Caching-aware Recommendations: Nudging User Preferences towards better Caching Performance

Livia Elena Chatzieleftheriou, Merkouris Karaliopoulos, Iordanis Koutsopoulos
Department of Informatics
Athens University of Economics and Business, Athens, Greece
Email:{liviachatzi, mkaralio, jordan}@aueb.gr

***Abstract*—Caching decisions by default seek to maximize some notion of social welfare: the content to be cached is determined so that the maximum possible aggregate demand over all users served by the cache is satisfied. Recommendation systems, on the contrary, are oriented towards user individual preferences: the recommended content should be most appealing to the user so as to elicit further content consumption.**

In our paper we explore how these, phenomenically conflicting, objectives can be jointly addressed. To this end, we depart radically from current practice with recommender systems, and we approach them as network traffic engineering tools that can actively shape content demand towards optimizing user- and network-centric performance objectives. We formulate the resulting joint theoretical optimization problem of deciding on the cached content and the recommendations to each user so that the cache hit ratio is maximized subject to a maximum tolerable distortion that the recommendation should undergo. We conclude on its complexity, and we propose a practical algorithm for its solution. The algorithm is essentially a form of lightweight control over the 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.

I. 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 [1]. 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% of the hours streamed at Netflix [2], whereas the related video recommendations generate about 30% of the overall views on YouTube [3].

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 mobile apps that interact with the 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), content servers within other networks. In the case of cellular networks, these 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 (QoE) 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. Secondly, by *routing user content requests dynamically* to different cell caches, it is possible to 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.*, *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

[1]https://openconnect.netflix.com/en/
[2]https://peering.google.com/

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* 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 aiming at higher caching efficiency and better users' QoE.*

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 range 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 model that captures the coupling between caching decisions and recommendations to a set of users. Our viewpoint to recommender systems raises some concerns, not least ethical ones. To this end, we introduce a measure called user preference distortion, to quantify how much the engineered recommendations distort the original user content preferences.
- We formulate the constrained optimization problem of maximizing the cache hit ratio subject to the maximum tolerable distortion of issued recommendations. The distortion tolerance is embedded as constraint to the problem formulation and marks an equally important dimension for assessing possible solutions to it.
- We devise a low-complexity practical heuristic algorithm that solves efficiently the problem above. 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. The algorithm and its complexity are analyzed in section IV.
- Finally, in section V, we provide a holistic assessment of the joint caching and recommendation algorithm, addressing both performance aspects and ethical concerns.

II. System model

A. Caches, content, 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, $\mathcal{U}$, as shown in Figure 1.

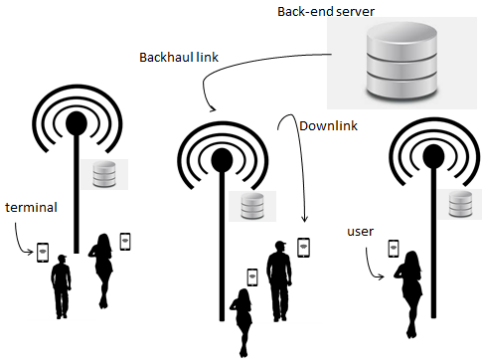


Fig. 1. Illustration of the system model. Caches are co-located with the wireless network microcells and serve users with content of their preference. Users interact with the system through a recommender system application. The content placed at the caches is determined with the help of a recommender system.

1) Caches: Caches of limited storage, $C_i, i \in \mathcal{C}$, are co-located with the wireless network microcells. The range of these cells is assumed to be identical and typically amounts to a few hundred meters. 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 catalog.

2) Content: Content items are relevant to one or more thematic categories. The detail and resolution of this categorical separation may vary (*e.g.*, "soccer" might be a distinct category but it may also be further split into English/French/Spanish soccer). We denote the number of different thematic categories by M. Such 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 $i \in \mathcal{I}$ has a finite size L_i and is represented by a feature vector f_i, whose j^{th} element $f_i(j), j \in [1, .., M]$ 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 $\sum_{j=1}^{M} f_i(j) = 1 \ \forall i \in \mathcal{I}$. 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.

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, (s)he could 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. This is in line with the user association practices in current wireless networks.

Users are described by similar feature vectors f_u as the content items. Now, each vector element $f_u(j), j \in [1, .., M]$, expresses how much user u is interested in content classified under thematic category j. We normalize these values as well,

[3]The combination of caching with dynamic user associations and content routing has been investigated in literature, *e.g.*, see [4].

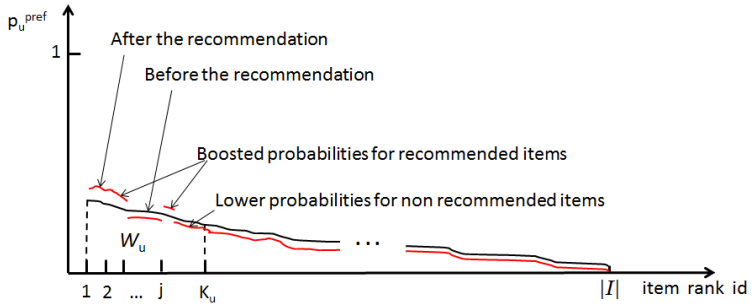


Fig. 2. A priori content preference distribution for arbitrary user u, its recommendation window $\mathcal{W}_u$ of size K_u (black), and the resulting content request probability after recommendations (red). The content items are ranked in decreasing order of user preference and only items in the recommendation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

i.e., $\sum_{j=1}^{M} f_u(j) = 1 \ \forall u \in \mathcal{U}$. Practically, content provision sites draw on the history of users' content downloads, and more broadly their interactions with the site, to infer these vectors.

B. Modeling the content preferences of users

The demand for content items is generally time-varying. Its dynamics usually evolve over a finite interval from the moment the content item enters the system, which may include several days or months [5]–[7]. 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. Namely, content demand predictions and caching decisions are made once every such an interval and the user content request patterns change slowly over the same interval.

On the user side, we distinguish between the inherent content preferences of users and the eventually issued content requests by them. More formally, each user u can be described by a content preference distribution, p_u^{pref}, $\sum_{i \in \mathcal{I}} p_u^{pref}(i) = 1$, over the different content items. This captures the original preferences of u over all items. Technically, the preference of u for item i, $p_u^{pref}(i)$, can be inferred from the feature vectors of u and i. In this paper, it is taken to be the cosine similarity index, a_{ui} of the two vectors [8], although any other measure of distributional similarity would be equally appropriate [9].

$$a_{ui} = \frac{\sum_{j=1}^{M} f_u(j) \cdot f_i(j)}{\sqrt{\sum_{j=1}^{M} f_u(j)} \sqrt{\sum_{j=1}^{M} f_i(j)}}$$

Normalizing these index values over all items for a given user u yields the content preference distribution p_u^{pref}.

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

However, the content users eventually request from the network also depends on the recommendations of the content provision site so that the probability with which user u requests item i, $p_u^{req}(i) \neq p_u^{pref}(i)$. We describe our modeling approach to the recommendations' impact separately in the next subsection since it forms a critical part of the overall system model.

C. The impact of recommendations on user content requests

Evidence of this impact is reported in [10] [3]. In both studies, it is found that a significant portion of content views is the result of recommendation mechanisms deployed by the sites. Hence, our starting modeling assumption is that a recommendation, at least when issued for a content item i that indeed matches the preferences of user u, increases the a priori probability that the user requests it.

This capability of recommender systems to shape the user demand for content turns them to a powerful tool at the hands of the content provider. In settings such as those outlined in the introduction section, where the content provider has also control over the cached content, the recommender system could be actively used to optimize network-centric performance objectives. This implies a departure from the nominal user-oriented mission of recommender systems and raises issues of ethical nature: how acceptable would be for users a system that actively manipulates its recommendations to them to serve its own performance objectives?

We argue that the answer to this question is not univocal but rather depends on the specific way recommendations are handled. 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. Instead of issuing recommendations for the top R items in u's content preference distribution p_u^{pref}, the system selects R items out of a *recommendation window* $\mathcal{W}_u$ determined by the top K_u items, $K_u > R$, as shown in Fig. 2a. Namely, our recommender system *inflates* the set of candidate-for-recommendation items for each user by a user-specific factor K_u/R. At the same time, it has two properties addressing the aforementioned ethical concerns regarding the manipulation of recommendations:

1) it preserves the user preference rank of recommended items: If an item i is recommended at higher rank than item j, it holds necessarily that $p_u^{pref}(i) \geq p_u^{pref}(j)$.

2) it controllably bounds the distortion it introduces to user preferences: In the worst case, the system will end up recommending items that are ranked in positions $[K_u - R + 1, K_u]$ in p_u^{pref} (ref. Fig. 2b), instead of the items in positions $[1, R]$. We can define the *user preference distortion* measure, Δ_u to be the ratio

$$\Delta_u(K_u, R) = 1 - \frac{\sum_{j=K_u-R+1}^{K_u} p_u^{pref}(j)}{\sum_{j=1}^{R} p_u^{pref}(j)}. \tag{2}$$

The denominator of the fraction in (2) equals to the cumulative user preference for the top R items, the ones a typical recommender system would recommend. The numerator, on the other hand, equals the user preference for the bottom R items in the recommendation window, *i.e.*, the rightmost items within $\mathcal{W}_u$ that our recommendation scheme is allowed to select. Hence, $\Delta_u(K_u, R)$ expresses the worst-case deviation, in terms of original user preferences for the recommended

items, that may result from the choices of our scheme when compared to a typical "honest" recommender system. As such, it poses an upper bound on the possible distortion of the original user content preferences.

Therefore, the size of the recommendation window K_u introduces a trade off. Higher K_u values let more flexibility in selecting items to recommend to users and shaping their demand in favor of caching efficiency. But they may result in higher distortion of user content preferences, as quantified by Δ_u in (2).

The system recommendations affect the relative user demand for all content items: they boost the demand for the recommended items and proportionately decrease the demand for remaining items. Yet, we lack experimental evidence about the exact way that recommendations modulate the a priori content preference distribution p_u^{pref} to yield the ultimate content request distribution p_u^{req}. As a result, modeling practice in literature is intuition- rather than evidence-driven. In [11], for example, the recommendations are mapped to a new distribution p_{rec}^u over the content items and $p_u^{req}(i)$ is taken to be equal to $max(p_u^{pref}(i), p_u^{rec}(i))$.

Here, we assume that recommendations provide all R items with an equal boost,

$$p_u^{rec}(i) = 1/R, \tag{3}$$

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 on the user content requests. Then the ultimate content request distribution is a convex combination of the two distributions, p_u^{pref} and p_u^{rec} so that

$$p_u^{req}(i) = w_u^r \cdot p_u^{rec}(i) + (1 - w_u^r) \cdot p_u^{pref}(i) \tag{4}$$

for the R items that are recommended to u, and

$$\widetilde{p}_u^{req}(i) = (1 - w_u^r) \cdot p_u^{pref}(i) \tag{5}$$

for the $|\mathcal{I}| - R$ items that are not recommended. The *recommendation weights* w_u^r in (4) and (5) express the importance user u assigns to recommendations. Equations (4) and (5) capture the way recommendations shape the content requests that are ultimately issued by user u. The request probabilities are boosted when compared to original preferences for recommended items[4] and decrease for non-recommended ones, so that the resulting content request distribution remains normalized (see Fig. 2b).

III. The joint caching and recommendations problem

The coordination of caching decisions, *i.e.*, which items to store in the cell cache(s), with the recommendation decisions, *i.e.*, which items to recommend to each user in the cell, aims at best serving both user- and network-centric performance measures. In what follows, we analyze these objectives and formulate the problem when the user-to-cell associations are fixed and do not take into account the user content preferences and cached content. This is in line with current practices in all wireless cellular networks, where users choose cell to associate with on the basis of radio signal quality parameters. In these cases, the caching and recommendation decisions for each cell are independent.

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, in our case study the aim is to

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

$$s.t. \quad \sum_{i\in\mathcal{I}} y_i L_i \leq C \tag{7}$$

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

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

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

$$K_u = max\{k | \Delta_u(k, R) \leq r_d\} \tag{10}$$

where $r_d \in [0, 1)$ is the upper bound on user preference distortion in (2) that should not be exceeded for any user.

There are two sets of binary decision variables: y_i=1 when item i is cached and y_i=0, otherwise; x_{ui}=1 when item i is recommended to user u and x_{ui}=0 when it is not. The optimal variable values are sought under two types of constraints. Inequality (7) reflects the cache storage capacity constraint, whereas equalities (8) ensure that R items are recommended to every user.

This problem is a (non-linear) generalization of the extensively studied 0-1 Knapsack problem (0-1 KSP) and, thus, it is NP-hard. In the following section, we propose a heuristic algorithm for the joint caching and recommendation problem.

IV. A heuristic algorithm for the joint caching and recommendation problem

A. Description of the algorithm

The algorithm proceeds in three steps, as shown in Fig. 3.

First, in the initialization step, the provisional set of recommended items $\mathcal{RC}_u^{in}$ is derived for each user. Input to this are the content preference probability distributions of users, and recommendations are made for the top-R items in the user preferences. Contrary to what would happen with a typical recommender system, these recommendations are not communicated to the end user. They are only relevant as intermediate result of the algorithm's operation.

[4]For practical values of R and realistic content preference distributions, the $1/R$ boost term is higher than the preference probability for any single item

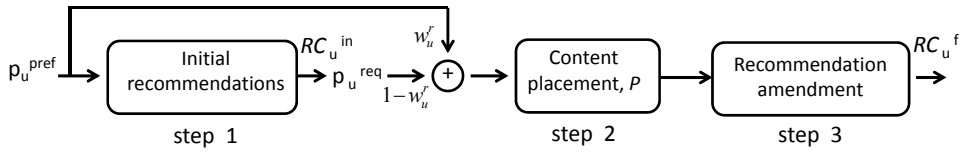


Fig. 3. Schematic outline of the three-step heuristic algorithm for the joint caching and recommendation problem. The three steps are presented in Algorithm 1 and, in more detail, in section IV-A.

Then, in the *content placement* step, we determine which content should be cached. To this end, we compute the content request probabilities according to (3), (4), and (5). All content items are assigned utilities that equal the aggregate content request probability they attract

$$util(i) = \sum_{u \in \mathcal{U}} p_u^{req}(i) \quad i \in \mathcal{I} \tag{11}$$

The optimal placement is then an instance of the 0-1 KSP. We use the Dynamic Programming (DP) FPTAS algorithm in ([12], §8.2) to obtain an 1-$\epsilon, \epsilon > 0$ approximation of the optimal solution. We denote $\mathcal{P}$ the resulting placement.

Finally, in the *recommendation amendment* step, the original recommendations to users are amended in an attempt 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 (10). Then we compare the item sets $\mathcal{P}$ and $\mathcal{RC}_u^{in}$ – two possibilities exist:

- $\mathcal{RC}_u^{in} \subseteq \mathcal{P}$: then the original recommendations derived in the initialization step remain intact (and the resulting user preference distortion is zero).
- $|\mathcal{RC}_u^{in} \bigcap \mathcal{P}| = F_1 < R$: then the F_1 items that appear in both sets are retained in the final recommendation list; $F = min(R - F_1, F_2)$, $F_2 = |(\mathcal{W}_u \setminus RC_u^{in}) \bigcap \mathcal{P}|$, 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 of items $\mathcal{RC}_u^f$ that are recommended to user u are in general different from the provisional set $\mathcal{RC}_u^{in}$ derived in the initialization step.

The algorithm is summarized in the pseudocode of Algorithm 1.

Algorithm 1 Joint recommendation and caching algorithm

Input: Probabilities p_u^{pref} and weights w_u^r, $\forall u \in \mathcal{U}$
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 $1/R$
for every user $u \in \mathcal{U}$ and item $i \in \mathcal{I}$ **do**
  Compute the content request probabilities $p_u^{req}(i)$ from (4) and (5).
end for
Step 2:
for every item $i \in \mathcal{I}$ **do**
  Compute its utility from (11)
end for
Use the Dynamic Programming $1 - \epsilon, \epsilon > 0$-approximate algorithm for solving the Knapsack problem 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 the list items that are both cached and within 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 it is filled up.
  end if
end for

B. Properties of the algorithm

It is straightforward to show that:

Proposition IV.1. *The cache content and the recommendations are stabilized by the end 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 11) so that

$$\exists\ j \in \mathcal{P}, j' \notin \mathcal{P} : util(j') > util(j) \tag{12}$$

However, the recommendation amendment step increases the utility of items already in the cache, when issuing recommendations for them 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 (12) 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 (injecting controllable distortion) to enhance the utility of the cached content (*i.e.*, the expected cache hit ratio) by nudging individual user preferences towards content that attracts demand from the overall user population. In the worst case, the algorithm recommends the bottom R of the top K_u items to every user u, leading at most to a Δ_u distance from the recommendations an 'honest" recommender system would do.

In the first step, our heuristic algorithm sorts the list of the items and finds the most preferred ones for each user, needing $O(|\mathcal{U}| \cdot |\mathcal{I}| \cdot \log |\mathcal{I}| + |\mathcal{U}| \cdot |\mathcal{I}|) = O(|\mathcal{U}| \cdot |\mathcal{I}| \cdot \log |\mathcal{I}|)$. In the second step the algorithm computes a utility for every item and then uses the DP 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 catalog size, the total computational complexity of our heuristic is $O(|\mathcal{U}| \cdot |\mathcal{I}|^2)$.

Unfortunately we cannot derive approximation guarantees for the cache hit ratio our algorithm achieves when compared to the optimal one without making very specific assumptions regarding the dissimilarity in the preferences of users. In the section that follows, we carry out simulations to gain more insights about the performance properties of our algorithm.

V. Evaluation of the heuristic algorithm

A. Simulation methodology

1) Datasets: In our experiments, we use synthetic datasets to model end user preferences and content items. This way we can control the experimentation settings and analyze the sensitivity of our algorithm to different unknowns such as the (dis)similarity of content preferences across users.

We model users and content items as feature vectors f_u and f_i, respectively (see section II). The elements of the two vectors are populated with values drawn randomly from the standard uniform distribution. The feature space is common to both and corresponds to M predefined thematic areas (M=8 in subsequent simulation runs).

2) Default parameters: Unless otherwise stated, the default parameter values for our simulations are $|\mathcal{U}|$=150 users and $|\mathcal{I}|$=1000 items, while the item size is sampled by a discrete uniform variable $\mathcal{U}\{1, L_{max}\}$. In our simulations, we recommend R=3, 5 or 10 items, whereas r_d values range in [0.01, 0.1]. The recommendation weights of users are let vary uniformly over different intervals within [0,1].

3) Comparison references: We consider three schemes as comparison references for our algorithm. The first one is a *zero-distortion* scheme that recommends to users the R items that are inferred to be most relevant to his/her preferences. This scheme essentially carries out only the first two of the three steps (initialization and content placement) described in section IV-A. It represents the current practice, where the recommender system acts completely independently from the caching system, bothering only to propose the most relevant content to each user.

The second comparison reference is an *unconstrained-distortion* scheme that is meant to represent the opposite extreme of the zero-distortion one. This scheme ranks the items in order of decreasing aggregate demand over all users, caches the top C of those in the cache and recommends to all users the same top R items. Thus, it recommends to users content that is cached, without accounting for the preference distortion constraint.

The third comparison is the Least Frequently Used (LFU) caching algorithm. The algorithm caches the items that have the maximum aggregate demand over all users, without recommending anything to them. The LFU maximizes the hit ratio at a single cache in the absence of recommendations.

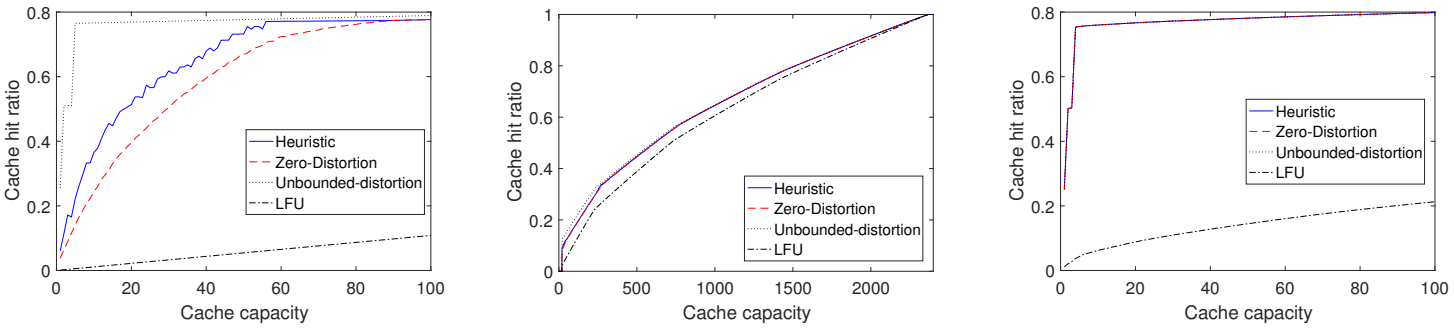
B. Simulation results

1) Caching-aware vs. unaware recommendations and the impact of recommendation weights: In the experiments of this subsection we vary the values of recommendation weights w_u^r in (5). We consider two different intervals where these weights take values in. In the first instance, $w_u^r \in (0.5, 1]$ (Figure 4a) and in the second $w_u^r \in (0, 0.2)$ (Figure 4b), $u \in \mathcal{U}$.

For small recommendation weights, the performance differentiation of the four schemes is negligible. As shown in Figure 4b, all schemes start with similarly small cache hit ratios at low cache capacities. As this capacity grows, the four schemes tend to get more similar and the achieved hit ratios tend almost linearly towards 1. On the other hand, for larger recommendation weight values, the three schemes that issue recommendations outperform LFU and achieve cache hit ratio values as high as 0.8 already for modest cache sizes, in the order of 5% of the catalogue. Figure 4a suggests that our heuristic improves over the zero-distortion scheme, while the unbounded distortion counterpart sets an upper bound for the achievable cache hit ratio.

When combined, figures 4a and 4b suggest, in accordance with intuition, that gains with the proposed scheme are achievable as far as users *do* assign importance to recommendations made to them.

2) The impact of heterogeneity in user content preferences: In this section, we want to explore how the heterogeneity of user content preferences affects the efficiency of our algorithm. To controllably vary 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 V-A; and a second user-agnostic probability distribution, p^{ext}, which is modeled



a. users with heterogeneous preferences ($w_u^e = 1$) $\forall u \in \mathcal{U}$ and $w_u^r \in (0.5, 1)$

b. users with heterogeneous preferences ($w_u^e = 1$) $\forall u \in \mathcal{U}$ and $w_u^r \in (0.01, 0.2)$

c. users with totally homogeneous preferences ($w_u^e = 0$).

Fig. 4. Cache hit ratio *vs.* user preferences distortion ratio: $|\mathcal{I}|$ = 1000, $|\mathcal{U}| = 150$, R=3, r_d=0.01

after a Zipf distribution, in line with the experimental evidence in [13] and [14]. The preference of user u for item i is then given by:

$$p_u^{pref}(i) = w_u^e \cdot p_u^{ego}(i) + (1 - w_u^e) \cdot p^{ext}(i) \qquad (13)$$

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

We experiment with two different values for w_e^u. In Fig. 4a-4b we had set $w_e^u = 1$, while in 4c, we let $w_e^u = 0$. Thus, in the two instances analyzed in section V-B1, users are totally heterogeneous, whereas in the third case the individual preferences are totally smoothed out. For $w_u^e = 0$ we notice that the three schemes that issue recommendations collapse to one. This is intuitively expected since, in this rather extreme case, the recommendations issued by the three schemes to the users are identical. Although Figures 4a, 4b and 4c present different patterns for the curves of the three recommendation-aware schemes, they all outperform the LFU scheme, thus reinforcing the positive impact recommendations can have on the cache hit ratio.

3) The impact of the number of recommended items: In Figure 4 we recommend R=3 items to every user. In the experiments of Figure 5a and 5b R=5 and R=10 items are recommended, respectively, to every user. First, we notice that increasing the number of recommended items decreases the achievable absolute hit ratio values but increases the relative gain of our heuristic over the zero-distortion scheme. This is a non-intuitive result, since fewer recommended items should be easier for the user to elaborate, leading to a bigger increase of hit ratio for the heuristic, in opposition to the zero-distortion scheme. Nevertheless, this behaviour could be expected from the definition of the recommendation window. Second, recommending more items to the users causes a right shift of the curves. This implies that a bigger cache capacity is needed to achieve the same cache hit ratio for given items and aggregate expected demand values.

4) The impact of the Distortion parameter, r_d: From Figure 5c we can notice that increase of r_d leads to increase of the hit ratio. This is in line with our intuition, because relaxing the distortion constraint we get closer to the performance of the unbounded distortion scheme. In Figure 5c, for small cache capacities, we can see an up to 80% increase of the hit ratio of the heuristic when compared to the zero-distortion scheme, introducing at most 10% of distortion in user preferences. In all these schemes, the LFU is significantly less effective than the three schemes issuing recommendations.

5) The impact of the catalogue size: Finally, as the total number of items in the system increases, the cache hit ratio decreases. Nevertheless, the heuristic and the zero-distortion reach the hit ratio of the unbounded-distortion scheme proportionally faster, *i.e.*, for smaller values of the ratio

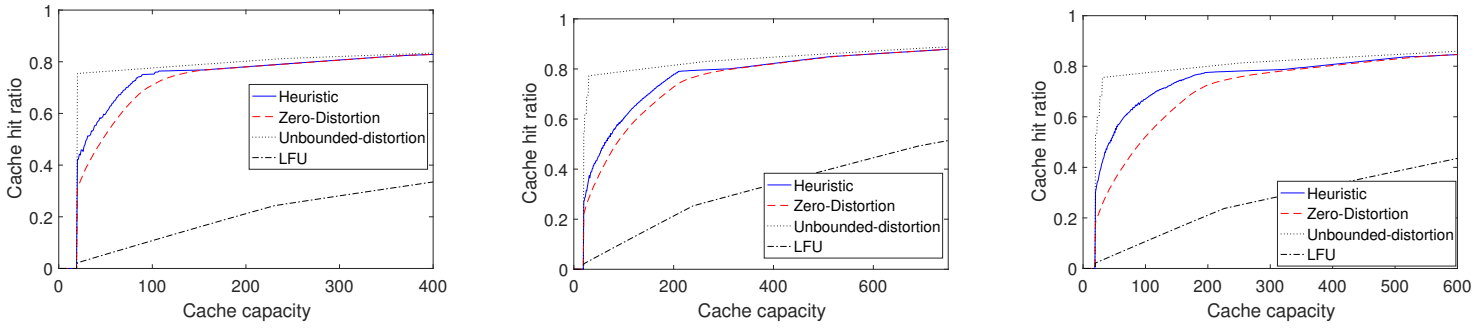
$$\alpha = \frac{\textit{cache capacity}}{\textit{catalogue size}} = \frac{C}{|\mathcal{I}|} \qquad (14)$$

when compared to a system that has fewer items. It is important that our heuristic needs 35% less cache space in order to reach the maximum achievable cache hit ratio. For this combination of parameters, we observe that the LFU scheme remains significantly worse than the other three schemes.

VI. Related Work

Content caching, and the often interchangeably used terms content placement and content replication, are classic networking themes that have been attracting research interest for at least twenty five years (*e.g.*, [15]). Much of this interest is due to the new spins that are given to the caching problem by technologies and trends that emerge in the course of time: Content Delivery Networks (*e.g.*, [16]), peer-to-peer systems (*e.g.*, [17]), IPTV (*e.g.*, [5]) and Content- and Information-Centric Networking architectures (*e.g.*, [18]).

More recently, caching experiences a new research thread in the context of mobile cellular networks and the small cell architectures in 4G and 5G networks [19]. One of the major concerns in these settings is how the user demand and content popularity can be accurately predicted. Besides the temporal



a. R=5, r_d= 0.01 b. R=10, r_d= 0.01 c. R=10, r_d= 0.1

Fig. 5. Cache hit ratio *vs.* number of recommended items and distortion parameter: $|\mathcal{I}|$ = 1000, $|\mathcal{U}| = 150$, $w_u^r \in (0.5, 1)$, $w_u^e = 1, \forall u \in \mathcal{U}$

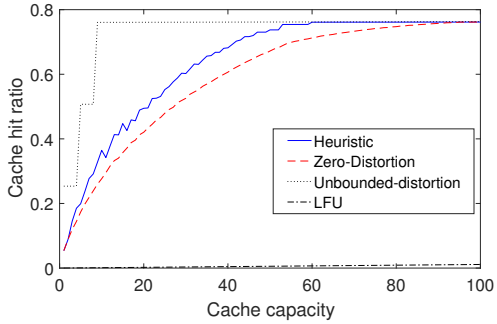


Fig. 6. $\mathcal{I}$=10000, R=10, r_d= 0.01, $w_u^e = 1$, and $w_u^r \in (0.5, 1)$

locality of content demand [6], such predictions should account for the user mobility and the small user populations that small cell caches present. To this end, a proactive caching approach for 5G wireless networks is proposed in [20]. The authors use data from a network operator and machine learning tools to predict content popularity and proactively fetch content to cell caches. In [21] a threshold-based policy is derived for caching content in the presence of small user populations. The policy consists in caching all content requested more times than a threshold and is shown to be asymptotically optimal regarding the hit rate. Their results further show that while a global cache learns faster, the local caches can be more accurate. 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 [4].

The impact of recommendations on user demand for content is studied in [3]. The authors report that the "Related Video" recommendation lists of Youtube is the main source of view for the majority of its content and that the position of an item in those lists plays a critical role for its popularity. Much sparser is the literature with respect to 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 [10], [11], [22]–[24]. The first three are concerned with video traffic, whereas the last two with generic content.

The authors in [11] 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 [22], 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 [23]. The aim is to determine how to replicate content within a CDN. Common to [11], [22], [23] 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 tradeoff 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 [10] and [24]. In the empirical study of [10], 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. Whereas in [24], the authors work with peer-to-peer (P2P) systems and propose heuristic recommendation algorithms accounting for both the content dissemination costs and user preferences. Compared to these two studies, in our work we focus on wireless networks and formalize the joint caching and recommendation problem under the assumption of *personalized* recommendations, *i.e.*, different items are recommended to each user. The results in section V suggest that such an approach could yield significant gains for the network performance and the users satisfaction without disrespecting their individual preferences.

VII. Conclusions and Discussion

Our work in this paper is motivated by the trend that wants content providers 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 since there is strong evidence that willingness to pay can be affected by online recommendations [25]. 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.

As a direct next step, we intend to evaluate the performance of our heuristic using real datasets[5] of rated content for inferring user preferences. An interesting direction for further work is to try to fully integrate such an heretic approach to recommendations with other resource management and traffic engineering functions in cellular networks to achieve more composite performance objectives such as load balancing across the radio network cells.

VIII. Acknowledgments

The authors acknowledge the support of Marie Curie IRSES grant PIRSES-GA-2010-269132 AGILENET.

References

[1] T. H. Sarkissian, "The Business Case for Caching in 4G LTE Networks," http://www.wireless2020.com/media/whitepapers.html, 2012, [Online; accessed 29-July-2016].

[2] C. A. Gomez-Uribe and N. Hunt, "The netflix recommender system: Algorithms, business value, and innovation," *ACM Trans. Management Inf. Syst.*, vol. 6, no. 4, pp. 13:1–13:19, 2016.

[3] R. Zhou, S. Khemmarat, and L. Gao, "The impact of youtube recommendation system on video views," in *Proc. 10th ACM SIGCOMM Internet Measurement Conference (IMC)*, Melbourne, Australia, November 2010, pp. 404–410.

[4] K. Poularakis, G. Iosifidis, A. Argyriou, and L. Tassiulas, "Video delivery over heterogeneous cellular networks: Optimizing cost and performance," in *Proc. IEEE INFOCOM '16*, Toronto, Canada, April 2014, pp. 1078–1086.

[5]https://grouplens.org/datasets/movielens/

[5] D. D. Vleeschauwer and K. Laevens, "Performance of caching algorithms for IPTV on-demand services," *IEEE Trans. on Broadcasting*, vol. 55, no. 2, pp. 491–501, 2009.

[6] 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," *Computer Communication Review*, vol. 43, no. 5, pp. 5–12, 2013.

[7] M. Cha, H. Kwak, P. Rodriguez, Y. Ahn, and S. B. Moon, "I tube, you tube, everybody tubes: Analyzing the world's largest user generated content video system," in *Proc. 7th ACM SIGCOMM Internet Measurement Conference (IMC)*, San Diego, California, October 2007, pp. 1–14.

[8] J. Leskovec, A. Rajaraman, and J. D. Ullman, *Mining of Massive Datasets, 2nd Ed*. Cambridge University Press, 2014.

[9] J. Vegelius, S. Janson, and F. Johansson, "Measures of similarity between distributions," *Quality and Quantity*, vol. 20, no. 4, pp. 437–441, 1986.

[10] 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. 48:1–48:20, Jun. 2015.

[11] M. Verhoeyen, J. D. Vriendt, and D. D. Vleeschauwer, "Optimizing for video storage networking with recommender systems," *Bell Labs Technical Journal*, vol. 16, no. 4, pp. 97–113, 2012.

[12] V. V. Vazirani, *Approximation algorithms*. Springer, 2001.

[13] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and zipf-like distributions: Evidence and implications," in *Proc. IEEE INFOCOM '99*, New York, NY, USA, March 1999, pp. 126–134.

[14] P. Gill, M. F. Arlitt, Z. Li, and A. Mahanti, "Youtube traffic characterization: A view from the edge," in *Proc. 7th ACM SIGCOMM Internet Measurement Conference (IMC)*, San Diego, California, USA, October 2007, pp. 15–28.

[15] A. Leff, J. L. Wolf, and P. S. Yu, "Replication algorithms in a remote caching architecture," *IEEE Trans. Parallel Distrib. Syst.*, vol. 4, no. 11, pp. 1185–1204, 1993.

[16] J. Kangasharju, J. Roberts, and K. W. Ross, "Object replication strategies in content distribution networks," *Computer Communications*, vol. 25, no. 4, pp. 376–383, Mar. 2002.

[17] S. Iyer, A. I. T. Rowstron, and P. Druschel, "Squirrel: A decentralized peer-to-peer web cache," in *Proc. 21st ACM PODC*, Monterey, CA, USA, July 2002, pp. 213–222.

[18] M. Zhang, H. Luo, and H. Zhang, "A survey of caching mechanisms in information-centric networking," *IEEE Communications Surveys and Tutorials*, vol. 17, no. 3, pp. 1473–1499, 2015.

[19] 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 '12*, Orlando, FL, USA, March 2012, pp. 1107–1115.

[20] E. Bastug, M. Bennis, E. Zeydan, M. A. Kader, A. Karatepe, A. S. Er, and M. Debbah, "Big data meets telcos: A proactive caching perspective," *CoRR*, vol. abs/1602.06215, 2016.

[21] 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 '16*, San Francisco, CA, USA, April 2016, pp. 1–9.

[22] 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 '16*, San Francisco, CA, USA, April 2016, pp. 1–9.

[23] M. A. Kâafar, S. Berkovsky, and B. Donnet, "On the potential of recommendation technologies for efficient content delivery networks," *SIGCOMM Computer Communication Review*, vol. 43, no. 3, pp. 74–77, 2013.

[24] D. Munaro, C. Delgado, and D. S. Menasché, "Content recommendation and service costs in swarming systems," in *Proc. IEEE International Conference on Communications ICC*, London, UK, June 2015, pp. 5878–5883.

[25] G. Adomavicius, J. Bockstedt, S. P. Curley, and J. Zhang, "Effects of online recommendations on consumers' willingness to pay," in *Proc. 2nd Workshop on Human Decision Making in Recommender Systems*, Dublin, Ireland, September 2012, pp. 40–45.