Chapter 21

# On the joint optimization of content caching and recommendations

*Livia Elena Chatzieleftheriou[1], Merkourios Karaliopoulos[1], and Iordanis Koutsopoulos[1]*

Content caching has been experiencing revived interest within the context of current and next generation wireless networks. It brings content closer to users, decreasing the *aggregate* delivery costs and service delays for network providers. Recommender systems have become integral components of content provision sites. They offer personalized recommendations, increasing the *individual* user satisfaction and engagement with the content provider's platform. Traditionally, caching and recommendation decisions are taken separately. However, there is a recent persistent trend where both network and content providers tend to deploy their own content delivery solutions. In light of this, we explore how the phenomenally conflicting objectives of content caching and recommendation can be jointly addressed.

In this chapter we approach recommender systems as network traffic engineering tools that actively shape content demand to serve both user- and network-centric performance objectives. We introduce a model that captures the coupling between caching decisions and issued recommendations. Based on experimental evidence, we describe the impact of recommendations on user content requests and present a systematic way of engineering the user 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 describe the "recommendation window" as a way to controllably bound the distortion that recommendations will undergo and discuss specific properties that the recommender system should have in order to ensure that the "Quality of Recommendations (QoR)" is kept high by the issued recommendations. We formulate a joint optimization problem for the content to cache and recommend to each user, aiming to maximize the cache hit ratio. The preference distortion tolerance is embedded as a constraint to this problem formulation and marks an equally important dimension for assessing possible solutions. We prove that the problem lacks those properties that would guarantee its approximability, and devise a low-complexity practical

[1]Athens University of Economics and Business, Athens, Greece

algorithm that solves it efficiently. 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. We thoroughly evaluate it, by establishing its main properties and analytical performance bounds. We conduct an extensive sensitivity analysis over various system parameters, both analytically and through simulations with real and synthetic dataset. Next we extend our model and approach for a system in which users can access content from multiple caches, and analyse the intermediate joint caching and user association problem. We then discuss open directions and provide a detailed taxonomy of the related work. Due to space limitations, we do not provide proofs for our results. However, the interested reader may refer to [1–3] for more propositions and their detailed proofs.

## 21.1 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. 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, the technological advances have totally reshaped the nature of multimedia services. The plethora of content made available to users creates them discomfort: the users either find quickly something of interest, or they abandon the service [4]. Nevertheless, the great variety of immediately available content should turn to users' advantage and recommender systems play a key role to this end.

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 improves the users' satisfaction and boosts their experience, increasing the number of content downloads and keeping them engaged with the platform. For instance, the Related Video recommendations generate about 30% of the overall views on YouTube [5], whereas the recommender system used by Netflix is considered responsible for about 80% of the hours streamed at Netflix. Moreover, the combined effect of personalization and recommendations saves Netflix more than $1B per year [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 servers storing content within other networks through either their own or third-party Content Delivery Networks (CDNs). In mobile cellular networks, in particular, these servers tend to be placed at their egress nodes rather than at their edge.

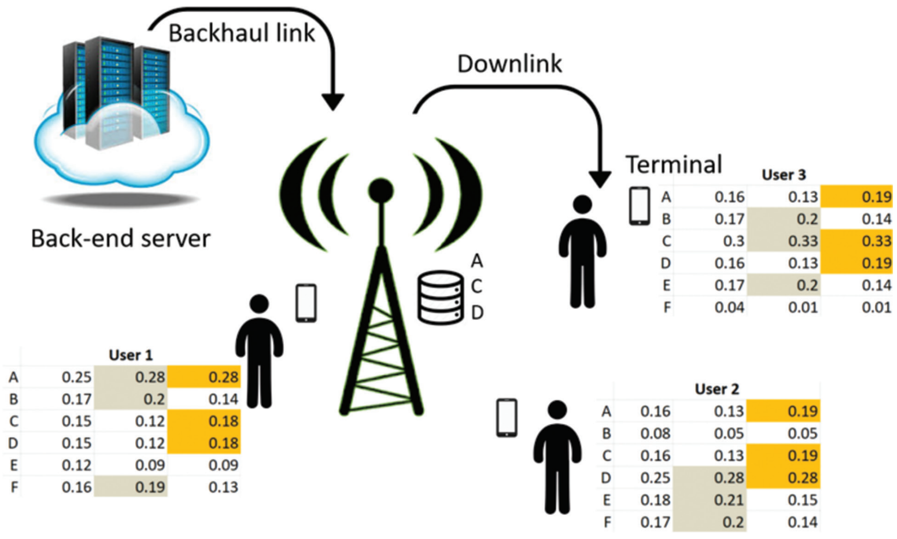
However, players with originally distinct roles in the business value chain, such as access network operators and content providers, tend for some time now to deploy their own content delivery solutions, albeit for different reasons. Content providers seek to acquire better control of the network access infrastructure so as to improve the QoE delivered to their subscribers. Netflix Open Connect[a] and Google Global Cache[b] are two widely known examples of CDN solutions owned by content providers. Access network operators, on the other hand, deploy CDNs (telco CDNs) 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 and can gain an advantage over their competitors in the pursuit of customers.

In the case of wireless networks, these trends motivate novel content provisioning scenarios, whereby the coordination of (at least) three different mechanisms can enhance 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, *routing user content 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:** Recommendations appear, thus, to have interesting repercussions for the design of caching algorithms. Consider, e.g., 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 topmost 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* (Figure 21.1).

The possibility to 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 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,

[a]openconnect.netflix.com/en/
[b]peering.google.com/



| | | User 3 | |
|---|---|---|---|
| A | 0.16 | 0.13 | 0.19 |
| B | 0.17 | 0.2 | 0.14 |
| C | 0.3 | 0.33 | 0.33 |
| D | 0.16 | 0.13 | 0.19 |
| E | 0.17 | 0.2 | 0.14 |
| F | 0.04 | 0.01 | 0.01 |

| | | User 1 | |
|---|---|---|---|
| A | 0.25 | 0.28 | 0.28 |
| B | 0.17 | 0.2 | 0.14 |
| C | 0.15 | 0.12 | 0.18 |
| D | 0.15 | 0.12 | 0.18 |
| E | 0.12 | 0.09 | 0.09 |
| F | 0.16 | 0.19 | 0.13 |

| | | User 2 | |
|---|---|---|---|
| A | 0.16 | 0.13 | 0.19 |
| B | 0.08 | 0.05 | 0.05 |
| C | 0.16 | 0.13 | 0.19 |
| D | 0.25 | 0.28 | 0.28 |
| E | 0.18 | 0.21 | 0.15 |
| F | 0.17 | 0.2 | 0.14 |

*Figure 21.1* *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, 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 the 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.*

ranks first in the user preferences but is of no interest to other users in the cells the user can associate with.

## 21.2 System model

We consider a wireless network with a set of small cells and a macrocell that work in conjunction, forming a two-layer heterogeneous network. In current communication networks, macro- and micro-cells are outdoor cells with a typical radius up to a few kilometer and a few hundred meters, respectively.

*Table 21.1 Notation table*

| **Symbol** | **Context** | **Symbol** | **Context** |
|---|---|---|---|
| $S_c$ | Storage capacity of cache $c$ | $M$ | Number of thematic categories |
| $L_i$ | Normalized length of item $i$ | $a_{ui}$ | Similarity of user $u$ and item $i$ |
| $\mathbf{f}^i$ | Feature vector values of content item $i$ over the $M$ thematic categories | $\mathbf{f}_u$ | Feature vector value of user $u$ over the $M$ thematic categories |
| $p_u^{pref}$ | Inherent content preference distribution of user $u$ | $p_u^{rec}$ | Probability distribution due to Recommendation |
| $p_u^{req}$ | Content item request probability distribution of user $u$ (recommended items) | $\widetilde{p}_u^{req}$ | 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 $S$ | $\mathcal{P}$ | Cache placement |
| $\mathcal{RC}_u^{in}$ | Provisional set of recommended items to $u$ | $\mathcal{RC}_u^{f}$ | Final set of recommended items to $u$ |
| $ZD$ | Zero-distortion scheme | $UD$ | Unbounded-distortion scheme |
| $CawR$ | Caching-aware recommendations Scheme | | |

### 21.2.1 Caches, content, users

Our model involves a set of caches $\mathcal{C}$, a catalog of content items (video clips), $\mathcal{I}$, and a set of users, $\mathcal{U}$ (Figure 21.1) (Table 21.1).

**Caches:** Caches are co-located with wireless network microcells. Each cache $c \in \mathcal{C}$ has limited storage capacity, $S_c$, 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 $\mathcal{P}_c$. An additional cache is installed on the macrocell or the cloud, operating as a backend server. This "cache" is assumed to have enough capacity to store copies of the entire catalog.

**Content:** Content items are relevant to one or more of a set of $\mathcal{U}$ 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. For example, "soccer" may form a distinct category, but it may also be further split into "English/French/Spanish soccer."

The $M$ thematic categories serve as a feature set that describes items. Namely, each item $i \in \mathcal{I}$ is represented by a feature vector $\mathbf{f}^i$, 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 $\sum_{j=1}^{M} f^i(j) = 1,\ \forall i \in \mathcal{I}$.

The content catalog includes content items of different sizes, e.g.,entire movies and their trailers. We denote by $L_i \geq 1$ the normalized size of content item $i$, i.e., the ratio of its actual size over the smallest item size of the catalog. Indicatively, consider the simple case where files come with only one resolution quality. Then, $l_{min} = 100s$ and $l_{max} = 100min = 6000s$ are typical values for the trailers' and movies' duration so that $L_i = \frac{l_{max}}{l_{min}}$ is in the order of 60[c]. Replicas of each content item may be stored in any subset of the small cell caches, besides the backend cache, depending on the actual caching decisions.

**Example 1:** Let the feature set be $\mathcal{A}$ = {Biography, Crime, Drama, Comedy, Romance}, and the set of contents $\mathcal{I}$ = {The Godfather, The Irishman, Amélie}. The "Godfather" falls under the Crime and Drama themes, the "Irishman" below Biography, Crime and Drama, and "Amélie" falls under the Comedy and Romance themes. Then, the feature vectors are $\mathbf{f}^{Godfather} = [0, 0.5, 0.5, 0, 0]$, $\mathbf{f}^{Irishman} = [0.33, 0.33, 0.34, 0, 0]$ and $\mathbf{f}^{Amélie} = [0, 0, 0, 0.5, 0.5]$. Alternatively, the feature vectors could be binary, so that a feature assumes the value 1 if a tag is relevant for the movie, and 0, otherwise. For example, the binary feature vector for "Amélie" would be $\mathbf{f}^{Amélie} = [0, 0, 0, 1, 1]$.

**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. The user might access different content items through different caches, each time dynamically changing her association depending on the requested content. In section 21.3, we assume that users do not change their association point in the network dynamically in response to content requests, while in section 21.7 we consider user association as an additional (indirect) road for nudging users' preferences on which we have control.

Users are described by similar feature vectors $\mathbf{f}_u \in [0, 1]^M$ as the content items. 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, *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 such as possible ratings of content items, to infer these vectors. We elaborate on this in section 21.6.1.

**Example 2:** Consider the feature set of Example 1 and the set of users $\mathcal{U} = \{u_1, u_2, u_3\}$. A way to quantify the relevance of features for the users is to consider the percentage of tags in the movies they rated. For example: $\mathbf{f}_{u_1} = [0.6, 0.4, 0, 0, 0]$, $\mathbf{f}_{u_2} = [0.2, 0.2, 0.2, 0.2, 0.2]$ and $\mathbf{f}_{u_3} = [0, 0, 0, 0, 1]$.

## *21.2.2 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

[c]http://www.stephenfollows.com/long-average-movie-trailer, last accessed on Feb 20, 2019.

out [6, 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 [8]. 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 $\mathbf{f}_u$ 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 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 $\mathbf{f}^i$. We use for this purpose the cosine similarity index, $a_{ui}$, of vectors $\mathbf{f}_u$ and $\mathbf{f}^i$. Note that other similarity measures could also be applicable [9]. Then, the content preference distribution $p_u^{pref}$ over all items is given by:

$$p_u^{pref}(i) = \frac{a_{ui}}{\sum_{i \in J} a_{ui}}, \text{ where } a_{ui} = \frac{\sum_{j=1}^{M} \mathbf{f}_u(j) \cdot \mathbf{f}^i(j)}{\sqrt{\sum_{j=1}^{M} (\mathbf{f}_u(j))^2} \sqrt{\sum_{j=1}^{M} (\mathbf{f}^i(j))^2}}. \tag{21.1}$$

**Example 3:** Consider the item-feature vector of Example 1 and the user-feature vector of Example 2. Then, the users' inherent preferences for the movies "The Godfather," "The Irishman" and "Amélie" are: $p_1^{pref}$ = [ 0.38, 0.62, 0], $p_2^{pref}$ = [ 0.33, 0.33, 0.33], $i$ = [ 0, 0, 1].

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

## 21.2.3 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 [6], 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 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 [5]. 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 21.2.1.** 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 the item's $i$ position in the recommendation list, $rnk_u(i) \in [1, R]$, and the number of recommended items,

In section 21.6.1 we consider specific instances of $R$ that satisfy this assumption. For a more detailed discussion please refer to section 6.1 in [1].

**Assumption 21.2.2.** The content request distribution is a convex combination of the two distributions, $p_u^{pref}$ and $p_u^{rec}$and is given by

$$p_u^{req}(i) = w_u^r \cdot p_u^{rec}(i) + (1 - w_u^r) \cdot p_u^{pref}(i) \tag{21.2}$$

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

$$\widetilde{p}_u^{req}(i) = (1 - w_u^r) \cdot p_u^{pref}(i) \tag{21.3}$$

for each one of the $(|\mathcal{I}| - R)$ items not recommended to user $u$. The user-specific recommendation weights $w_u^r$ in (21.2) and (21.3) express the importance user $u$ attaches to recommendations in the r-th position of their recommendation list.

Equations (21.2) and (21.3) 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 Figure 21.2).

### *21.2.4 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 Figure 21.2 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 Figure 21.2.

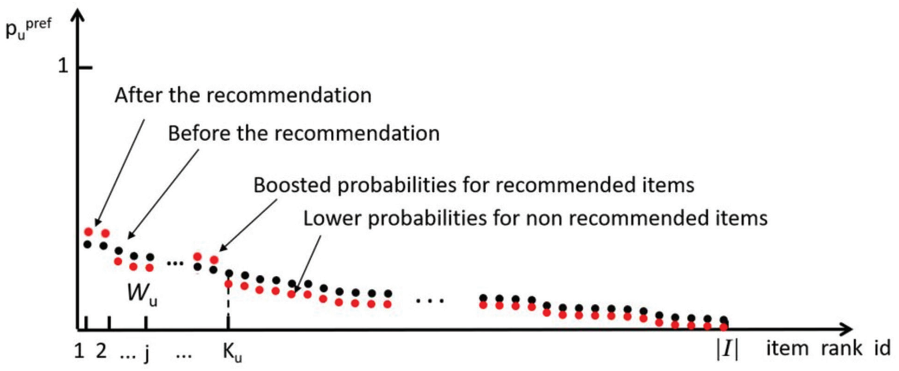


*Figure 21.2 Impact of recommendations on user requests. The content items are ranked in decreasing order of user preference and only items within the recommendation window are recommended. A priori content preference distribution for arbitrary user u, its recommendation window $\mathcal{W}_u$ of size $K_u$ (black), and the resulting content request probability after recommendations (red).*

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 ensures the QoR, preserving two properties addressing the ethical concerns described in section 21.1 regarding the possible 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.

Although the rank of recommended items in the original user content preferences is preserved, it could happen that the absolute rank of file popularities may change after the recommendations. For example, an item that is high in the inherent user's rank and that is not recommended may move lower in her rank than an item that was originally bellow it but has been recommended.

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. Figure 21.2), instead of the items in top-R positions $\{1, 2, \ldots R\}$. We define the worst-case *user preference distortion* measure, $\Delta_u$ to be

$$\Delta_u(K_u, R) = 1 - \frac{\sum\limits_{\substack{j:rnk_u(j)\in \\ [K_u-R+1,K_u]}} p_u^{pref}(j)}{\sum\limits_{j:rnk_u(j)\in[1,R]} p_u^{pref}(j)}. \tag{21.4}$$

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

The denominator in (21.4) equals 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 (21.4).

## 21.3 The joint caching and recommendations problem

In this section we assume that the user associations with network (small) cells depend on the quality of radio signals and the load of the radio network, without taking into account the availability of content at the co-located caches. A direct consequence of this assumption, which is in line with user association practices in current mobile cellular networks, is that the caching decisions can be made independently in each cache-enabled small cell. Any given cache placement $\{P_c : c \in \mathcal{C}\}$ may satisfy a portion of the total demand in the cell as measured by the cache hit ratio[d]

$$H = \frac{\sum\limits_{u\in\mathcal{U}} \sum\limits_{i\in\mathcal{P}} p_u^{req}(i)}{\sum\limits_{u\in\mathcal{U}} \sum\limits_{i\in\mathcal{I}} p_u^{req}(i)}. \tag{21.5}$$

On the user side, the requirement is to maximize the portion of her content 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 back-end server, the utilization of backhaul links is lower. In other words, the maximization of the cache hit ratio serves both user- and network-oriented objectives.

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 $i$, otherwise; and $x_{ui} = 1$ when item

[d]Since in this section the caching decisions are made independently for each cache, we drop the index $c$ from subsequent references to cache placements, i.e., we write $\mathcal{P}$ instead of $\mathcal{P}_c$.

$y_i, x_{ui} \in \{0,1\}\ \ u \in \mathcal{U}, i \in \mathcal{W}_u$. is recommended to user $u$ and $x_{ui} = 0$ when it is not. The objective of the joint caching and recommendation decisions is then to find the cache and recommendation policy that maximizes the hit ratio, as defined below:

$$\max_{\mathbf{y},\mathbf{x}} \sum_{u\in\mathcal{U}} \sum_{i\in\mathcal{W}_u} y_i(x_{ui}p_u^{req}(i) + (1 - x_{ui})\widetilde{p}_u^{req}(i)) \tag{21.6}$$

$$s.t. \sum_{i\in\mathcal{I}} y_i L_i \leq S \tag{21.7}$$

$$\sum_{i\in\mathcal{W}_u} x_{ui} = R, \quad \forall u \in \mathcal{U} \tag{21.8}$$

$$y_i, x_{ui} \in \{0,1\} u \in \mathcal{U}, i \in \mathcal{W}_u. \tag{21.9}$$

In (21.6) and (21.8), $\mathcal{W}_u$ denotes items within the recommendation window of user $u$. The cardinality of this set is

$$K_u = \max\{k|\Delta_u(k,R) \leq r_d(u)\}, \tag{21.10}$$

where $r_d(u) \in [0,1)$ denotes the user-specific *preference distortion tolerance*, an upper bound on user preference distortion in (21.4) 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 (21.7) reflects the cache storage capacity constraint, whereas equalities (21.8) ensure that exactly $R$ items are recommended to every user.

## *21.3.1 Problem complexity and approximability properties*

We refer to the problem (21.6)–(21.9) as the Joint Caching and Recommendations Problem (JCRP). In [1] we prove that:

**Proposition 1.** The Joint Caching and Recommendations Problem is NP-complete.

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 $S$, $Y$ any two subsets of $S$ satisfying $X \subseteq Y \subseteq S$. A set function $f: 2^S \rightarrow R$ is called *monotone* if $f(X) \leq f(Y), \forall X, Y \subseteq S$. Moreover, $f$ is called *submodular*, if for any element $e \in S \setminus Y$, it holds that $f_X(e) \geq f_Y(e)$, 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 $f_X(e) \geq f_Y(e)$,, it holds that $f_X(e) \leq f_Y(e)$. In [1] we prove that:

**Proposition 2.** The objective function of *JCRP* is non monotone.

**Proposition 3.** The objective function of *JCRP* is neither submodular nor supermodular.

Hence, more general techniques that yield approximability guarantees for optimization problems with (monotone) submodular objective functions, and have been often applied to caching problems, 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 (section 21.4.2), it is computationally simple (section 21.4.1) and exhibits excellent performance (section 21.6).

## 21.4 An algorithm for the joint caching and recommendation problem

### *21.4.1 Description of the algorithm*

Our Caching-aware Recommendations (CawR) algorithm (see Algorithm 1) proceeds in three steps, as shown in Figure 21.3.

In the first step, 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 the *content placement* step, where we determine which content should be cached. To this end, we compute the content request probabilities according to (21.2), (21.3). All content items are assigned utilities that equal the aggregate request probability they attract:

$$v(i) = \sum_{u \in \mathcal{U}} p_u^{req}(i) \quad i \in \mathcal{I}. \tag{21.11}$$

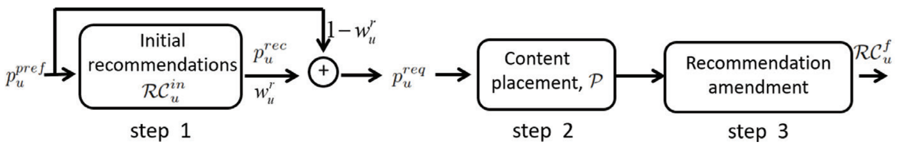


*Figure 21.3 Schematic outline of the three-step algorithm for the JCR problem*

The optimal placement is then an instance of the 0-1 Knapsack Problem (KP). We use the Dynamic Programming (DP) Fully Polynomial Time Approximation Scheme (FPTAS) algorithm in ([10], §8.2) to obtain an $(1-\varepsilon)$, $\varepsilon > 0$ 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 her recommendation window from (21.10). 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; $F = \min\{R - F_1, F_2\}$, $F_2 = |(\mathcal{W}_u \setminus RC_u^{in}) \bigcap \mathcal{P}|$, most preferred cached items appearing in the recommendation window of $u$ (but not in the recommendation set derived in the first 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 first 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. However, in [1] we prove that this algorithm terminates in a single round:

**Proposition 4.** CawR terminates after a single execution of the recommendation amendment step.

Hence, the algorithm essentially determines 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 Figure 21.1, CawR first sets the provisional recommendation list $RC_u^{in}$ to the $W_u$ 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

**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:
 1: Set the initial recommendation list $\mathcal{RC}_u^{in}$ to the $R$ most preferred items of the user and $p_{rec}$
    to $f(rnk_u(i), R)$ (ref. Assumption 21.2.1)
 2: for every user $u \in \mathcal{U}$ and item $i \in \mathcal{I}$ do
 3:     Compute the content request probabilities $p_u^{req}(i)$ from (21.2) and (21.3).
 4: end for
    Step 2:
 5: for every item $i \in \mathcal{I}$ do
 6:     Compute its utility from (21.11).
 7: end for
 8: Use the DP $(1-\varepsilon)$-approximation algorithm, $\varepsilon > 0$, to solve the 0-1 KSP and derive the
    content placement $\mathcal{P}$.
    Step 3:
 9: for every user $u$ do
10:     Add items in $\mathcal{RC}_u^{in} \bigcap \mathcal{P}$ to the final recommendation list $\mathcal{RC}_u^f$ for $u$.
11:     if $\mathcal{RC}_u^f$ is not full then
12:         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.
13:     end if
14:     if $\mathcal{RC}_u^f$ is still not full then
15:         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.
16:     end if
17: end for
```

the utility of each item from (21.11) 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 Figure 21.1. The cache hit ratio without recommendations is 0.58. When recommendations are issued for the top-3 items in users' preferences (3rd column), the cache hit ratio drops to 0.55, because non-cached items are recommended. On the contrary, 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.

### 21.4.2 Complexity of the CawR algorithm

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 $O(|\mathcal{I}| + |\mathcal{I}| \cdot |\mathcal{P}|) = O(|\mathcal{I}|^2)$ since the cache capacity is upper bounded by the total catalog 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 $O(|\mathcal{U}| \cdot \max_u(|\mathcal{W}_u|) \cdot |\mathcal{P}|)$. Since the size of the recommendation window is naturally bounded by the catalog size, the total computational complexity of CawR is bounded by$O(|\mathcal{U}| \cdot |\mathcal{I}|^2)$.

In the two sections that follow, we draw on both analysis (21.5) and simulations with real and synthetic datasets (21.6) to gain further insight to the properties of the proposed algorithm.

## 21.5 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, $\{p_u^{pref}\}$.

### *21.5.1 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 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. In [1] we prove that:

**Proposition 5.** When recommendations follow assumptions 21.2.1 and 21.2.2 and the user preference distortion tolerances are relaxed, $W_u \equiv \mathcal{I}$, $\forall u \in \mathcal{U}$ in (21.6) and (21.8), 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 6.** The expected cache hit ratios achieved by the three schemes that issue recommendations satisfy:

$$H_{ZD} \leq H_{CawR} \leq H_{UD}. \tag{21.12}$$

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 Prop. 5, $H_{UD}$ 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

$$H_{CawR} \leq H_{OPT} \leq H_{UD}. \tag{21.13}$$

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 (21.13) with numerical simulations in section 21.6.

### *21.5.2 Sensitivity analysis to parameters of CawR*

#### 21.5.2.1 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 (21.4), 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 $H_{CawR}$.

**Proposition 7.** $H_{CawR}$ 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 $H_{CawR}$ of with the $\{r_d(u)\}$ parameters [1].

#### 21.5.2.2 Sensitivity of $H_{CawR}$ to the individual user content preferences

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

**Proposition 8.** 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. 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 $K_u = R$.

## 21.6 Experimental evaluation of CawR

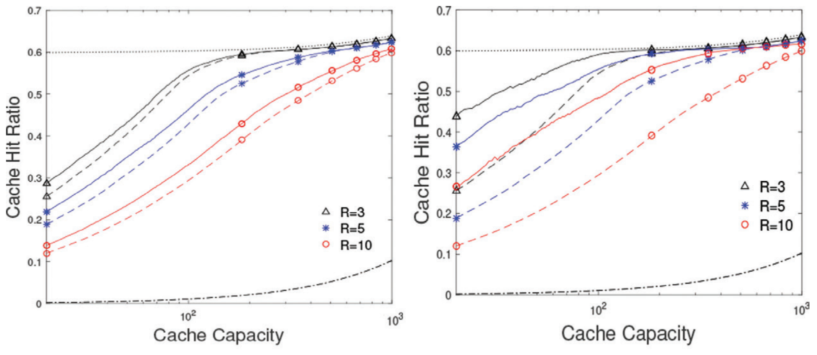
### *21.6.1 Datasets and default parameter settings*

We use both synthetic and real datasets [11] 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 21.2). 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 21.5. 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.

**MovieLens dataset [11]:** We analyze different samples of 700 users and 10.000 items of the catalog. Each of these movies is described by $M = 19$ thematic tags, i.e. the feature set contains 19 themes. In populating our model, we draw on the fact that a user *did* rate a specific item, rather than the actual rating she assigned to it. More specifically, the item tags are directly used to generate the item-feature vectors, as described in Example 1. 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 in Example 2.

**Synthetic datasets:** In this set of experiments, the elements of the two vectors are populated with values drawn from random probability distributions, by default from 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 distribution $j \in [1, M]$, with default $L_{max} = 4$. Our baseline assumption is that recommendations provide all $|\mathcal{U}|$ items in the list with



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

*Figure 21.4* Experiments with MovieLens traces. *Cache hit ratio, H, 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.*

an equal boost $p_u^{rec}(i) = 1/R$, 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, $R$, are aligned with experimental evidence in [5], 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 $U(0.5, 0.7)$.

## *21.6.2 Trace-Driven simulations*

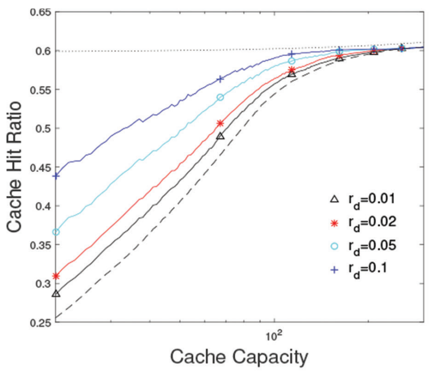
Figure 21.4 and 21.5 validate the performance bounds we found analytically for the cache hit ratio under CawR in section 21.5. The extent to which these bounds are tight depends on the number of issued recommendations per user, $R$, and the preference distortion tolerance parameter, $r_d$.

**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 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 $R = 10$

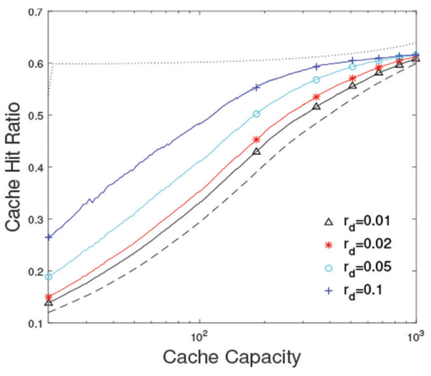
and $r_d$=0.01, ZD performs up to 16.6% worse than CawR in terms of cache hit ratio, especially for really small instances of cache capacity (Figure 21.4a). For $r_d$=0.1, this gap grows to 121% (Figure 21.4b). At the same time, CawR reaches the 96% and 97% of the performance of the UD scheme, for $r_d$=0.01 and $r_d$=0.1 respectively, and realistic cache capacities (Figure 21.4a-21.4b). 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% more cache capacity than CawR to reach the maximum achievable cache hit ratio when $R$=10 (Figure 21.4a). For $r_d$=0.1, this value climbs to 147% (Figure 21.4b). In all cases, the LFU scheme is significantly outperformed by schemes that use recommendations.

**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 (Figure 21.4b), 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 Figure 21.5(a-b), but also in Figure 21.4(a-b), the CawR performance increases monotonically with $r_d$ moving away from its lower bound (ZD scheme) towards its upper bound (UD scheme). Indicatively, for $r_d$ = 0.01 and $r_d$ = 0.1, we evidence with CawR cache hit ratios up to 71% higher than those under the ZD scheme at very small caches (Figure 21.5a). The respective performance gain increases to 121% for $R$ = 10 (Figure 21.5b).



(a) $R$ = 3



(b) $R$ = 10

*Figure 21.5* Experiments with MovieLens traces. *Cache hit ratio, H, 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.*

Regarding cache capacity requirements, even for distortion tolerance values smaller than $r_d = 0.01$ and $R = 3$, CawR needs up to 35% less storage capacity to converge to the upper bound confronted to the ZD scheme. This gain escalates up to 65% when $r_d = 0.1$ (Figure 21.5a).

**The impact of cache capacity, catalog 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 catalog size, cache size, and population of users. They report the positive impact of ratio $\alpha = \frac{\text{cache capacity}}{\text{catalog size}} = \frac{C}{|\mathcal{I}|}$ 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. Related figures can be found in [1].

## 21.6.3 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.

**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 (21.3), namely $w_u^r \in [0, 0.2]$ and $R$. The achieved cache hit ratio for each scenario is shown in Figure 21.6.

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% of the total catalog size, CawR scores similarly to the ZD scheme, needing up to 35% less cache capacity. Compared to the UD scheme, CawR reaches the 89% of the performance of the UD scheme, performing always more than 14% 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% higher than under ZD, for modest cache capacities in the order of 2% of the catalog size. In terms of cache capacity, CawR equals ZD needing up to 32% less cache capacity even for capacities lower than the 2% of the total catalog size. Compared to the UD scheme, for high recommendation weights, CawR reaches the 97% of the performance of the UD scheme, performing always more than 6% better than the ZD scheme. Overall, Figure 21.6 indicates that the more users assign importance to recommendations made to them, the higher the advantage of our algorithm over the “honest” recommendation scheme.

**Heterogeneity in user content preferences:** Due to space limitations, this section is briefly presented. For a more detailed presentation of the results regarding the heterogeneity in user content preferences, the interested reader may see [1].

The heterogeneity in user content preferences can manifest itself in multiple ways. To control 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 21.2.2; and a second user-agnostic

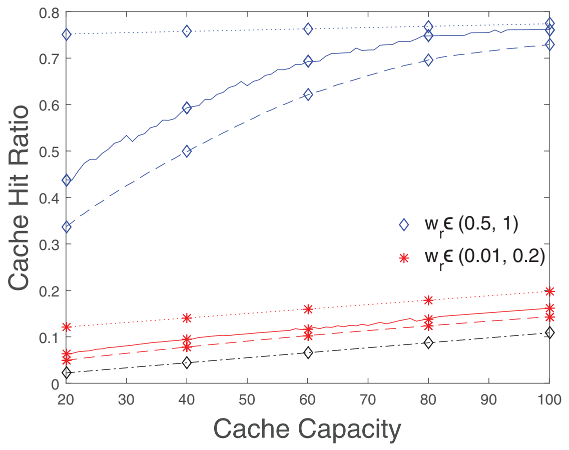


*Figure 21.6* Experiments with Synthetic datasets. *Cache hit ratio capacity as a function of user recommendation weights, R=10,* $r_d$*=0.01,* $|\mathcal{I}| = 1.000$ *items. Dotted lines correspond to the UD scheme, dashed lines to ZD, dash-dot lines to LFU, and solid lines to the CawR.*

probability distribution, $w_u^e = 0$, which is modeled after a Zipf distribution, in line with experimental evidence [12].

We have experimented with different values for $w_e^u$, concluding that 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 ($w_u^e = 0 \ \forall u \in \mathcal{U}$). Moreover, we experimented with the impact of the content preference distribution. Namely, we let various Zipf and uniform distributions serve as the user content preference distributions, permuting the order of items for each one of them. We 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. Our results are inline with the previously proposed Propositions.

## 21.7 User associations as additional control parameter: extending the model

Until now, our analysis has assumed that the associations of users to small cells are driven by physical proximity and radio signal quality considerations, in line with current practices in mobile cellular networks. As a result, the caching and recommendation decisions are taken independently in each cell, as detailed in section 21.4.

However, both in current generation cellular systems and, even more, in the ultra dense radio network architectures envisaged for 5G and beyond systems [13], users have the choice to associate with multiple different cells. Controlling then which users are associated with each cell and which content is stored at each cell-colocated cache may offer additional degrees of freedom when trying to satisfy locally the user demand for content and alleviate the load on the backhaul links. While recommendations shape the individual user demand, the user associations intervene on its spatial distribution across the radio network.

In principle, the two mechanisms may be exercised over different time scales. For instance, with online caches the cache placement may be reiterated upon each and every content item request by a user. Alternatively, the cached content may be determined periodically (e.g., every day or half a day) out of estimates for the local content demand and remain fixed for the respective interval of time. On the other hand, a new user association to a small cell may let its cache intact or trigger changes that budget for her individual content demand distribution.

In general, the more aggressively the network carries out these control functions, the better the performance it can achieve (smaller content access delay on the user side, offloading of mobile backhaul) at the expense of computational resources. More recently, in light of the disruptive network capacity and latency targets advertised by 5G cellular networks [14], the wireless community has been investigating scenarios of joint content caching and user association, where the stored content across the network small caches and the associations of users to small cells are simultaneously determined [15–17]. Putting this into the context of real mobile networks, it essentially implies that the cache placements and user associations are revisited every time a new user joins the network associating with a cell. We expand on this thread adding the recommendation dimension to the problem and explore how these three control mechanisms, i.e., content caching, recommendations and user associations can be jointly tuned to enhance the network performance.

## 21.7.1 Model extension

We consider, as before, a two-layer heterogeneous network with cache-enabled small cells: multiple Small Base Stations (SBSs) and a Macro Base Station (MBS). A user can be associated either to *one* SBS $B_c$,, *or* to the MBS $M$. The association of a user to an SBS is the preferred alternative since it can exploit the capacity and caching capability of the SBSs more efficiently, preserving the radio resources of the macro cell. Nevertheless, a user can always associate with the MBS, as a fallback solution, when the association to an SBS is not feasible. At any given time slot, each user $u \in \mathcal{U}$ is located within the range of a different subset of SBSs. We define as $\mathcal{N}(u)$ the "neighborhood" of user $u$, the set of cells user $u$ can be associated with. Each SBS can serve a set of mobile users within its range, while the MBS can serve the users within range of any SBS. Each cell $c$ has a limited service rate, $B_c$, split among all user devices associated with it. A minimum downlink data rate is guaranteed to each user $u$, incurring an association cost $b_{cu}$ for her association to cache $c$. This cost is strongly related to the physical distance between the user and the cell. In fact, a more distant user generates higher association cost in terms of transmit power for the cell, network interference or even power consumption (for her own device). Thus, the number of users that an SBS can serve is limited by the aggregate *association cost*, which should not be higher than $B_c$, for all cache-enabled SBSs $c \in \mathcal{C}$.

### 21.7.2 The Joint Caching, Recommendations and Association problem

The objective of the Joint Caching, Recommendation, and Association (JCRA) problem is to maximize the portion of total demand that can be satisfied by all caches, assuming also control over the user association process. Let $\{x_{ui}\}$, $\{y_{ic}\}$ and $\{z_{cu}\}$ be three sets of binary decision variables, with $x_{ui} = 1$ if item $i$ is recommended to user $u$ and $x_{ui} = 0$ otherwise; $y_{ic} = 1$ if item $i$ is cached in cache $c$ and $y_{ic} = 0$ otherwise; $z_{cu} = 1$ if user $u$ is associated to cache $c$ and $z_{cu} = 0$ otherwise. Then, reusing the notation in section 21.3, the JCRA problem is formulated as:

$$\max_{\mathbf{y},\mathbf{x},\mathbf{z}} \sum_{u\in\mathcal{U}} \sum_{c\in\mathcal{C}} \sum_{i\in;I} y_{ic} z_{cu} (x_{ui} p_u^{req}(i) + (1 - x_{ui}) \widetilde{p}_u^{req}(i)) \tag{21.14}$$

$$s.t. \sum_{i\in\mathcal{I}} y_{ic} L_i \leq S_c, \qquad \forall c \in \mathcal{C} \tag{21.15}$$

$$\sum_{u\in\mathcal{U}} b_{cu} z_{c_u} \leq B_c, \qquad \forall c \in \mathcal{C} \tag{21.16}$$

$$\sum_{c\in\mathcal{N}(u)} z_{cu} \leq 1, \qquad \forall u \in \mathcal{U}, \tag{21.17}$$

$$x_{ui} = 0, \qquad \forall i \notin \mathcal{W}_u, u \in \mathcal{U}, \tag{21.18}$$

$$\sum_{i\in\mathcal{I}} x_{ui} = R, \qquad \forall u \in \mathcal{U} \tag{21.19}$$

$$y_{ic}, x_{ui}, z_{cu} \in \{0, 1\}, \; u \in \mathcal{U}, i \in \mathcal{I}, c \in \mathcal{C}, \tag{21.20}$$

Constraints (21.15) and (21.16) reflect the cache storage and service cost constraints for each cache, respectively. Constraints (21.17) capture the unique association of users either to cells in their neighbourhood or to the MBS. Constraints (21.18) and (21.19) ensure that the content items recommended to each user are always within her recommendation window and that exactly $R$ items will be recommended to her, respectively.

Controlling the user-to-cache association creates an additional optimization potential. However, it makes the (already difficult) optimization problem even more difficult. In [3], we use generalization arguments to prove that:

**Proposition 9.** The Joint Caching, Recommendations and Association problem is NP-hard.

### *21.7.3 Heuristic algorithms for the JCRA problem*

We present two heuristic solutions to the JCRA problem. The core of both algorithms is the same, i.e., an iterative process for jointly determining the user associations to the small cells and the content to be stored at the small cell caches (see section 21.7.3.1). What changes between the two algorithms is the way recommendations are introduced.

The first heuristic proceeds in three phases:

1. *Initialization phase:*
   (a) At each SBS, assume that all users within its range are associated with it.
   (b) Determine the cache placements at the SBSs caches by solving independent instances of the 0-1 KP, where the utilities of content items equal the aggregate user demand they satisfy if honest recommendations are issued to users (ref. section 21.4, $r_d = 0$).
2. *Iterative phase: Until the cache hit ratio in (21.14) is no further improved do:*
   (a) Given the cache placements determined in (1b), update the user associations at each SBS solving an instance of Generalized Assignment Problem (GAP). Users correspond to items and cells to bins, the cell-specific user costs are the association costs $c$ and the cell-specific user profits are the user demands covered by the current cell cache placement assuming honest recommendations.
   (b) With the new user associations computed in (2a), recompute the content item utility values and solve anew the $C$0-1 KP instances for each SBS assuming honest recommendations.
3. *Recommendation amendment phase:* Adjust recommendations to be both within each user's recommendation window $z_{cu} = 1$ and the cache placement at the cell of user's attachment. The user associations and the cell cache placements are the outputs of the last iteration in (2a) and (2b), respectively.

This heuristic mimics the rationale of the CawR scheme in Algorithm 1. Namely, it uses honest recommendations, corresponding exactly to the user content preferences, as an intermediate step to determine the cache placements (here, also, the user associations). It then amends the recommendations in line with the cache placements to boost the portion of demand that can be locally served by the small cell caches.

An apparent alternative would be to not account for (honest) recommendations at all in the first and second phase of this heuristic and only derive them once as part of the third phase, after the iterations for the determination of cached content and user associations terminate. With this second heuristic, recommendations are *caching-aware* and still boost the part of user content demand that can be satisfied locally. Yet, they do not have an impact on what is eventually cached at the small cells and who are the users served by each of them.

Both heuristics demand an algorithmic solution for the intermediate joint content caching and user association problem. We turn to this problem next.

#### 21.7.3.1 The Intermediate Joint Caching and Association problem

The intermediate Joint Caching and Association (JCAP) problem results from the JCRA problem when omitting the recommendation-related variables and constraints.

$$\max_{\mathbf{y},\mathbf{z}} \sum_{u\in\mathcal{U}} \sum_{c\in\mathcal{C}} \sum_{i\in\mathcal{I}} y_{ic} z_{cu} p_u^{pref}(i)) \tag{21.21}$$

$$s.t.(21.15), (21.16), (21.17) \tag{21.22}$$

$$y_{ic}, z_{cu} \in \{0, 1\},\ u \in \mathcal{U}, i \in \mathcal{I}, c \in \mathcal{C}, \tag{21.23}$$

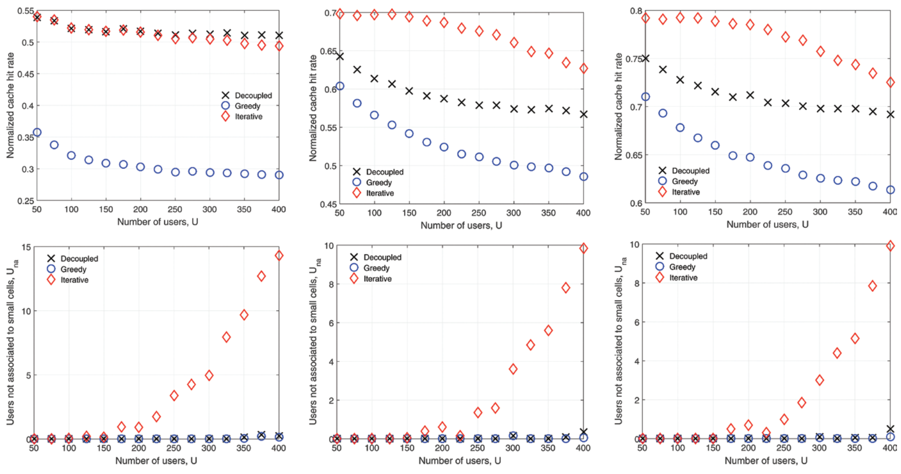
JCAP is an instance of bilinear programming, a special class of non-convex quadratic programming. To show that it is NP-hard, it suffices to remark that when we fix variables $\{y_{ic}\}$, we get the generalized-assignment type problem first treated in [18]. This problem is equivalent to the maximum GAP[e], where SBSs correspond to agents (knapsacks) and users to jobs (items) with agent-specific requirements $\{b_{uc}\}$ and profits $\{\sum_{i\in\mathcal{I}} y_{ic} p_u^{pref}(i)\},\ c \in \mathcal{N}_u$. Since the maximum GAP is an NP-hard problem, the generalization (JCAP) of its equivalent problem in [18] is NP-hard as well. We tackle this problem in [20] and [21], but due to space limitation we will restrict our presentation to the first.

In [20], we propose an iterative algorithm for solving JCAP that corresponds to the first two phases of the heuristic for the JCRA problem (see section 21.7.3), if recommendations are let aside. We first prove its correctness and discuss its computational complexity. Then we proceed in a two-phase evaluation of the algorithm. In the first phase, we consider small problem instances and compare it against the optimal solution, as this can emerge from linearization techniques and use of standard ILP solvers. Table 21.2 gathers statistics on $\Delta H$, the empirical gap between the cache hit rates achieved by the two solutions and $G = \frac{\Delta H}{HR_{opt}}$x100%, the percentage ratio of $\Delta H$ to the optimal value. The median of $G$ over all experiments is practically 0% and the two solutions coincide in more than half of the experiments under two scenarios for the content demand distribution across the network: uniform at random and with strong spatial locality properties.

*Table 21.2 Accuracy of the iterative heuristic*

| Comparison Scenario | var Users random demand | var Users Clustered demand | var Items random demand |
|---|---|---|---|
| median Δ H(G) | 0.001(0.2%) | 0(0%) | 0.004(0.81%) |
| 95th perc. ΔH(G) | 0.097(17.7%) | 0.1(14.2%) | 0.081(13.3%) |
| max ΔH(G) | 0.161(29.2%) | 0.21(25%) | 0.189(35.2%) |

[e]The inequalities in (21.17) are replaced by equalities in the maximum GAP. The equivalence of the two problems is shown in e.g., [19], pp. 190-191



a. No spatial locality, $L_n = 0.3$ b. Spatial locality $N_{cl} = 10$, $L_n = 0.1$ c. Spatial locality $N_{cl} = 10$, $L_n = 0.2$

*Figure 21.7 Comparison of the three heuristics as a function of the number of users: $I=1000$, $C=20$, $B_c=200$, $l_{max}=12$, $b_{max}=20$*

In a second phase, we consider more realistic problem instances and compare the heuristic against simpler algorithms for tackling the user association and content caching tasks: a greedy-like algorithm and another heuristic, called *Decoupled*, which decouples the user association decisions from the content caching ones and tackles user associations first.

Of particular interest are the experiments that look into the sensitivity of the algorithms to the existence or not of spatial locality in the content demand. Under no spatial locality, the decoupled heuristic competes with our iterative heuristic, and even slightly outperforms it for high numbers of users, as shown in Figure 21.7a. At those user levels, the decoupled approach can match all users to SBSs, whereas our iterative heuristic is forced to associate a few with the MBS. The non-satisfied demand of these users corresponds to the marginal performance gain of the decoupled heuristic. The greedy heuristics cannot compete with either of the two alternatives.

When there is spatial locality, the performance of all three heuristics improves (Figure 21.7b). In this scenario, it matters more *which* users will be grouped in a cell rather than *how many*. Factoring the content preferences of users in its decisions, the iterative heuristic clearly outperforms the decoupled one. This performance gain fades out as the number of users grows since the latter manages to squeeze more users in small cells. For more details on the performance characteristics of the iterative heuristic, the interested reader is referred to [20].

## 21.8 Open directions

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. An open research direction consists in translating these largely experimental findings into solid models and validating the presented findings 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 chapter, we considered 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.

Another direction is to consider that the different mechanisms (content caching, recommendations and user association) are controlled by different entities with opposing interests and objectives. For example, the recommendation is controlled by the content provider with the aim to maximize the user engagement and relevance of content recommended; the caching is controlled by the infrastructure provider with the aim to maximize hit ratio; and the association is controlled by the user to minimize the delivery delay or the energy consumption. Game-theoretical tools, either cooperative or not, could be used to describe such kind of scenarios, with each player holding her own objective.

## 21.9 Related work

Related works about the interplay between caching and Recommender Systems (RS) are divided in two main categories: RS are considered either as predicting tools in order to decide a better caching placement (e.g., [6, 22]), or as traffic engineering mechanisms, used as demand-shaping tools. In both categories, RS are used as proxies for inferring the content popularity. However, the last are distinctly different regarding the way they approach RS: 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, their approach to increasing the utility of cache content could be seen as dual to the first: Rather than struggling for accurate predictions of the users' demand for content, they nudge the users' demand towards content items that are common in their preferences.

Due to space limitations, we will provide a taxonomy only for the second category (Figure 21.8), in which our work has made significant contribution. We identify five key factors that differentiate existing works that use the RS as demand-shaping tools: Do they jointly optimize RS and caching decisions? How are recommendations accepted by users? Which is the RS mechanism they use to issue recommendations? Do they take into consideration the general Quality of Service (QoS) that users will receive? Last but not least, are they concerned about the Quality of the issued Recommendations (QoR) and the ethics around "nudged" recommendations?

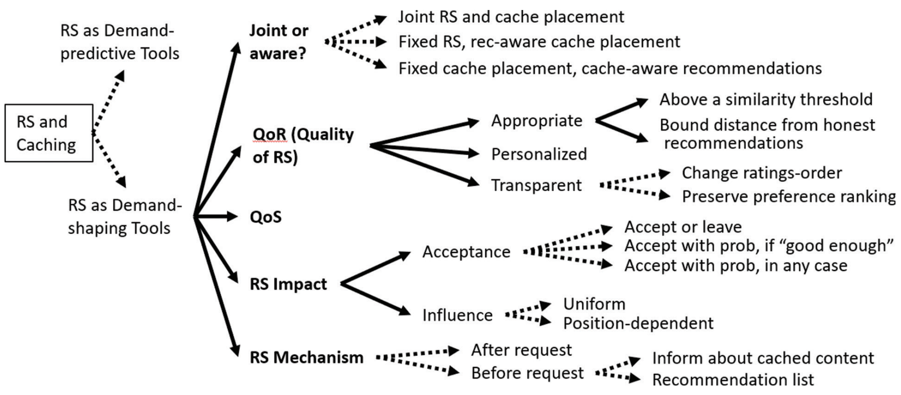

RS as Demand-predictive Tools
RS and Caching
RS as Demand-shaping Tools
Joint or aware?
Joint RS and cache placement
Fixed RS, rec-aware cache placement
Fixed cache placement, cache-aware recommendations
QoR (Quality of RS)
Appropriate
Above a similarity threshold
Bound distance from honest recommendations
Personalized
Transparent
Change ratings-order
Preserve preference ranking
QoS
RS Impact
Acceptance
Accept or leave
Accept with prob, if "good enough"
Accept with prob, in any case
Influence
Uniform
Position-dependent
RS Mechanism
After request
Before request
Inform about cached content
Recommendation list

*Figure 21.8 Taxonomy of works considering the interplay between RS and wireless caching. Solid lines under a category imply that all subcategories can be chosen (the various criteria that characterize works), while dotted lines imply that a work can belong only to one subcategory (either it considers uniform or position-dependent recommendation influence)*

We briefly discuss these factors and their main subcategories, presenting some representative works.

**Joint or aware?** Works are distinguished based on the relation of caching and recommendation decisions. The "soft cache hits" in [23] considers systems where users requesting uncached content are informed about cached content which is similar to their request, and systems where alternative content is directly delivered to them. For a given cache placement, the goal is to find a set of recommendations that will maximize the cache hit ratio. On the contrary, a cache placement is a priori given and only recommendations are decided for sequential recommendations in [24, 25] and list reordering [5] or rating shaping [8] are proposed, aiming at the same goal. The same in [26], aiming to the service cost minimization in P2P networks. We are actually different from this thread of works in *jointly taking both recommendations and caching decisions*.

Influenced by [5], in [2] we introduced a mathematical model that captures the coupling between caching decisions and personalized recommendations. Then [1, 27–30], studied how to jointly take these decisions under different assumptions. In [3] the user-to-AP association is an additional decision taken jointly with the others, capturing the case of Telefonica, the giant network provider that also holds the huge Latin-American content provider Movistar. Works [31, 32] jointly consider recommendation and content pushing decisions, while [33] considers coded caching and recommendations.

**RS Impact.** We consider two factors that characterize the impact of RS on the users' final requests: The probability of acceptance of the recommendation and, if accepted, the distribution of the influence across various positions in the recommendations list. Regarding the acceptance, in [26] it is assumed that users either accept the recommendation, or leave the system. Other works consider users that accept a recommendation with a user-specific [1–3] or universal [24, 25, 27, 30, 34] probability if it is “good enough,” while in [23, 33] they accept it with a probability in any case, and in [35] this probability is learned. Regarding the impact after acceptance, in [2] we assumed uniform influence across items in the recommendation list, while in [1] we introduced a model according to which the demand attracted by recommendations in a non-increasing function of both the item’s position in the recommendation list and the number of recommended items, inline with experimental evidence [5] and providing insights about characteristics of the problem. Similar models are considered also in [3, 25, 29].

**RS mechanism.** Most of the works provide recommendations before the users’ requests, either as a recommendation list as described above, or by informing users about the cached content [35]. In [23] the authors recommend alternative cached content to users *after* their request for uncached content.

**Quality of Service (QoS).** The QoS is taken into consideration in [3] by considering that users receive content from a BS if the received SINR is greater than threshold, ensuring it as an embedded constraint in the optimization problem. In [29] the authors consider a probability of users offloading content from caches, based on their distances, while in [34] users are directly asked through questionnaires.

**Quality of Recommendations (QoR) and ethics.** We consider three factors that characterize the QoR: Appropriateness, personalization and transparency. *Appropriateness* is about being *good enough* and matching adequately the users’ preferences. *Personalization* is about taking into consideration the specific inherent preferences of the *individual users* when issuing recommendations, in contrast to issuing generic recommendations to all users in the system. These are not equivalent: A recommendation can be personalized while not being adequate, e.g., by violating the user’s preferences over an accepted threshold. It can be appropriate for many users but not personalized, e.g., when issuing recommendations about the most popular items to all users. In that case, the recommendations will be adequate to the majority of users, albeit not personalized. *Transparency* is about being transparent with the user, in anything that could violate their preferences. In both of the previous cases, recommendations can be non-transparent if users are not informed about their preferences’ violation.

In [26] and [8] these aspects are not taken into consideration: The authors in [26] propose to recommend content cached by peers and those in [8] to modify the ratings shown to users, without personalization or adequateness constraints. In [33], although the personalized model of [2] is adopted, the interplay with coded caching does not allow to consider the distortion of the users’ preferences in the issued recommendations. Some other works consider adequate but not personalized recommendations. In [29], recommendations that are above a certain (global) threshold of quality, same for all uses in the system. Works [5, 23, 24, 27] and [26], indirectly consider a kind of adequateness in recommendations, in the sense that they recommend cached content similar to that requested by users, or reorder the recommendation lists of YouTube putting the cached on top of them [5]. However, this item-item similarity-based approach does not necessarily lead to personalized recommendations. Moreover, it could end up recommending items from the same, very restricted, pool of cached items.

Our viewpoint to RS raises some concerns, not least ethical ones. To this end, in [2] we introduced a measure called *preference distortion tolerance*, to quantify and strictly bound how much the engineered recommendations distort the original user content preferences. The algorithm we proposed is essentially a form of light-weight 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. An important consideration of the conservative [1, 2] or more general [3] quality metrics we presented is that recommendations should be *both personalized and adequate*, following their original reason of existence: to provide users with *personalized, appealing* content, in order to increase their overall satisfaction and engagement with the platform. Additionally, to the best of our understanding, discussion about *ethics and transparency* of network-related recommendations is only made in [1, 2] and [3], where both a bounded distortion from the users' inherent preferences is guaranteed and a preferences rank-preserving recommendation list is issued. By *carefully* nudging the individual user demand towards content that attracts preference from many users, the RS can result in higher cache hit ratios and enhanced QoE for users. Our results suggest that the proposed approach could yield significant gains for the network performance and the users' satisfaction without disrespecting their individual preferences.

## 21.10 Conclusions

This thread of works 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 these works, 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. 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 catalog size.

This chapter includes material from the following paper: "Jointly Optimizing Content Caching and Recommendations in Small Cell Networks," IEEE Transactions on Mobile Computing, 2018. Reproduced by permission of IEEE.

## Funding

This work was supported by the CHIST-ERA grant CHIST-ERA-18-SDCDN-004, through the General Secretariat for Research and Innovation (GSRI), grant number T11EPA4-00056. This work was supported by the Hellenic Foundation for Research and Innovation (H.F.R.I.) under theFirst Call for H.F.R.I. Research Projects to support Faculty members and Researchers and the procurementof high- cost research equipment grant (Project Number: HFRI-FM17-352).

## References

[1] Chatzieleftheriou L.E., Karaliopoulos M., Koutsopoulos I. 'Jointly optimizing content caching and recommendations in small cell networks'. *IEEE Transactions on Mobile Computing*. 2019;**18**(1):125–38.

[2] Chatzieleftheriou L.E., Karaliopoulos M., Koutsopoulos I. Caching-aware recommendations: Nudging user preferences towards better caching performance. *In: Proc. IEEE INFOCOM*; 2017. pp. 1–9.

[3] Chatzieleftheriou L.E., Darzanos G., Karaliopoulos M., Koutsopoulos I. 'Joint user association, content caching and recommendations in wireless edge networks'. *ACM SIGMETRICS Performance Evaluation Review*. 2018;**46**(3):12–17.

[4] Gomez-Uribe C.A., Hunt N. 'The netflix recommender system: algorithms, business value, and innovation'. *ACM Trans on Management Information Systems*. 2016;**6**(4):13:1–13:19.

[5] Krishnappa D.K., Zink M., Griwodz C., Halvorsen P. 'Cache centric video recommendation: an approach to improve the efficiency of YouTube caches'. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)Tomm*. 2015;**11**(4):1–20.

[6] Verhoeyen M. Optimizing for video storage networking with recommender systems'. *Bell Labs Technical Journal*. 2012;**16**(4):97–113.

[7] Lee M.-C., Molisch A.F., Sastry N., Raman A. 'Individual preference probability modeling and parameterization for video content in wireless caching networks'. *IEEE/ACM Transactions on Networking*. 2019 Apr;**27**(2):676–90.

[8] Tadrous J., Eryilmaz A., El Gamal H., *et al.* 'Proactive content download and user demand shaping for data networks'. *IEEE/ACM Transactions on Networking*. 2015;**23**(6):1917–30.

[9] Vegelius J., Janson S., Johansson F. 'Measures of similarity between distributions'. *Quality and Quantity*. 1986;**20**(4):437–41.

[10] Vazirani V.V. *Approximation algorithms*. Springer; 2001.

[11] Harper F., Konstan J. 'The movielens datasets: history and context'. *ACM Transactions on Interactive Intelligent Systems (TiiS)*. 2015;**5**(4):1–19.

[12] Paschos G., Iosifidis G., Caire G. 'Cache optimization models and algorithms'. *Foundations and Trends in Communications and Information Theory*. 2020;**16**(3 - 4):156–345.

[13] Andreev S., Petrov V., Dohler M., Yanikomeroglu H. 'Future of ultra-dense networks beyond 5G: harnessing heterogeneous moving cells'. *IEEE Communications Magazine*. 2019;**57**(6):86–92.

[14] Andrews J.G., Buzzi S., Choi W., *et al.* 'What will 5G be?' *IEEE Journal on Selected Areas in Communications*. 2014 June;**32**(6):1065–82.

[15] ElBamby M.S., Bennis M., Saad W., *et al.* Content-aware user clustering and caching in wireless small cell networks. *In: International Symposium on Wireless Communications Systems (ISWCS)*; 2014. pp. 945–9.

[16] Pantisano F., Bennis M., Saad W., *et al.* Cache-aware user association in backhaul-constrained small cell networks. *In: International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt). IEEE*; 2014. pp. 37–42.

[17] Wang Y., Tao X., Zhang X., Mao G. 'Joint caching placement and user association for minimizing user download delay'. *IEEE Access*. 2016;**4**:8625–33.

[18] Chalmet L., Gelders L. Lagrangian relaxations for a generalized assignment-type problem. *In: Proc. Second European Congress on Operations Research*; 1976. pp. 103–9.

[19] Martello S., Toth P. 'Knapsack Problems'. *Algorithms and Computer Implementations*. New York, NY, USA: John Wiley & Sons, Inc; 1990.

[20] Karaliopoulos M., Chatzieleftheriou L.E., Darzanos G., Koutsopoulos I . On the joint content caching and user association problem in small cell networks. *In: IEEE International Conference on Communications Workshops (ICC Workshops)*; 2020. pp. 1–6.

[21] Darzanos G., Chatzieleftheriou L.E., Karaliopoulos M., Koutsopoulos I . Content preference-aware user association and caching in cellular networks. *In: WiOPT Workshop on Content Caching and Delivery in Wireless Networks (CCDWN)*; 2020. pp. 1–8.

[22] Kaafar M.A., Berkovsky S., Donnet B., *et al.* 'On the potential of recommendation technologies for efficient content delivery networks'. *ACM SIGCOMM Computer Communication Review*. 2013;**43**(3):74–7.

[23] Sermpezis P., Giannakas T., Spyropoulos T., Vigneri L. 'Soft cache hits: improving performance through recommendation and delivery of related content'. *IEEE Journal on Selected Areas in Communications*. 2018;**36**(6):1300–13.

[24] Giannakas T., Sermpezis P., Spyropoulos T. Show me the Cache: Optimizing Cache-Friendly Recommendations for Sequential Content Access. *In: IEEE WoWMoM*; 2018. pp. 1–9.

[25] Giannakas T., Spyropoulos T., Sermpezis P. The Order of Things: Position-Aware Network-friendly Recommendations in Long Viewing Sessions. *In: WiOpt*; 2019. pp. 1–6.

[26] Munaro D., Delgado C., Menasché D.S. Content recommendation and service costs in swarming systems. *In: ICC*; 2015. pp. 1–6.

[27] Kastanakis S., Sermpezis P., Kotronis V., Dimitropoulos X. CABaRet: Leveraging Recommendation Systems for Mobile Edge Caching. *In: ACM SIGCOMM workshops: MECOMM*; 2018. pp. 1–7.

[28] Liu D., Yang C. A Learning-based approach to joint content caching and recommendation at base stations. *In: IEEE GLOBECOM*; 2018. pp. 1–6.

[29] Qi K., Chen B., Yang C., *et al.* Optimizing caching and recommendation towards user satisfaction. *In: 2018 10th International Conference on Wireless Communications and Signal Processing (WCSP)*; 2018. pp. 1–7.

[30] Guo K., Yang C. 'Temporal-spatial recommendation for caching at base stations via deep reinforcement learning'. *IEEE Access*. 2019;**7**:58519–32.

[31] Lin Z., Chen W. Joint pushing and recommendation for susceptible users with time-varying connectivity. *In: IEEE GLOBECOM*; 2018. pp. 1–6.

[32] Liu D., Yang C. 'A deep reinforcement learning approach to proactive content pushing and recommendation for mobile users'. *IEEE Access*. 2019;**7**:83120–36.

[33] Zhu B., Chen W. Coded caching with joint content recommendation and user grouping. *In: IEEE GLOBECOM*; 2018. pp. 1–6.

[34] Sermpezis P., Kastanakis S., Pinheiro J.I., *et al*. 'Towards QoS-Aware recommendations'. *In: arXiv: 1907.06392*. 2019:1–6.

[35] Guo K., Yang C., Liu T. Caching in base station with recommendation via q-learning. *In: 2017 IEEE Wireless Communications and Networking Conference (WCNC)*; 2017. pp. 1–6.