Jointly Optimizing Content Caching and Recommendations in Small Cell Networks

Livia Elena Chatzieleftheriou, Merkouris Karaliopoulos, and Iordanis Koutsopoulos

Abstract—Caching decisions typically seek to cache content that satisfies the maximum possible demand aggregated over all users. Recommendation systems, on the contrary, focus on individual users and recommend to them appealing content in order to elicit further content consumption. In our paper, we explore how these, phenomenally conflicting, objectives can be jointly addressed. First, we formulate an optimization problem for the joint caching and recommendation decisions, aiming to maximize the cache hit ratio under minimal controllable distortion of the inherent user content preferences by the issued recommendations. Then, we prove that the problem is NP-complete and that its objective function lacks those monotonicity and submodularity properties that would guarantee its approximability. Hence, we proceed to introduce a simpler heuristic algorithm that essentially serves as a form of lightweight control over recommendations so that they are both appealing to end-users and friendly to network resources. Finally, we draw on both analysis and simulations with real and synthetic datasets to evaluate the performance of the algorithm. We point out its fundamental properties, provide bounds for the achieved cache hit ratio, and study its sensitivity to its own as well as system-level parameters.

Index Terms—content caching, recommender systems, small cells, algorithmic design, wireless networks

1 Introduction

Content caching has been experiencing revived interest in recent years within the context of current and next generation wireless cellular networks. The soaring demand for mobile video services pushes caching functionality towards the wireless network edge [2]. Storing popular content at caches close to the user results in enhanced Quality of Experience (QoE) for the end users and smaller footprints of the bandwidth-demanding mobile video traffic within the wireless network.

On the other hand, recommender systems have become integral components of content provision sites. Their mission is to make personalized recommendations for movies, video clips, music songs or other content items that best match the interests and preferences of individual users. This, in turn, improves the users' satisfaction and boosts their engagement in the sense that it increases the number of content downloads. For instance, the recommender system used by Netflix is considered responsible for about 80 percent of the hours streamed at Netflix [3], whereas the Related Video recommendations generate about 30 percent of the overall views on YouTube [4].

Typically, the recommendation engine and the caches at the wireless network are owned and managed by different entities. Recommender systems are controlled by content providers through apps that interact with users, whereas the caching infrastructure is typically possessed and controlled by the wireless network operator. Content providers often insert, through their own or third-party Content Delivery Networks (CDNs), servers storing content within other networks. In the case of cellular networks, these servers tend to be placed at their egress nodes rather than at their edge.

However, a persistent trend is that players with originally distinct roles in the business value chain, such as access network operators and content providers, tend to deploy their own content delivery solutions, albeit for different reasons. Content providers seek to acquire better control of the network access infrastructure so as to improve the Quality of Experience delivered to their subscribers. Netflix Open Connect[1] and Google Global Cache[2] are two widely known examples of CDN solutions owned by content providers. Access network operators, on the other hand, primarily seek to minimize costs related to the delivery of video traffic through external networks. At the same time, by having storage servers closer to the end- users, telco CDNs can provide their subscribers with faster content access.

In the case of wireless networks, these trends motivate novel content provisioning scenarios, whereby the coordination of (at least) three different mechanisms is deemed feasible towards optimizing user- and network-centric performance measures. First, *content replication* at the caches of different (small) cells can be used to maximize the locally-served demand and optimize the access delays experienced by users, hence their QoE. At the same time, caching reduces the traffic at the backhaul links and maximizes the cache hit ratio. Second, dynamically *routing user content*

• *The author are with the Department of Informatics, Athens University of Economics and Business, Athens 104 34, Greece.*
E-mail: {liviachatzi, mkaralio, jordan}@aueb.gr.

Manuscript received 28 May 2017; revised 1 Mar. 2018; accepted 12 Apr. 2018. Date of publication 30 Apr. 2018; date of current version 3 Dec. 2018.
(Corresponding author: Livia Elena Chatzieleftheriou.)
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TMC.2018.2831690

1. openconnect.netflix.com/en/
2. peering.google.com/

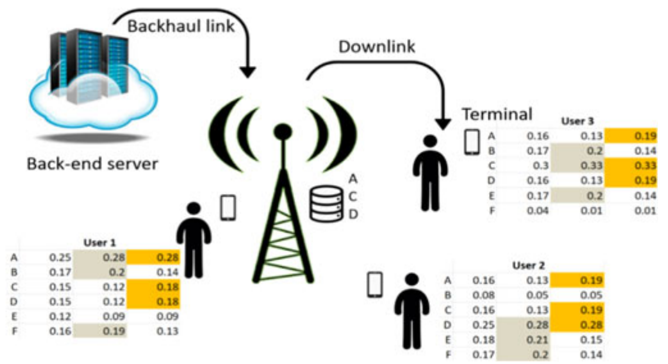


Fig. 1. *Illustration of the system model and toy example*. Caches are co-located with small cells and serve users with content of their preference. The example shows three users with their content preference distributions (2nd column) over six items A-F (1st column). The three items (A, C, and D) that attract the highest aggregate preference from all users are locally cached, yielding an expected cache hit ratio of 0.58. Consider a recommender system issuing content recommendations to users and a naive model for their impact: When an item is recommended to a user the preference (demand) for that item is boosted by 0.03. On contrary, when an item is not recommended to a user, the preference of that user for this item decreases by 0.03. When recommendations are issued for the top-3 items in users' preferences (3rd column), the cache hit ratio drops to 0.55. When recommendations are issued for items that are both cached and of adequate interest to users, yet not necessarily within the top-3 set (4th column), the cache hit ratio increases to 0.67.

requests to different cell caches, it can balance the request load across caches, again improving user QoE but also the network resource usage. Finally, *recommender systems* can be carefully used to *nudge* users towards more network-friendly content request patterns, i.e., they can *shape* content demand for the benefit of the caching mechanism.

Motivation. This last mechanism appears to have interesting repercussions for the design of caching algorithms. It is, in fact, the *coupling* between these two mechanisms, content caching and recommender systems, that serves as the main motivation for this work. Consider, for example, a cell cache serving the users who are associated with the cell. The rough idea is that the recommender system does not necessarily issue recommendations for content that ranks as the top most relevant according to the recommendation algorithm; instead, it could recommend to individual users cached content that still matches *adequately* with their preferences and, at the same time, attracts strong demand from many other users. Hence, anticipating that its recommendations affect the content access patterns of users, *the recommender system seeks to gently blunt some of the heterogeneity in users' demands, thus aiming at higher caching efficiency and better users' QoE* (Fig. 1).

The possibility to dynamically route content requests through different (small) cells within reach of the user adds further degrees of freedom to the problem. Besides *which* content to store, the caching decisions concern *where* to store it. Hence, a joint caching and recommendation algorithm could cache and recommend to a user content items that rank second, third, or lower in his/her preferences, as long as these items are in great demand in at least one of the cells that lie within the coverage of the user. It would instead avoid caching and recommending an item that, say, ranks first in the user interests but is not popular among other users in any of the cells the user can associate with.

Our Contributions. The main novelty of our work is that we take a different viewpoint to recommender systems. We approach them as an additional traffic engineering mechanism that can also help improve performance measures on the wireless network side. As a result:

- We define a system model in Section 2 that captures the coupling between caching decisions and issued recommendations. Our viewpoint to recommender systems raises some concerns, not least ethical ones. Hence, we introduce a measure called *preference distortion tolerance*, to quantify how much the engineered recommendations distort the original user content preferences.
- We formulate an optimization problem for the joint task of caching and recommending content to each user in Section 3. The objective of the Joint Caching and Recommendations Problem (JCRP) is to maximize the cache hit ratio under controlled preference distortion tolerance. We prove that the JCRP is NP-complete and that its objective function lacks those monotonicity and submodularity properties that would guarantee its approximability.
- We devise a low-complexity practical heuristic algorithm that solves efficiently the JCRP in Section 4. The algorithm is essentially a form of lightweight control over user recommendations, so that the recommended content is both appealing to the end-user and more friendly to the caching system and the network resources.
- Finally, in Sections 5 and 6, we thoroughly evaluate the proposed algorithm. The evaluation is carried out both analytically and through simulations with real and synthetic datasets and establishes main properties and performance bounds of the algorithm. A number of propositions are stated here and some of them are discussed briefly. The reader may refer to [5] for more propositions and their detailed proofs.

2 System Model

2.1 Caches, Content, and Users

Our model involves a set of caches $\mathcal{C}$, a catalogue of content items (e.g., video clips), $\mathcal{I}$, and a set of users, $\mathcal{U}$ (Fig. 1).

2.1.1 Caches

Caches are co-located with wireless network microcells that typically have a range in the order of a few hundred meters. Each cache $z \in \mathcal{C}$ has limited storage, C_z, which is measured in normalized file size units. At any point in time, it stores a finite set of files, referred to as the cache placement $\mathcal{P}_z$. An additional cache is installed on a backend server, e.g., in the cloud. This "cache" is assumed to have enough capacity to store copies of the entire catalogue.

2.1.2 Content

Content items are relevant to one or more of a set of M thematic categories. The detail and resolution of this categorical separation may vary. Such information may be stored in content metadata in the form of (hierarchical) tags. (e.g., "soccer" might be a distinct category, but it may also be further split into English/French/Spanish soccer). Such

TABLE 1
Notation Table

Notation	Context
C_z	Storage capacity of cache z
M	Number of thematic categories
$\mathbf{L}_i$	Normalized length of item i
$\mathbf{f}^i(j)$	Feature vector value of content item i in feature j
$\mathbf{f}_u(j)$	Feature vector value of user u in feature j
p_u^{pref}	Inherent content preference distribution of user u
a_{ui}	Similarity of user u and item i
p_u^{rec}	Probability distribution due to recommendation
p_u^{req}	Content item request probability distribution of user u (recommended items)
$\tilde{p}_u^{req}$	Content item request probability distribution of user u (non-recommended items)
R	Number of recommended items
$\mathcal{W}_u$	Recommendation window of user u
K_u	Length of $\mathcal{W}_u$
r_d	Preference distortion tolerance
H_s	Cache hit ratio under scheme s
$\mathcal{RC}_u^{in}$	Provisional set of recommended items to u
$\mathcal{P}$	Cache placement
$\mathcal{RC}_u^{f}$	Final set of recommended items to u
ZD	Zero-distortion scheme
UD	Unbounded-distortion scheme
$CawR$	Caching-aware recommendations scheme

information may be stored in content metadata in the form of (hierarchical) tags.

The M thematic categories serve as the feature set that describes items. Namely, each item $i \in \mathcal{I}$ of finite normalized size L_i is represented by a feature vector $\mathbf{f}^i$, whose jth element $\mathbf{f}^i(j), j \in [1,..,M]$ denotes the score of item i in feature j, i.e., how relevant is item i to thematic category j. These relevance scores assume values in [0,1] and are normalized so that $\sum_{j=1}^{M} \mathbf{f}^i(j) = 1, \forall i \in \mathcal{I}$. Replicas of each content item may be stored in any set of the small cell caches, besides the backend cache, depending on the actual caching decisions.

2.1.3 Users

At any point in time, each user $u \in \mathcal{U}$ is located within range of a different subset of the network (micro)cells. Theoretically, the user might access different content items through different caches, each time dynamically changing his/her association depending on the requested content.[3] In this paper, we assume that users do not change their association point in the network dynamically in response to content requests. Users are rather statically associated with certain cells, depending on the quality of radio signals and the load of the radio network, in line with user association practices in current wireless networks. A direct consequence of this assumption is that the caching decisions are made independently in each cache-enabled small cell.

Users are described by similar feature vectors $\mathbf{f}_u \in [0,1]^M$ as the content items. Each vector element $\mathbf{f}_u(j), j \in [1,..,M]$ expresses how much user u is interested in content classified under thematic category j. We normalize these values as well, i.e., $\sum_{j=1}^{M} \mathbf{f}_u(j) = 1, \ \forall u \in \mathcal{U}$.

3. The combination of caching with dynamic user associations and content routing has been investigated in literature, e.g., see [6].

Practically, content provision sites draw on the history of users' content downloads, and more broadly their interactions with the site, to infer these vectors. We elaborate on this in Section 6.1.

2.2 Content Preferences of Users

The demand for content items is time-varying. It grows for some finite time after the content item first becomes available for download, and then it gradually fades out [7], [8]. Our work concerns time scales over which the demand for each item can be considered "fixed", in the order of a few hours within a day [9], [10]. Namely, content demand predictions and caching decisions are made once every such an interval, and user content request patterns change slowly over that interval.

On the user side, we distinguish between inherent content preferences of users and the eventually issued content requests by them. Hence, each user u can be described by a content preference distribution, $p_u^{pref}(i), i \in \mathcal{I}$, with $\sum_{i \in \mathcal{I}} p_u^{pref}(i) = 1$, which captures his/her original preferences over all items. The preference of user u for item i, $p_u^{pref}(i)$, can be inferred from the feature vectors $\mathbf{f}_u$ and $\mathbf{f}^i$. In this paper, we use for this purpose the cosine similarity index, a_{ui}, of vectors $\mathbf{f}_u$ and $\mathbf{f}^i$ [11],[4] defined as

$$a_{ui} = \frac{\sum_{j=1}^{M} \mathbf{f}_u(j) \cdot \mathbf{f}^i(j)}{\sqrt{\sum_{j=1}^{M} \mathbf{f}_u(j)} \sqrt{\sum_{j=1}^{M} \mathbf{f}^i(j)}}.$$

Normalizing these index values over all items yields the content preference distribution p_u^{pref}

$$p_u^{pref}(i) = \frac{a_{ui}}{\sum_{i \in \mathcal{I}} a_{ui}}. \quad (1)$$

However, the content that users eventually request also depends on the recommendations issued to them, so that the probability with which user u requests item i, $p_u^{req}(i)$, differs from $p_u^{pref}(i)$. We describe our modeling approach to the recommendations' impact in the next paragraph.

2.3 The Impact of Recommendations on User Content Requests

The system recommendations affect the relative user demand for all content items. In principle, they boost the demand for the recommended items and at the same time proportionately decrease the demand for remaining items. It is less clear how recommender systems *quantitatively* modulate the a priori preferences of a user p_u^{pref} so as to yield the ultimate content request distribution p_u^{req}. Modeling in literature tends to be both intuition- and evidence-driven. In [13], for example, the recommendations are mapped to a new distribution p_u^{rec} over the content items and $p_u^{req}(i)$ is taken to be equal to $\max\{p_u^{pref}(i), p_u^{rec}(i)\}$. On the other hand, there is strong experimental evidence that both the number of

4. Measures of similarity between distributions such as the Proportional Similarity, the Kullback-Leibler distance, and the Hellinger coefficient could also be used in this respect [12].

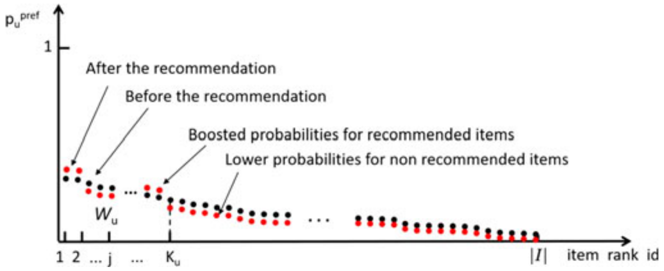


Fig. 2. Impact of recommendations on user requests. The content items are ranked in decreasing order of user preference and only items within the recommendation window are recommended. A priori content preference distribution for arbitrary user u, its recommendation window $\mathcal{W}_u$ of size K_u (black), and the resulting content request probability after recommendations (red).

recommended items and their order within the recommendation list have a high impact on the a posteriori distribution of the users' demand. For instance, the "Related Video" recommendation lists of YouTube is shown to be the main source of requests for its content, since items ranked higher up in a list of recommended items attract more user interest than items at lower positions (e.g., [4], [14]). This finding is more relevant for users that access content through small form-factor devices such as mobile phones, in which case it is less convenient to scroll down the whole recommendation list.

Hence, in modeling the impact of recommendations, we make two assumptions:

Assumption 2.1. *The impact of a recommendation for item i on the demand that is eventually expressed by a user u for this item is a non-increasing function $f(.)$ of both item's i position in the recommendation list, $rnk_u^L(i) \in [1, R]$, and the number of recommended items, R.*

In Section 6.1 we consider specific instances of p_u^{rec} that satisfy this assumption. For a more detailed discussion please refer to 6.1 in [5].

Assumption 2.2. *The content request distribution is a convex combination of the two distributions, p_u^{pref} and p_u^{rec} and is given by*

$$p_u^{req}(i) = w_u^r \cdot p_u^{rec}(i) + (1 - w_u^r) \cdot p_u^{pref}(i) \tag{2}$$

for each of the R items that are recommended to u, and by

$$\widetilde{p}_u^{req}(i) = (1 - w_u^r) \cdot p_u^{pref}(i) \tag{3}$$

for each one of the $(|\mathcal{I}| - R)$ items not recommended to user u. The recommendation weights w_u^r in Equations (2) and (3) express the importance user u attaches to recommendations.

Equations (2) and (3) capture the way recommendations shape content requests that are ultimately issued by user u. The request probabilities are boosted when compared to the initial ones for recommended items, and they decrease for non-recommended ones, so that the resulting content request distribution remains a probability distribution (see Fig. 2).

2.4 Engineering the User Recommendations

This capability of recommender systems to shape user demand for content renders them a powerful tool for content demand shaping. This way, recommender systems could be actively used to optimize network-centric performance objectives, in what marks a departure from their nominal user-oriented mission. Our approach to this is summarized in Fig. 2 and described in what follows.

Assume that the recommender system seeks to recommend R new items to each user u, where R may range from 1 up to a few (e.g., 5-10) items. Instead of issuing recommendations for the top R items in u's content preference distribution p_u^{pref}, the system selects R items among the ones residing within a *recommendation window* $\mathcal{W}_u$ that is defined by the top K_u items, where $K_u > R$, as shown in Fig. 2. Namely, the recommender system artificially *inflates* the set of candidate items for recommendation for each user u by a *user-specific* factor K_u/R. When doing so, the system preserves two properties addressing the ethical concerns described in Section 1 regarding the manipulation of recommendations:

- *It preserves the rank of recommended items in the original user content preferences.* If an item i is recommended at higher rank than item j, it holds necessarily that $p_u^{pref}(i) \geq p_u^{pref}(j)$.
- *It controllably bounds the distortion that its recommendations introduce to the original user content preferences.*

In the worst case, the system will end up recommending items that are ranked in positions $\{K_u - R + 1, K_u - R + 2, \ldots, K_u\}$ in decreasing order of user content preferences (ref. Fig. 2), instead of the items in top-R positions $\{1, 2, \ldots R\}$. We define the worst-case *user preference distortion* measure, Δ_u, to be

$$\Delta_u(K_u, R) = 1 - \frac{\sum\limits_{\substack{j: rnk_u(j) \in \\ [K_u - R + 1, K_u]}} p_u^{pref}(j)}{\sum\limits_{j: rnk_u(j) \in [1, R]} p_u^{pref}(j)}, \tag{4}$$

where $rnk_u(i), i \in \mathcal{I}$, is the rank of item i in user's u content preferences.

The denominator in Equation (4) equals to the total request probability of user u for the top R items, which are the ones a typical recommender system would recommend. The numerator, on the other hand, equals to the total request probability of user u for the bottom R items in the recommendation window. Hence, $\Delta_u(K_u, R)$ expresses the worst-case deviation from initial user request probabilities, that may result from the choices of our scheme when compared to a typical "honest" recommender system. As such, this metric is admittedly conservative and it denotes an upper bound on the possible distortion of original user request probabilities.

The size of the recommendation window, K_u, introduces an interesting trade-off. Higher K_u values allow for more flexibility in selecting items to recommend to users and shaping their demand in favor of caching efficiency, as it will be seen in the sequel. But at the same time, a higher K_u value may result in higher distortion of user preferences, as can be readily seen in Equation (4).

3 The Joint Caching and Recommendations Problem

The net outcome of these recommendations is that a given cache placement $\{P_z : z \in C\}$ may satisfy a portion of the total demand in the cell, as measured by the cache hit ratio[5]

$$H = \frac{\sum_{u\in\mathcal{U}}\sum_{i\in\mathcal{P}} p_u^{req}(i)}{\sum_{u\in\mathcal{U}}\sum_{i\in\mathcal{I}} p_u^{req}(i)}. \quad (5)$$

On the user side, the requirement is to maximize the number of requests that can be satisfied by the cell cache (cache hits). This results in lower content access delays and higher user QoE. At the same time, since fewer requests have to be satisfied by the backend server, the utilization of backhaul links is lower. In other words, the maximization of the cache hit ratio serves both types of requirements.

Formally, let $\{y_i\}$ and $\{x_{ui}\}$, $i\in\mathcal{I}, u\in\mathcal{U}$, be two sets of binary decision variables with $y_i=1$ when item i is cached and $y_i=0$, otherwise; and $x_{ui}=1$ when item i is recommended to user u and $x_{ui}=0$ when it is not. The objective of the joint caching and recommendation decisions are then to

$$\max_{\mathbf{y},\mathbf{x}} \quad \sum_{u\in\mathcal{U}}\sum_{i\in\mathcal{W}_u} y_i(x_{ui}p_u^{req}(i) + (1-x_{ui})\widetilde{p}_u^{req}(i)) \quad (6)$$

$$s.t. \quad \sum_{i\in\mathcal{I}} y_i L_i \le C \quad (7)$$

$$\sum_{i\in\mathcal{W}_u} x_{ui} = R, \quad \forall u\in\mathcal{U} \quad (8)$$

$$y_i, x_{ui} \in \{0,1\} \quad u\in\mathcal{U}, i\in\mathcal{W}_u. \quad (9)$$

In Equations (6) and (8), $\mathcal{W}_u$ denotes items within the recommendation window of user u. The cardinality of this set is

$$K_u = \max\{k|\Delta_u(k,R)\le r_d(u)\}, \quad (10)$$

where $r_d(u)\in[0,1)$ denotes the user-specific *preference distortion tolerance*, an upper bound on user preference distortion in Equation (4) that should not be exceeded for any user. Our formulation implies that the system could provide users with the opportunity to determine themselves how much distortion tolerance they are willing to tolerate. Inequality Equation (7) reflects the cache storage capacity constraint, whereas equalities Equation (8) ensure that exactly R items are recommended to every user.

3.1 Problem Complexity and Approximability Properties

We refer to the problem Equation (6), (7), (8), (9) as the Joint Caching and Recommendations Problem (JCRP).

Proposition 3.1. *The Joint Caching and Recommendations Problem is NP-complete.*

Proof. We work with the decision problem variant of JCRP, to which we refer as the Joint Caching and Recommendation Decision Problem (JCRDP).

JCR Decision Problem. Consider a set of users $\mathcal{U}$, a catalogue $\mathcal{I}$ of content items with lengths $L_i, i\in\mathcal{I}$, user demand distributions p_u^{pref} and recommendation window sets $\mathcal{W}_u$, $u\in\mathcal{U}$, a cache capacity C and a real number $Q>0$. Given Equations (7), (8), (9) and semantics of $\{y_i\}$, $\{x_{ui}\}$ as in the $JCRP$, are there a cache placement and item recommendations to each user so that

$$\sum_{u\in\mathcal{U}}\sum_{i\in\mathcal{W}_u} y_i(x_{ui}p_u^{req}(i) + (1-x_{ui})\widetilde{p}_u^{req}(i)) \ge Q, \quad (11)$$

where $p_u^{req}(i)$ and $\widetilde{p}_u^{req}(i))$ are given by Equations (2) and (3)?

Let a $JCRDP$ instance be denoted as $JCRDP(\mathcal{U},\mathcal{I}, p^{pref},\mathcal{W},R,C,L,Q)$. Given a cache placement and lists of recommendations for each user, we can check in polynomial time whether they satisfy or not Equations (7), (8) and (11). Hence, the $JCRDP$ lies in NP.

Consider an instance of $JCRDP$ with such preference distributions of users that their recommendation windows are of size R. Then, the recommendations are trivially fixed for each user; namely, $x_{ui}=1$ for each item i that ranks in the first R positions in the individual preference distributions and $x_{ui}=0$ otherwise. The $JCRDP$ now simplifies to asking whether there are $\{y_i\}$ variables so that:

$$\sum_{u\in\mathcal{U}} y_i v_i > Q \quad (12)$$

$$s.t. \sum_{i\in\mathcal{I}} y_i L_i \le C \quad (13)$$

$$y_i\in\{0,1\} \quad i\in\mathcal{I}, \quad (14)$$

where v_i is given by

$$v_i = \sum_{u\in\mathcal{U}} x_{ui}p_u^{req}(i) + (1-x_{ui})\widetilde{p}_u^{req}(i)).$$

Equations (12), (13), (14) constitute the decision version of the 0-1 Knapsack Problem (KSP), where l and v are the sizes and values, respectively, of the items that have to be packed within space C.

Moreover, assuming an oracle that solves $JCRDP$, returning the recommendations lists for all users and the cache placement, we can directly solve the decision version of the 0-1 KSP. Conversely, given the solution to the 0-1 KSP, we get the caching policy for the special case of the problem, where the recommendations are trivially obtained. □

As a first step towards an efficient approximation algorithm, we investigate the monotonicity and submodularity of the $JCRP$ ($JCRDP$) objective function. We recall the relevant definitions in what follows. Let S be a finite set of elements (interchangeably called universe or ground set) and X, Y any two subsets of S satisfying $X\subseteq Y\subseteq S$. A set function $f: 2^S\rightarrow R$ is called *monotone* if $f(X)\le f(Y)$, $\forall X,Y\subseteq S$. Moreover, f is called *submodular*, if for any element $e\in S\setminus Y$, it holds that $f_X(e)\ge f_Y(e)$, where $f_A(e)=f(A\cup e)-f(A)$. Namely, the extra marginal benefit from adding an item to a set decreases as this set grows larger. Similarly, f is called supermodular, if for any element $e\in S\setminus Y$, it holds that $f_X(e)\le f_Y(e)$.

We now claim that:

Proposition 3.2. *The objective function of the JCRP is non monotone.*

5. Since the caching decisions are made independently for each cache (ref. 2.1.3), we drop the index z from subsequent references to cache placements, i.e., we write $\mathcal{P}$ instead of $\mathcal{P}_z$.

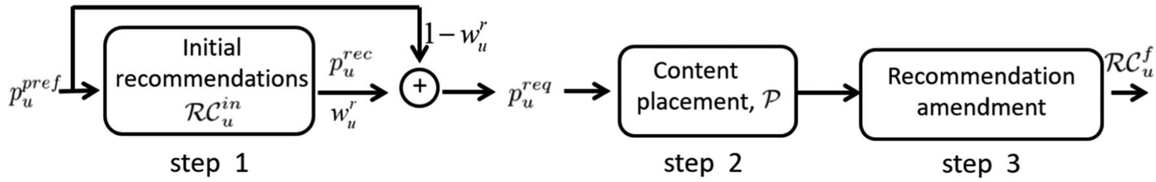


Fig. 3. Schematic outline of the three-step algorithm for the joint caching and recommendation problem.

Proof. The objective function in Equation (6) corresponds to the non-normalized expected cache hit ratio of the *JCRP*. Each content item i that is a candidate to be cached carries non-negative utility

$$\begin{aligned} v(i) &= \sum_{u\in\mathcal{U}} p_u^{req}(i) \\ &= \sum_{u\in\mathcal{U}}(1-w_u^r)p_u^{pref}(i) + \sum_{u:i\in\mathcal{W}_u} w_u^r p_u^{rec}(i). \end{aligned} \tag{15}$$

The first summand captures the original demand for the item and the second one the added value the item acquires due to recommendations made to users for it.

The objective function is monotone if it increases or, in the worst case, remains constant, every time a new item is added to the cache (y_i changes from 0 to 1) or to the recommendation list of any user (x_{ui} changes from 0 to 1).

Monotonicity with respect to y_i. Consider that we add item i to the cache, that is y_i=1. Then the cache hit ratio will increase by a factor

$$\sum_{u\in\mathcal{U}}(1-w_u^r)p_u^{pref}(i) + \sum_{u:x_{ui}=1} w_u^r p_u^{rec}(i)$$

and the system will have a positive gain.

Monotonicity with Respect to x_{ui}. Now consider the case where an item i is inserted in the recommendation list of user u, i.e., $x_{ui}(i)=1$. When this happens, the following scenarios are possible with respect to its impact on a specific user and the cache placement:

(c1) Item i is cached (y_i=1).

(c2) Item i is not cached (y_i=0) and is recommended at a lower rank than any other *cached* item in the recommendation list of user u.

(c3) Item i is not cached (y_i=0) and seizes the position of a cached item, say j with y_j=1, in the recommendation list of u (item j, and possibly other items, move to lower positions in this list).

In cases (c1) and (c2) the contribution of item i to the cache hit rate is positive and zero, respectively. On the contrary, in (c3), the contribution of i is negative since it reduces the value of the cached item j, as well as that of any other items that shift to lower-rank positions in the recommendation list of u (recall assumption 2.1). Thus, the objective function in Equation (6) is non monotone. □

Proposition 3.3. *The objective function of the JCRP is neither submodular nor supermodular.*

Proof. The objective function would be submodular (resp. supermodular) if it exhibited consistently decreasing (resp. increasing) returns as new items are added to the cache or the recommendation list of any user.

Consider sets X, Y to be two sets of cached items with $X \subseteq Y$. When a new item $i \in \mathcal{I}\setminus Y$ is added to the cache (y_i=1), the gain in cache hit ratio is positive and fixed (refer to Equation (16)).

When a new item i is recommended to user u, i.e., $x_{ui}=1$, the following cases are possible:

(c1) Item i is cached neither in X nor in Y.

(c2) Item i is cached in both X and Y.

(c3) Item i is cached in Y but not in X.

In (c1), the marginal gain in cache hit ratio is zero for both sets X and Y.

In (c2), the marginal gain in cache hit ratio exhibits diminishing returns and the objective function behaves as a submodular one. More specifically, if k_u cached items are already in the recommendation list of user u, the added value $w_u^r p_u^{rec}(i)$ that i generates for the cache hit ratio is that of a recommendation in the position k_u+1 in the list. But k_u can only increase for every user as the cache grows and the impact of a recommendation for i is a decreasing function of its rank in the list (ref. Assumption 2.1).

In (c3), the marginal gain in cache hit ratio increases with enlarged cache placements. Marginal gain is achieved only for set Y. For X, the cache hit ratio does not change (if there are no cached items following item i in the recommendation list of u) or even decreases (if i precedes one or more cached items in the recommendation list of u), as in case (c3) in Proposition 3.2. Thus, the objective function behaves as a supermodular one.

Overall, the objective function is neither submodular nor supermodular with respect to variables x_{ui}. □

Summarizing Propositions 3.2 and 3.3, the objective function is neither monotone nor submodular. Hence, more general techniques that yield approximability guarantees for this category of functions, e.g., [15], [16], and have been often applied in caching problems e.g., [17], [18], are not applicable to the *JCRP*.

In the following section, we propose an efficient heuristic algorithm for solving the JCRP. Although the algorithm does not lend to rigorous approximability analysis (ref. Section 4.2), it is computationally simple (ref. Section 4.1) and exhibits excellent performance (ref. Section 6).

4 An Algorithm for the Joint Caching and Recommendation Problem

4.1 Description of the Algorithm

Our Caching-aware Recommendations (CawR) algorithm proceeds in three steps, as shown in Fig. 3 and it is summarized in the pseudocode of Algorithm 1.

The first step is an *initialization* step where a provisional set of recommended items $\mathcal{RC}_u^{in}$ is derived for each user. Input to this step are the content preference probability distributions of users. Recommendations are made for the top-R items in the user preferences but, contrary to what would happen with a typical recommender system, these recommendations are not communicated to the user; they are only relevant as intermediate result of the algorithm's operation.

The second step is a *content placement* step, where we determine which content should be cached. To this end, we compute the content request probabilities according to Equations (2), (3). All content items are assigned utilities that equal the aggregate request probability they attract:

$$v(i) = \sum_{u \in \mathcal{U}} p_u^{req}(i) \quad i \in \mathcal{I}. \tag{16}$$

The optimal placement is then an instance of the 0-1 KSP. We use the Dynamic Programming (DP) FPTAS algorithm in ([19], section 8.2) to obtain an (1-ϵ), $\epsilon > 0$ approximation of the optimal solution; let $\mathcal{P}$ denote this placement.

Finally, in the *recommendation amendment* step, the original recommendations to users are amended so as to maximize the utility (i.e., expected attracted requests) of the cached content. To this end, we first identify for each user u the set of K_u items in his/her recommendation window from Equation (10). Then we compare the item sets $\mathcal{P}$ and $\mathcal{RC}_u^{in}$. Two possibilities exist:

- If $\mathcal{RC}_u^{in} \subseteq \mathcal{P}$, then the original recommendations derived in the initialization step remain intact (and the resulting user preference distortion is zero).
- If $|\mathcal{RC}_u^{in} \bigcap \mathcal{P}| = F_1 \in [0, R-1]$, then the F_1 items that appear in both sets are retained in the final recommendation list. $F = \min\{R - F_1, F_2\}$, where $F_2 = |(\mathcal{W}_u \setminus RC_u^{in}) \bigcap \mathcal{P}|$, most preferred cached items appearing in the recommendation window of u (but not in the recommendation set derived in the initialization step), are added to the recommendation list for u, replacing the bottom-F items in $\mathcal{RC}_u^{in}$; and, if there is still space ($F_1 + F < R$), the remaining recommendations are made for the $(R - F_1 - F)$ least popular items out of the $(R - F_1)$ remaining (non-cached) items in $\mathcal{RC}_u^{in}$.

The final set, $\mathcal{RC}_u^f$, of R items that are recommended to user u are in general different from the equal-size provisional set $\mathcal{RC}_u^{in}$ derived in the initialization step. Since the values of p_u^{req} after the recommendation amendment step are different, one might think that the algorithm returns to the content placement step and runs another round of steps 2 and 3. In Proposition 4.1 we prove that this is not the case, and that our algorithm terminates in a single round.

Proposition 4.1. *CawR terminates after a single execution of the recommendation amendment step.*

Proof. It suffices to show that the recommendation amendment step does not motivate any change in the cached content. Such a change would occur if the modification of recommendations resulted in change of item utilities (see Equation (16)) so that

$$\exists\ j \in \mathcal{P}, j' \notin \mathcal{P} : v(j') > v(j). \tag{17}$$

However, the recommendation amendment step increases the utility of an item already in the cache, when issuing recommendations for it to additional users than those in the initialization step; or, in the worst-case, it leaves it intact. At the same time, it reduces or leaves intact the utility of items that have not been included in $\mathcal{P}$ during the content placement step. Hence, condition in Equation (17) cannot be fulfilled and the cache placement $\mathcal{P}$ does not change. □

Hence, the algorithm essentially sets the cache placement on the basis of the original recommendations to users (with zero user preference distortion). Then, it selectively changes recommendations to nudge individual user preferences towards content that attracts demand from the overall user population. This way, the utility of the cached content, i.e., the demand it attracts, grows and the expected cache hit ratio increases.

In the toy example of Fig. 1, CawR first sets the provisional recommendation list RC_u^{in} to the R most preferred items for every user, as highlighted in the 2nd column. The resulting demand distributions are shown in the 3rd column. These recommendations are not shown to the end-users. In the second step, CawR computes the utility of each item from Equation (16) and caches items A,C,D. In the final step, CawR compares the recommendation window W_u of each user u with $\{A, C, D\}$ and determines the final lists of recommendations $\mathcal{RC}_u^f$, which are issued to the users. In our case, $\mathcal{W}_{u1} = \{A, B, C, D\}$, $\mathcal{W}_{u2} = \{A, C, D, E, F\}$, and $\mathcal{W}_{u3} = \{A, B, C, D, E\}$, respectively. The sets $\mathcal{RC}_u^f$ are those highlighted in the 4th column in Fig. 1.

Algorithm 1. The CawR Algorithm

Input: Probabilities p_u^{pref}, weights w_u^r, recommendation windows $\mathcal{W}_u$, $\forall u \in \mathcal{U}$, number of recommendations R
Output: Content placement $\mathcal{P}$ and recommended item sets $\mathcal{RC}_u^f, \forall u \in \mathcal{U}$.

Step 1:
Set the initial recommendation list $\mathcal{RC}_u^{in}$ to the R most preferred items of the user and p_{rec} to $f(rnk_u^L(i), R)$ (ref. Assumption 2.1).
for every user $u \in \mathcal{U}$ and item $i \in \mathcal{I}$ **do**
  Compute the content request probabilities $p_u^{req}(i)$ from Equations (2) and (3).
end for
Step 2:
for every item $i \in \mathcal{I}$ **do**
  Compute its utility from Equation (16).
end for
Use the DP $(1-\epsilon)$-approximation algorithm, $\epsilon > 0$, to solve the 0-1 KSP and derive the content placement $\mathcal{P}$.
Step 3:
for every user u **do**
  Add items in $\mathcal{RC}_u^{in} \bigcap \mathcal{P}$ to the final recommendation list $\mathcal{RC}_u^f$ for u.
  if $\mathcal{RC}_u^f$ is not full **then**
    Add to set $\mathcal{RC}_u^f$ items that are both cached and within $\mathcal{W}_u$ in decreasing order of preference probability till it is filled up.
  end if
  if $\mathcal{RC}_u^f$ is still not full **then**
    Add to the list items out of the remaining ones in $\mathcal{RC}_u^{in}$ in decreasing order of preference probability till the set is filled with R recommended items.
  end if
end for

4.2 Complexity of the CawR Algorithm

The manipulation of recommendations injects controllable distortion to the original user content preferences. In the worst case, the algorithm recommends to user u the bottom R of the top-K_u items, resulting in at most Δ_u distortion of the recommendations an "honest" recommender system would make.

In the first step, our algorithm sorts the list of the items and finds the most preferred ones for each user at time $O(|\mathcal{U}| \cdot |\mathcal{I}| \cdot \log |\mathcal{I}| + |\mathcal{U}| \cdot |\mathcal{I}|) = O(|\mathcal{U}| \cdot |\mathcal{I}| \cdot \log |\mathcal{I}|)$. In the second step the algorithm computes a utility for every item and then uses the DP FPTAS algorithm for the 0-1 KSP. This implies a complexity of $O(|\mathcal{I}| + |\mathcal{I}| \cdot |\mathcal{C}|) = O(|\mathcal{I}|^2)$ since the cache capacity is upper bounded by the total catalogue size. In the third step, the algorithm compares the items within the recommendation window of each user against the cache placement to define the final recommendations, leading to $O(|\mathcal{U}| \cdot \max_u(|\mathcal{W}_u|) \cdot |\mathcal{C}|)$. Since the size of the recommendation window is naturally bounded by the catalogue size, the total computational complexity of CawR is $O(|\mathcal{U}| \cdot |\mathcal{I}|^2)$.

In the two sections that follow, we draw on both analysis Equation (5) and simulations with real and synthetic datasets Equation (6) to gain further insight to the properties of the proposed algorithm.

5 Properties and Performance Bounds of our Algorithm

In this section, we compare the performance of our algorithm to benchmark recommender schemes and analyze its sensitivity to the maximum distortion tolerance parameters, $\{r_d(u)\}$ and the user content preference distributions, $\{p_u^{pref}\}$. We codify our results as propositions, referring the reader to [5] for their proofs and a more detailed discussion.

5.1 Benchmark Recommender Schemes

We consider three alternative schemes for determining which content to cache and which to recommend to each user. They serve as plausible comparison references for our algorithm and help set bounds for its performance.

Zero-Distortion (ZD) Scheme. The scheme recommends to each user the top-R items in his/her preferences and caches the C items attracting the highest aggregate demand, after accounting for the impact of recommendations. Hence, the recommendations follow precisely the user preferences, in line with what recommender systems nominally do, and the cache placement adapts to them.

Unbounded-Distortion (UD) Scheme. This scheme ranks items in order of decreasing aggregate demand over all users, after taking into account the factors $(1 - w_u^r)$ that weigh users' original preferences. It then caches the top-C of those and recommends to all users the *same* top-R items. Contrary to the ZD scheme, it is now the cache placement that determines the individual recommendations, catering for no bounds on the resulting distortion of the inherent user preferences.

Proposition 5.1. *When recommendations follow Assumptions 2.1 and 2.2 and the user preference distortion tolerances are relaxed, i.e.,* $W_u \equiv \mathcal{I}\ \forall u \in \mathcal{U}$ *in Equations (6) and (8), the UD scheme is the optimal solution to the JCRP.*

Least Frequently Used (LFU) Caching Algorithm. The algorithm caches items that attract the maximum aggregate demand over all users, but it does not recommend anything to them. It is known that LFU maximizes the cache hit ratio for a single cache in the absence of recommendations.

Denoting the expected cache hit ratio each scheme achieves by $H_s, s \in \{ZD, UD, CawR\}$, we can show that

Proposition 5.2. *The expected cache hit ratios achieved by the three schemes that issue recommendations satisfy:*

$$H_{ZD} \leq H_{CawR} \leq H_{UD}. \tag{18}$$

Thus, the performance of the unbounded- and zero-distortion schemes set an upper and a lower bound, respectively, for what is achievable with CawR. Notably, due to Proposition 5.1, H_{UD} sets an upper bound to the cache hit ratio under the optimal algorithm, H_{OPT}, for the JCRP (when the distortion constraints are not relaxed), and, eventually, for the cache hit ratio achieved by the CawR algorithm. Namely

$$H_{CawR} \leq H_{OPT} \leq H_{UD}. \tag{19}$$

It is less intuitive how LFU compares with the three schemes issuing recommendations since they weigh the inherent user content preferences with factors $(1 - w_u^r)$ to determine the cache placement. We explore these comparative relationships as well as the tightness of the bounds in Equation (19) with numerical simulations in Section 6.

5.2 Sensitivity Analysis to Parameters of CawR

5.2.1 Monotonicity and Submodularity of H_{CawR} with Respect to the Distortion Tolerance Parameters

The parameters $\{r_d(u)\}, u \in \mathcal{U}$ in CawR control the amount of distortion that is tolerated with respect to the original user content preferences. Higher $r_d(u)$ values imply larger W_u sizes, as can be seen from Equation (4), and higher chances to find cached items in them. This is interpreted into higher cache hit ratio when at least one of the currently issued recommendations is for a non-cached item. In that case, CawR can replace its recommendation(s) for one (or more) of these items with recommendations for one (or more) of the cached items that are included in the enlarged W_u.

Proposition 5.3. H_{CawR} *is a monotonically increasing function of the distortion tolerance parameters* $\{r_d(u), u \in \mathcal{U}\}$.

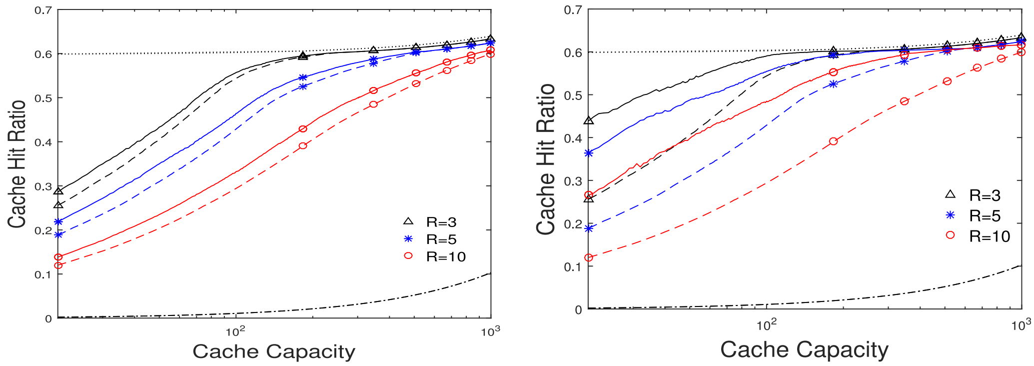
Less strict claims can be made about the submodularity of H_{CawR} with the $\{r_d(u)\}$ parameters [5].

5.2.2 Sensitivity of H_{CawR} to the Individual User Content Preferences

Although the exact values of H_{CawR} and H_{ZD} may vary widely depending on the user content preference distributions, $\{p_u^{pref}, u \in \mathcal{U}\}$, we can state that:

Proposition 5.4. *As the individual content preference distributions become more skewed,* H_{ZD} *and* H_{CawR} *tend to converge with each other.*

The key remark is that as the distributions $\{p_u^{pref}\}$ become more skewed, the recommendation windows become smaller.



(a) $r_d(u)$=0.01, $\forall u \in \mathcal{U}$ (b) $r_d(u)$=0.1, $\forall u \in \mathcal{U}$

Fig. 4. *Experiments with MovieLens traces*. Cache hit ratio, H, versus capacity for different number of recommended items, R. UD (dotted line) and LFU (dash-dot line) are not affected by R, solid lines correspond to CawR and dashed ones to ZD.

Therefore, the additional flexibility of CawR in selecting items to recommend tends to vanish and its recommendations to the users coincide with those of the ZD scheme. In fact, the recommendations of the two schemes exactly coincide when $K_u = R$.

6 Experimental Evaluation of our Algorithm

6.1 Datasets and Default Parameter Settings

We use both synthetic and real datasets [20] to derive the user and item feature vectors, $\mathbf{f}_u$ $u \in \mathcal{U}$ and $\mathbf{f}^i$ $i \in \mathcal{I}$, respectively, and then infer the user content preference distributions p_u^{pref} (see Section 2). The purpose of using real datasets is three-fold: a) to show how our model of user preferences and content items can be informed by real data; b) to drive a more "realistic" evaluation of the cache hit ratio CawR achieves; and (while doing so) c) to validate the theoretical claims made in Section 5. With synthetic datasets, on the other hand, we can control the experimentation settings and analyze the sensitivity of our algorithm to important variables such as the (dis)similarity of content preferences across users. Experiments were performed with the Movielens dataset as well as with synthetic datasets.

6.1.1 MovieLens Dataset

The analyzed dataset is a subset of the MovieLens project dataset [20]. These data include ratings of movies in a 0-5 rating scale by a big population of users and have been used widely in studies of recommender systems. Each of these movies is described by $M = 19$ thematic tags (e.g., drama, comedy, sports). In our experiments, we analyze different samples of 700 users and 10000 items of the catalogue, as retrieved in October 2016.

In populating our model, we draw on the fact that a user *did* rate a specific item, rather than the actual rating (s)he assigned to it. More specifically, the item tags are directly used to generate the item feature vectors. If an item is described by m different tags, the respective positions of the item feature vector are set to $1/m$ and the remaining ones to zero. For instance, if item i is described by "action" and "comedy", we set $\mathbf{f}^i(j) = 0.5$, for those j values corresponding to "action" and "comedy". Then, if $\mathcal{I}_r(u)$ is the set of items that user u has rated, we estimate the element $\mathbf{f}_u(j)$, $j \in [1, M]$, of user's u feature vector as

$$\mathbf{f}_u(j) = \frac{\sum_{i\in\mathcal{I}_r(u)} \mathbf{f}^i(j)}{\sum_{j=1}^{M} \sum_{i\in\mathcal{I}_r(u)} \mathbf{f}^i(j)}. \tag{20}$$

6.1.2 Synthetic Datasets

In this set of experiments, the elements of the two vectors are populated with values drawn from random probability distributions (the default one is the standard uniform distribution). The default parameter values for simulations with synthetic datasets are $|\mathcal{U}|$=150 users, $|\mathcal{I}|$=1000 items and $M = 8$ thematic areas.

For simulations with both types of datasets, the normalized item size L_i is sampled from a discrete uniform variable $\mathcal{U}(1, L_{\max})$, with default $L_{\max} = 4$. Our baseline assumption is that recommendations provide all R items in the list with an equal boost

$$p_u^{rec}(i) = 1/R, \tag{21}$$

which fades out with R. The intuition is that the fewer the recommendations are, the less cognitive load they demand from users to process them, and the more significant their impact is on the eventual user content requests. This assumption is more realistic for users accessing content from large-display devices, where it is more comfortable to scroll down the recommendations' list.

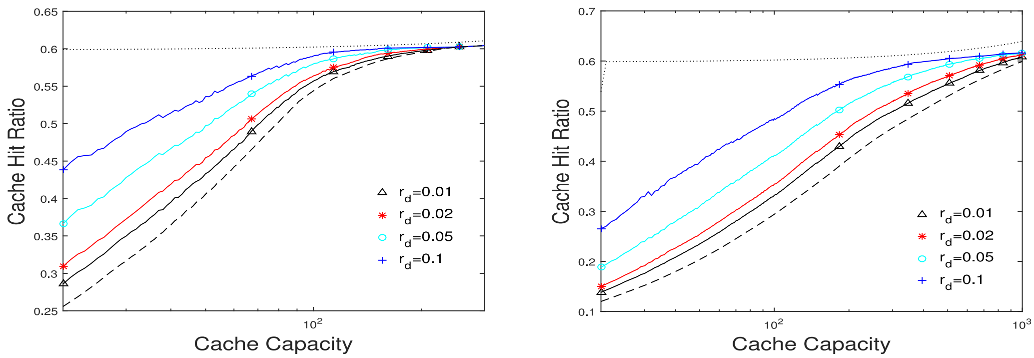
The choices of user recommendation weights, w_u^r, are aligned with experimental evidence in [14], according to which YouTube users request one of the top 10 recommended items with a probability that varies in [0.5, 0.7]. Thus, in our experiments the user recommendation weights are sampled from $\mathcal{U}(0.5, 0.7)$.

6.2 Trace-Driven Simulations

Figs. 4 and 5 validate the performance bounds we found analytically for the cache hit ratio under CawR in Section 5. The extent to which these bounds are tight depends on the number of issued recommendations per user, R, and the preference distortion tolerance parameter, r_d.

6.2.1 The Impact of the Number of Recommended Items on Cache Hit Ratio

On the one hand, R does not affect the performance of UD and LFU schemes. On the other hand, the achievable cache



(a) $R = 3$ (b) $R = 10$

Fig. 5. *Experiments with MovieLens traces.* Cache hit ratio, H, versus capacity as a function of the preference distortion tolerance, r_d (identical for all users). Dotted and dashed lines correspond to the UD and ZD scheme, respectively. The four intermediate curves correspond to CawR.

hit ratio under CawR and ZD deteriorates with higher R values. Intuitively, as the recommendation effect is spread across more items, some of it is wasted because it gets harder to find R cached items within the users' recommendation window (in the case of CawR) and among the top-R items in their preferences (in the case of ZD). Hence, the upper bound becomes looser as the number of recommendations grows. Interestingly, so does the lower bound when the relative gain of CawR over the zero-distortion scheme grows with R. This is the case as far as the size of the recommendation window size K_u is higher than R so that CawR has higher chances to find a cached item to recommend. For $R = 10$ and r_d=0.01, ZD performs up to 16.6 percent worse than CawR in terms of cache hit ratio, especially for really small instances of cache capacity (Fig. 4a). For r_d=0.1, this gap grows to 121 percent (Fig. 4b). At the same time, CawR reaches the 96 and 97 percent of the performance of the UD scheme, for r_d=0.01 and r_d=0.1 respectively, and realistic cache capacities (Fig. 4a, 4b). An alternative way to quantify the benefits of our algorithm is by looking into cache capacity requirements. For r_d=0.01, ZD needs up to 35 percent more cache capacity than CawR to reach the maximum achievable cache hit ratio when R=10 (Fig. 4a). For r_d=0.1, this value climbs to 147 percent (Fig. 4b).

In all cases, the LFU scheme is significantly outperformed by schemes that use recommendations.

6.2.2 The Impact of the Preference Distortion Tolerance, r_d, on Cache Hit Ratio

This enhanced flexibility of CawR is also the reason why it approaches the upper bound faster (i.e., for smaller cache) when r_d=0.1 (Fig. 4b), increasing its advantage over the ZD scheme that is insensitive to r_d.

The only scheme that is affected by the preference distortion tolerance parameter is CawR. Higher values of r_d provide the scheme with more flexibility in recommending items that are simultaneously parts of the cache placement and the user recommendation windows. Hence, as can be seen in Figs. 5a and 5b, but also in Figs. 4a and 4b, the CawR performance increases monotonically with r_d moving away from its lower bound (ZD scheme) towards its upper bound (UD scheme). Indicatively, for $R = 3$ and $r_d = 0.1$, we evidence with CawR cache hit ratios up to 71 percent higher than those under the ZD scheme at very small caches (Fig. 5a). The respective performance gain increases to 121 percent for $R = 10$ (Fig. 5b).

Regarding cache capacity requirements, even for distortion tolerance values smaller than $r_d = 0.01$ and $R = 3$, CawR needs up to 35 percent less storage capacity to converge to the upper bound confronted to the ZD scheme. This gain escalates up to 65 percent when $r_d = 0.1$ (Fig. 5a).

6.2.3 The Impact of Cache Capacity, Catalogue Size, and Number of Users on Cache Hit Ratio

A final set of experiments with the MovieLens traces addresses basic scaling properties of the algorithm, i.e., how the cache hit ratio varies as a function of the content catalogue size, cache size, and population of users.

Figs. 6(a-b), report the positive impact of ratio

$$\alpha = \frac{\text{cache capacity}}{\text{catalog size}} = \frac{C}{|\mathcal{I}|} \tag{22}$$

on the cache hit ratio. We further notice that the CawR and the ZD schemes approach the hit ratio of the UD scheme slower when compared to a system with fewer items.

6.3 Simulations with Synthetic Datasets

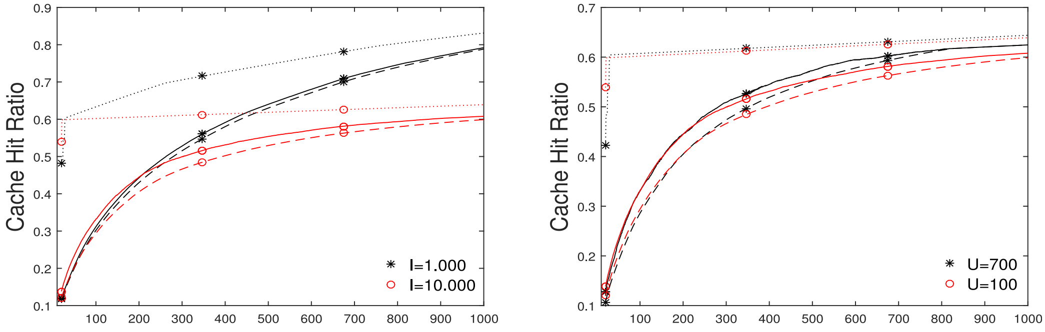
In this subsection, we explore how the performance of our algorithm is affected by the sensitivity of users to recommendations and the heterogeneity in their content preferences.

6.3.1 User Recommendation Weights

In the experiments of this subsection we consider two scenarios for the range of the user recommendation weights w_u^r in Equation (3), namely $w_u^r \in [0, 0.2]$ and $w_u^r \in [0.5, 1]$. The achieved cache hit ratio for each scenario is shown in Fig. 7.

As is expected, for small recommendation weights, the cache hit ratio under all recommendation schemes is small and comparable to that achieved under the recommendation-agnostic LFU scheme. However, CawR remains attractive when cache space is a concern. With storage spaces smaller than the 5 percent of the total catalogue size, CawR scores similarly to the ZD scheme, needing up to 35 percent less cache capacity. Compared to the UD scheme, CawR reaches the 89 percent of the performance of the UD scheme, performing always more than 14 percent better than ZD.

On the contrary, for larger recommendation weight values, UD, ZD and CawR schemes differ more clearly from each other. The cache hit ratio under CawR is 33 percent



(a) R = 10, r_d =0.01, U=700

(b) R = 10, r_d =0.01. I = 10000

Fig. 6. *Experiments with MovieLens traces*. Cache hit ratio, H, versus capacity as a function of catalogue size (left) and number of users (right). Dotted lines correspond to the UD scheme, dashed to the ZD scheme and solid lines to CawR.

higher than under ZD, for modest cache capacities in the order of 2 percent of the catalogue size. In terms of cache capacity, CawR equals ZD needing up to 32 percent less cache capacity even for capacities lower than the 2 percent of the total catalogue size. Compared to the UD scheme, for high recommendation weights, CawR reaches the 97 percent of the performance of the UD scheme, performing always more than 6 percent better than the ZD scheme. Overall, Fig. 7 indicates that the more users assign importance to recommendations made to them, the higher the advantage of our algorithm over the "honest" recommendation scheme.

6.3.2 Heterogeneity in User Content Preferences

The heterogeneity in user content preferences can manifest itself in multiple ways. To control this heterogeneity, we generate the user content preference distribution as a convex combination of two distributions: a user-specific component p_u^{ego}, which is modeled as described earlier in Section 2.2; and a second user-agnostic probability distribution, p^{ext}, which is modeled after a Zipf distribution, in line with experimental evidence in [21] and [22]. The preference of user u for item i is then given by:

$$p_u^{pref}(i) = w_u^e \cdot p_u^{ego}(i) + (1 - w_u^e) \cdot p^{ext}(i). \tag{23}$$

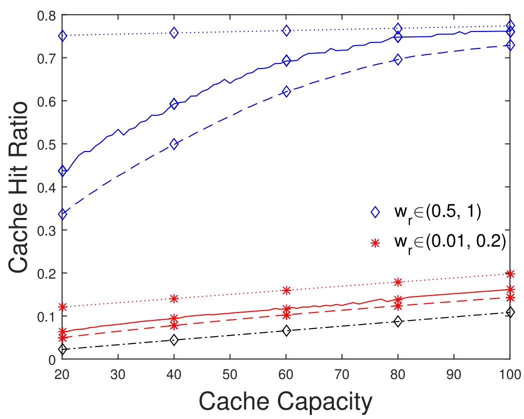


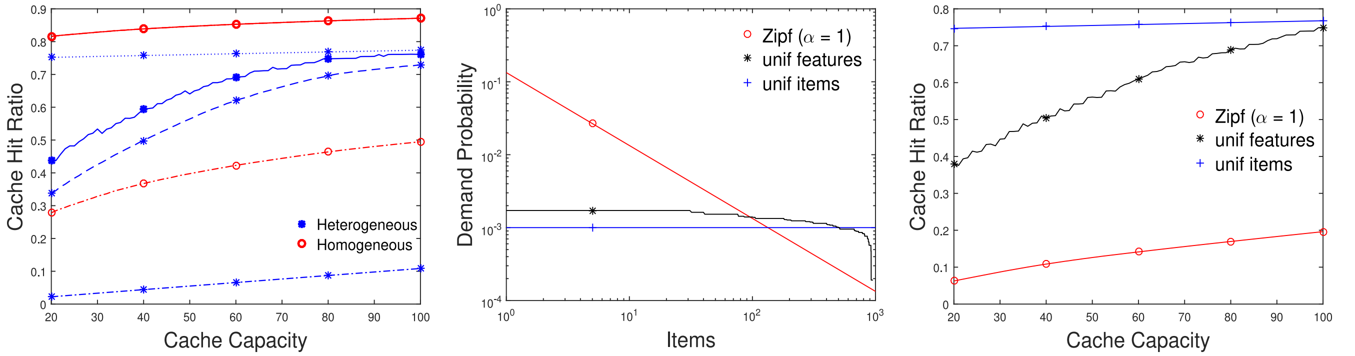
Fig. 7. *Experiments with Synthetic datasets*. Cache hit ratio versus capacity as a function of user recommendation weights, R=10, r_d=0.01, $|\mathcal{I}|$=1.000 items. Dotted lines correspond to the UD scheme, dashed lines to ZD, dash-dot lines to LFU, and solid lines to the CawR.

The rationale for introducing p_u^{ext} is that user preferences emerge as the combined result of individual preferences and external promotional and marketing actions that set global trends in content popularity. The weight w_u^e captures how these two influences mix for user u. Smaller w_u^e values emphasize the component distribution that is common across users, smoothing out the intrinsic user heterogeneity.

We have experimented with different values for w_e^u. In Fig. 8a we depict results from experiments where $w_e^u = 0$ ('o') and $w_e^u = 1$ ('*'), $\forall u \in \mathcal{U}$. For $w_u^e = 0$ the three schemes that issue recommendations collapse to one. Moreover, the total achieved cache hit ratio is higher when the content preference distributions are homogeneous ($w_u^e = 0 \ \forall u \in \mathcal{U}$). Both remarks can be explained intuitively since, in the rather extreme case where $w_u^e = 0 \ \forall u \in \mathcal{U}$, the recommendations issued by the three schemes to the users are identical and for the R most popular items in the cache.

A second experiment with user heterogeneity concerns the impact of the content preference distribution. Namely, we let the three functions shown in Fig. 8b serve as the user content preference distributions, permuting the order of items for each one of them. The functions include a Zipf distribution with skewness parameter α=1 ('o'), a uniform distribution over all items ('+'), and the distribution that emerges when the elements of the user feature vector $\mathbf{f}_u$ are drawn from the standard uniform distribution ('*').

Although the skewness of the last two distributions is similar, the achieved cache hit ratio, when the demand distributions follow them, is very different. When the demand is equally split over all items, the cache hit ratio is higher than when users' preferences are equally split over features. In the first case, the recommendation window of *every* user will contain the entire catalogue. Hence, CawR exhibits the highest possible flexibility when choosing items to recommend and cache. On the contrary, since the items in the catalogue can be very different, when user preferences are randomly spread among features, dissimilarities do emerge in the demand. Finally, the lowest cache hit ratio emerges when demand distributions follow a Zipf law with skewness parameter equal to unity. Such a result is expected from Proposition 5.4. These results essentially confirm that the performance of CawR, much as that of other caching schemes, is quite sensitive to the shape of the original user content preferences, which set hard bounds on what is achievable.



(a) $R = 3$, $r_d = 0.01$ (b) Content preference distributions (c) $R = 5$, $r_d = 0.01$

Fig. 8. *Experiments with Synthetic datasets.* Cache hit ratio versus capacity as a function of content preference heterogeneity (a,c) and content preference distributions (b). On the left plot, dotted lines correspond to the UD scheme, dashed lines to ZD, dash-dot lines to LFU, and solid lines to the CawR. All schemes are depicted for homogeneous ('o') and heterogeneous ('*') user demand distributions. On the central, we depict three different p_u^{pref} distributions (log-log-scale). On the right, the cache hit ratio under CawR is plotted for the three distributions in (b).

In Fig. 8c we only plot the cache hit ratio achieved by CawR, but its comparative advantage over ZD still holds. It is more pronounced for Zipf distributions with higher skewness paramater a and fades out as a tends to zero, inline with the prescriptions of Proposition 5.4.

7 Model Extensions and Future Work

7.1 Quantifying the Distortion of User Preferences

In our analysis, the distortion of user preferences is defined in Equation (4) as the normalized difference between the sums of preference probabilities for the actually recommended items and those recommended by the "honest" recommender system. This is clearly only one of many possible alternatives for quantifying the *distance* between the two sets of items.

In the current formulation of the JCRP, the distortion constraint is conservative. Any choice of items within the recommendation window will satisfy it; but so do combinations of items, some of which lie outside this window. In other words, the recommendation window, as defined in Equations (4) and (10), marks a sufficient but not necessary condition for bounding the distortion of the original user content preferences by r_d.

One apparent alternative would be to get rid of the hard notion of recommendation window and allow arbitrary selections of recommended items as far as the distortion measure does not exceed a certain threshold. Then in the formulation of the optimization problem in Section 3, we would replace the requirement $i \in \mathcal{W}_u$ in Equation (9) with an additional constraint

$$1 - \frac{\sum_{i \in \mathcal{I}} x_{ui} p_u^{pref}(i)}{\sum_{i=1}^{R} p_u^{pref}(i)} \leq r_d(u). \tag{24}$$

Although this may sound more intuitive, in practice it introduces considerable amount of computational complexity to the solution of Equation (9). Specifically, the choice of items would become a demanding combinatorial problem for non trivial instances.

7.2 Future Work

Understanding the precise way recommendations shape individual content preferences is a research thread per se, mainly pursued in the context of marketing and recommender systems (e.g., [23]). A plausible direction for future work consists in translating these largely experimental findings into solid models and validating our findings in this paper with them as starting points. This should address the human behavior heterogeneity, possibly leveraging data-driven machine learning techniques and principles from behavioral science.

In this work we consider one round of interaction between the users and the system. An interesting extension would be to consider how the dynamics of this interaction evolve in the long-term under such recommendations, addressing aspects like the trust users build in the system and including possible system penalization (e.g., churn effects). In a fully transparent system, where users are aware of the nudging recommendation practices, users might even choose themselves and trade the distortion their recommendations will undergo.

From the network/system point of view, a basic assumption for our analysis has been that the caching and recommendation decisions are made independently in each cache-enabled base station. Users are associated with a single cell according to a content-unaware criterion such as the received signal strength. An obvious problem extension would be to let users associate with multiple cells simultaneously, and access content dynamically from different caches. This setting would call for jointly optimizing the content caching, routing and recommendation decisions.

8 Related Work

Caching is a classic theme in data networking that has recently been experiencing a new research thread in the context of 4G and 5G mobile cellular networks and their small cell architectures [18]. One of the major concerns in these settings is how the user demand and content popularity can be accurately predicted. Besides the temporal locality of content demand [8], such predictions should account for user mobility and the small user populations that small cell caches present. To this end, proactive (e.g., [9], [10]) and reactive (e.g., [24]) caching approaches are proposed. At the same time, the combination of small with macro cells yields further possibilities to coordinate caching with content routing through different cells, as shown in [6].

Much sparser is the literature on the interplay between caching and recommender systems, which is the core theme of our paper. To the best of our knowledge, they are jointly considered in [13], [14], [25], [26], [27], [28], [29]. The first four are concerned with video traffic, whereas the rest address generic content.

The authors in [13] appear to be the first who base caching decisions on personalized recommendations issued by recommender systems. They use synthetic datasets to compare their recommendation-driven caching scheme with a conventional popularity prediction one. They report small gains for their scheme that disappear in practical scenarios with ten or more users served by the cache. In [25], the authors derive conditions under which it pays off to look into the spatial variation of content demand and fill the cache with content that is locally, rather than globally, most popular. Recommendation-based techniques are proposed, albeit not quantitatively evaluated, also in [27]. The aim is to determine how to replicate content within a CDN. Common to [13], [25], [27] is the fact that recommender systems are used as proxies for inferring the content popularity. We are distinctly different from them, regarding the way we approach recommender systems: not just as alternative predictors of content demand but also as demand-shaping tools that can actively be used to trade-off user- and network-centric performance objectives. In fact, our approach to increasing the utility of cache content could be seen as dual to the one taken by these three studies. Rather than struggling for accurate predictions of the users demand for content, our algorithm nudges the users demand towards content items that are common in their preferences.

Hence, conceptually most relevant to our work are the studies in [14], [26], [28], [29]. In the empirical study of [14], the authors achieve an increase of the YouTube cache hit ratio by reordering the videos shown to the users under the Related Videos list so that already cached ones occupy the first positions. In [28], the authors work with peer-to-peer (P2P) systems and propose heuristic recommendation algorithms accounting for both the content dissemination costs and user preferences. In [29] a proactive resource allocation and demand shaping framework is presented. The authors analyze theoretically the delivery costs and their reduction through proactive caching. In their recommender system, they modify the ratings shown to users in order to enhance the certainty of user demand. The ”soft cache hits” in [26] are motivated by the entertainment-oriented content consumption in current Internet. The idea is to recommend alternative (cached) content to users who request content not placed in the cache. The authors present a variety of optimization problems regarding soft-hits and cache placements in wireless environments and algorithms to solve the proposed problems.

Contrary to [14], [28] and [29], our focus is on wireless networks. Compared to all previous works, we formalize the joint caching and recommendation problem under *personalized* recommendations, i.e., different items are recommended to each user. We introduce the distortion tolerance measure to cater for how aggressively recommendations attempt to shape the demand for content in favor of the caching performance. Our results in Sections 5 and 6 suggest that the proposed approach could yield significant gains for the network performance and the users satisfaction without disrespecting their individual preferences.

9 Conclusions

Our work in this paper is motivated by the trend that wants both content providers and network operators also assuming roles in content delivery by owning and managing content delivery networks. We have looked into the possible benefits that can arise for the end users and the network when there is some coordination between recommender systems and caching decisions. This coordination, at least in this work, implies that recommender systems actively engineer the recommendations issued to users in ways that enhance the caching performance. Practically, this engineering consists in recommending content that may not necessarily rank top in the inferred user content preferences but still score high in them. By carefully nudging the individual user demand towards content that attracts preference from many users, the recommender system can result in higher cache hit ratios and enhanced QoE for end users.

Practitioners in areas like e-commerce are more familiar with this demand-shaping (more broadly: behavior-shaping) dimension of recommendations. There is strong evidence that the willingness to consume can be affected by online recommendations [23]. Their manipulation there aims at nudging consumers to spend more on products and services. We rather advocate their “manipulation” for “good” purpose, as an additional network traffic engineering tool that can be used to jointly optimize or balance user- and network-oriented performance objectives.

We have attempted to show this potential in the context of wireless networks with small cells. At the same time, we tried to explicitly and systematically address ethical concerns that are raised by this approach. Simulation results show that the proposed caching-aware recommender systems bring significant caching performance gains that persist over a broad range of parameters for the diversity in users' preferences, the capacity of caches, the number of recommended items and the content catalogue size.

Acknowledgments

This research has been co-financed by the Operational Program ”Human Resources Development, Education, and Lifelong Learning” and is co-financed by the European Union (European Social Fund) and Greek national funds. The work of the authors was supported by the European Commission H2020 Research Program under Grant Number 688768 netCommons (Network Infrastructure as Commons). The authors wish to thank Prof. Stratis Ioannidis for useful discussions. Part of this work has been accepted to the *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, May 1-4, 2017, Atlanta, GA, USA [1].

References

[1] L. E. Chatzieleftheriou, M. Karaliopoulos, and I. Koutsopoulos, “Caching-aware recommendations: Nudging user preferences towards better caching performance,” in *Proc. IEEE INFOCOM*, May 2017, pp. 784–792.

[2] T. H. Sarkissian, “The Business Case for Caching in 4G LTE Networks,” 2012, Online available: www.wireless2020.com/media/whitepapers.html, 2012.

[3] C. A. Gomez-Uribe and N. Hunt, "The Netflix recommender system: Algorithms, business value, and innovation," *ACM Trans. Manage. Inf. Syst.*, vol. 6, no. 4, pp. 13:1–13:19, Jan. 2016.
[4] R. Zhou, S. Khemmarat, and L. Gao, "The impact of YouTube recommendation system on video views," in *Proc. 10th ACM SIGCOMM Conf. Internet Meas.*, Nov. 2010, pp. 404–410.
[5] L. E. Chatzieleftheriou, M. Karaliopoulos, and I. Koutsopoulos, "Technical report: Jointly optimizing content caching and recommendations in small cell networks," Tech. Rep., 2017. [Online]. Available: http://www2.aueb.gr/users/liviachatzi/files/Reports/2018-CKK-TMC-Report -Jointly-Optimizing-Caching-Recommendations-Small-Cell-Networks.pdf
[6] K. Poularakis, G. Iosifidis, A. Argyriou, and L. Tassiulas, "Video delivery over heterogeneous cellular networks: Optimizing cost and performance," in *Proc. IEEE INFOCOM*, Apr. 2014, pp. 1078–1086.
[7] D. D. Vleeschauwer and K. Laevens, "Performance of caching algorithms for IPTV on-demand services," *IEEE Trans. Broadcasting*, vol. 55, no. 2, pp. 491–501, Jun. 2009.
[8] S. Traverso, M. Ahmed, M. Garetto, P. Giaccone, E. Leonardi, and S. Niccolini, "Temporal locality in today's content caching: Why it matters and how to model it," *ACM Comput. Commun. Rev.*, vol. 43, no. 5, pp. 5–12, 2013.
[9] S. Shukla and A. A. Abouzeid, "Proactive retention aware caching," in *Proc. IEEE INFOCOM*, May 2017, pp. 766–774.
[10] J. Tadrous and A. Eryilmaz, "On optimal proactive caching for mobile networks with demand uncertainties," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2715–2727, Oct. 2016.
[11] J. Leskovec, A. Rajaraman, and J. D. Ullman, *Mining of Massive Datasets*, 2nd Ed. Cambridge Univ. Press: Cambridge, U.K., 2014.
[12] J. Vegelius, S. Janson, and F. Johansson, "Measures of similarity between distributions," *Springer, Quality Quantity*, vol. 20, no. 4, pp. 437–441, 1986.
[13] M. Verhoeyen, J. D. Vriendt, and D. D. Vleeschauwer, "Optimizing for video storage networking with recommender systems," *Bell Labs Tech. J.*, vol. 16, no. 4, pp. 97–113, 2012.
[14] D. K. Krishnappa, M. Zink, C. Griwodz, and P. Halvorsen, "Cache-centric video recommendation: An approach to improve the efficiency of youtube caches," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 11, no. 4, pp. 1–20, Jun. 2015.
[15] A. Kulik, H. Shachnai, and T. Tamir, "Approximations for monotone and nonmonotone submodular maximization with knapsack constraints," *Math. Oper. Res.*, vol. 38, no. 4, pp. 729–739, 2013.
[16] C. Chekuri, J. Vondrk, and R. Zenklusen, "Submodular function maximization via the multilinear relaxation and contention resolution schemes," *SIAM J. Comput.*, vol. 43, no. 6, pp. 1831–1879, 2014.
[17] S. C. Borst, V. Gupta, and A. Walid, "Distributed caching algorithms for content distribution networks," in *Proc. IEEE INFOCOM*, Mar. 2010, pp. 1–9.
[18] 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.
[19] V. V. Vazirani, *Approximation Algorithms*. New York, NY, USA: Springer, 2001.
[20] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *ACM Trans. Interact. Intell. Syst.*, vol. 5, no. 4, pp. 1–19, Dec. 2015.
[21] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and zipf-like distributions: Evidence and implications," in *Proc. IEEE INFOCOM*, Mar. 1999, pp. 126–134.
[22] P. Gill, M. F. Arlitt, Z. Li, and A. Mahanti, "Youtube traffic characterization: A view from the edge," in *Proc. 7th ACM SIGCOMM Conf. Internet Meas*, Oct. 2007, pp. 15–28.
[23] G. Adomavicius, J. Bockstedt, S. P. Curley, and J. Zhang, "Effects of online recommendations on consumers' willingness to pay," in *Proc. ACM Workshop Human Decision Making Recommender Syst.*, Sep. 2012, pp. 40–45.
[24] M. Leconte, G. S. 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.
[25] S. Dernbach, N. Taft, J. Kurose, U. Weinsberg, C. Diot, and A. Ashkan, "Cache content-selection policies for streaming video services," in *Proc. IEEE INFOCOM*, Apr. 2016, pp. 1–9.
[26] P. Sermpezis, T. Spyropoulos, L. Vigneri, and T. Giannakas, "Femtocaching with soft cache hits: improving performance with related content recommendation," in *Proc. 2017 IEEE Global Communications Conf. (GLOBECOM 2017)*, Singapore, December 4–8, 2017, pp. 1–7.
[27] M. A. Kâafar, S. Berkovsky, and B. Donnet, "On the potential of recommendation technologies for efficient content delivery networks," *ACM Comput. Commun. Rev.*, vol. 43, no. 3, pp. 74–77, 2013.
[28] D. Munaro, C. Delgado, and D. S. Menasché, "Content recommendation and service costs in swarming systems," in *Proc. IEEE Int. Conf. Commun*, Jun. 2015, pp. 5878–5883.
[29] J. Tadrous, A. Eryilmaz, and H. E. Gamal, "Proactive content download and user demand shaping for data networks," *IEEE/ACM Trans. Netw.*, vol. 23, no. 6, pp. 1917–1930, Dec. 2015.

Livia Elena Chatzieleftheriou received the diploma degree in applied mathematical and physical sciences from the National Technical University of Athens (NTUA), Greece, in 2015. She is currently working toward the PhD degree in the Department of Informatics, Athens University of Economics and Business (AUEB), Athens, Greece. Her research interests include wireless networks, caching, and recommender systems.

Merkouris Karaliopoulos received the diploma degree in electrical and computer engineering from the Aristotelian University of Thessaloniki, Greece, in 1998, and the PhD degree in electrical and computer engineering from the University of Surrey, United Kingdom, in 2004. He is a senior researcher with the Department of Informatics, Athens University of Economics and Business, Greece. He had postdoctoral research appointments with the Computer Science Department, University of North Carolina at Chapel Hill, North Carolina, in 2006, with the Swiss Federal Institute of Technology (ETH Zurich), Zurich, Switzerland (2007-2010), with the Department of Informatics and Telecommunications, University of Athens, Greece (2010-2013), and with the Center of Research and Technology Hellas (CERTH), Thessaloniki, Greece (2013-2015). His research interests include wireless networking and performance analysis.

Iordanis Koutsopoulos received the diploma degree in electrical and computer engineering from the National Technical University of Athens (NTUA), Greece, in 1997 and the MS and PhD degrees in electrical and computer engineering from the University of Maryland, College Park (UMCP), Maryland, in 1999 and 2002, respectively. He is now an associate professor with the Department of Informatics, Athens University of Economics and Business (AUEB). He was an assistant professor (2013-2015) with AUEB. Before that, he was an assistant professor (2010-2013) and lecturer (2005-2010) with the Department of Computer Engineering and Communications, University of Thessaly. He received the single-investigator European Research Council (ERC) competition runner- up award for the project RECITAL: Resource Management for Self-coordinated Autonomic Wireless Networks (2012-2015). His research interests include network control and optimization, with applications on wireless networks, social and community networks, crowd-sensing systems, smart-grid, and cloud computing.