang, R. Sun, and D. Wang, “Clustered device-to-device caching based on file preferences,” in Proc. IEEE PIMRC, 2016.

[21] 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, 2016.

[22] 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 Computing Surveys, vol. 47, no. 1, p. 3, 2014.

[23] T. Hofmann, “Probabilistic latent semantic analysis,” in Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence, 1999.

[24] ———, “Latent semantic models for collaborative filtering,” ACM Trans. Inf. Syst., vol. 22, no. 1, pp. 89–115, 2004.

[25] A. P. Dempster, N. M. Laird, and D. B. Rubin, “Maximum likelihood from incomplete data via the EM algorithm,” J. Royal Statist. Soc. B, 1977.

[26] E. Bastug, J.-L. Guéneégo, and M. Debbah, “Proactive small cell networks,” in Proc. IEEE ICT, 2013.

[27] P. Gill, M. Arlitt, Z. Li, and A. Mahanti, “Youtube traffic characterization: a view from the edge,” in Proc. ACM SIGCOMM, 2007.

[28] 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. ACM SIGCOMM, 2007.

[29] B. Chen, C. Yang, and A. F. Molisch, “Cache-enabled device-to-device communications: Offloading gain and energy cost,” IEEE Trans. Wireless Commun., early access.

[30] C. Yao, B. Chen, C. Yang, and G. Wang, “Energy-saving pushing based on personal interest and context information,” in Proc. IEEE VTC Spring, 2016.

[31] W. j. Hsu, T. Spyropoulos, K. Psounis, and A. Helmy, “Modeling time-variant user mobility in wireless mobile networks,” in Proc. IEEE INFOCOM, 2007.

[32] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, “An analysis of approximations for maximizing submodular set functions,” Mathematical Programming, vol. 14, no. 1, pp. 265–294, 1978.

[33] T. Hofmann, “Unsupervised learning by probabilistic latent semantic analysis,” Machine learning, vol. 42, no. 1, pp. 177–196, 2001.

[34] F. M. Harper and J. A. Konstan, “The movielens datasets: History and context,” ACM Transactions on Interactive Intelligent Systems (TaIIS), vol. 5, no. 4, p. 19, 2016.

[35] K. S. Trivedi, Probability and Statistics With Reliability, Queuing and Computer Science Applications. John Wiley and Sons Ltd., 2002.

[36] 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, 2016.

[37] 3GPP, “TR 36.814 technical specification group radio access network,” 2014.

[38] T. Camp, J. Boleng, and V. Davies, “A survey of mobility models for ad hoc network research,” Wiley Online Library Wireless commun. and mobile comput., vol. 2, no. 5, pp. 483–502, 2002.