Approximation Guarantees for the Joint Optimization of Caching and Recommendation

Marina Costantini, Thrasyvoulos Spyropoulos, Theodoros Giannakas, Pavlos Sermpezis

Approximation Guarantees for the Joint Optimization of Caching and Recommendation

Marina Costantini*, Thrasyvoulos Spyropoulos*, Theodoros Giannakas* and Pavlos Sermpezis†
*EURECOM, Sophia-Antipolis, France
†FORTH-ICS and Aristotle University of Thessaloniki, Greece

***Abstract*—Caching popular content at the network edge can benefit both the operator and the client by alleviating the backhaul traffic and reducing access latency, respectively. Recommendation systems, on the other hand, try to offer interesting content to the user and impact her requests, but independently of the caching policy. Nevertheless, it has been recently proposed that designing caching and recommendation policies separately is suboptimal. Caching could benefit by knowing the recommender's actions in advance, and recommendation algorithms could try to favor cached content (among equally interesting options) to improve network performance and user experience. In this paper we tackle the problem of optimally making caching and recommendation decisions jointly, in the context of the recently introduced "soft cache hits" setup. We show that even the simplest (one user, one cache) problem is NP-hard, but that the most generic problem (multiple users, femto-caching network) is approximable to a constant. To the best of our knowledge, this is the first polynomial algorithm with approximation guarantees for the *joint* problem. Finally, we compare our algorithm to existing schemes using a range of real-world data-sets.**

I. Introduction

Storing popular content at the edge of future wireless networks has recently rekindled interest in caching research. Placing content in caches close to the user ensures a better delivery with higher bitrate, shorter delay, etc. [1] as well as reduced congestion on the transport links and core servers. At the same time, recommendation systems greatly affect user requests. It is reported that up to 80% of requests on popular content distribution platforms come from user recommendations [2], [3]. Nevertheless, the traditional role of a recommendation system has been to bring forward items from a vast catalogue that best match the users' interests. Where the content is cached and how this affects delivery (e.g., streaming rate) does not usually play a role in the decision of most recommenders.

Recently, major content providers like Netflix and Google started partnering with Internet Service Providers (ISPs) to implement their own CDN solutions inside the network [4], [5]. This naturally brings together content caching and recommendations, as now the same entity can control and coordinate both, towards better user experience, lower network costs, or both. As a result, some recent works propose caching and recommendation policies that take into account this interplay [6]–[12]. Nevertheless, many of these works still focus on one side of the problem, e.g., network-friendly recommendations [7], [12], or recommendation-aware caching policies [8]. Some works that do try to modify both the caching and recommendation policies are usually based on heuristics [9], [11].

Our focus, in this paper, is in the recently proposed context of *soft cache hits* [8]. There, if the requested content is not locally cached, the user might agree to accept an alternative (but related) content that *is* locally available. One reason for this acceptance might be better streaming quality for similarly interesting content (e.g., it is known that low bitrate can lead to increase in the abandonment rate [13]). Another reason is that the network or content provider that benefits thus from network cost reduction, is willing to offer appropriate incentives to counterbalance the occasional utility loss (e.g. zero-rating). While soft cache hits were formally introduced in [8], [14], other related works also implicitly or explicitly assume soft cache hits. E.g, the pliable index coding of [12] allows for content of lower preference to be served to the user, if that benefits the transmission scheme, while the works of [9], [10] introduce a "distortion" parameter that allows lower (but bounded) relevance content to be recommended, if that content is cached. Hence, the framework of soft cache hits targetted in this work has more general applicability.

In this paper we depart from the assumption of [8] that any content in the cache can be recommended to the user. This new setting is significantly more realistic, since standard caches at the edge can contain in the order of 100-1000 contents [4], while only a few 5-20 contents may be recommended to the user in small devices such as smartphones or laptops. However, we now need to choose not only the contents to cache but also the recommendations offered to the user, which renders the problem considerably more challenging.

Our main contributions can be summarized as follows:

- Starting from the framework of [8], we introduce (per user) recommendation variables, and formulate the problem of jointly optimizing both sets of (caching and recommendations) variables.
- We show that even the simplest version of the joint problem (one user, one cache) is NP-hard, and that straightforward application of existing submodularity-based frameworks [8], [15] does not lead to approximation guarantees.
- We show that using a primal decomposition of the joint problem into an inner (recommendation related) problem and an outer (caching related) problem, leads to an iterative yet efficient (i.e., polynomial) algorithm with constant approximation to the jointly optimal solution. To our best knowledge, this is the first approximability result of such a joint problem.
- Using a range of real-world datasets we show that the proposed algorithm always outperforms existing

heuristics, and discuss the implications of different content types on relative and absolute performance of these schemes.

II. Problem Setup

We tackle the problem of jointly designing which contents to cache at each small cell (SC) and what contents to recommend to each user in the network. The goal is to maximize the number of requests served with the local caches either by (i) providing the user with her original request ("direct" cache hit) or (ii) offering her an alternative content related to her request and stored locally ("soft" cache hit). Our model relies on the hypothesis that since most of the material distributed by content providers such as Netflix and YouTube is entertainment-oriented, the user can be flexible and may accept (with a certain probability) a substitute that is close enough to her original request. Indeed, recommendations have shown to have a significant impact on the user behavior when browsing in entertainment-oriented platforms [16]. Thus our model seeks to provide the user with good recommendations not only from the point of view of showing relevant content that might interest her, but that can also be retrieved easily to have it fast and in good quality. We describe each of the components of our system model in detail below.

A. Network and Caching Model

We consider a set of SCs $\mathcal{M}$ and a set of users $\mathcal{I}$, each of them connected to at least one SC in $\mathcal{M}$. We indicate whether a user i can retrieve content from a base station m with the binary variables $q_{im} \in \{0,1\}$. Our model can be easily extended to consider the uncertainty in the user-SC association by making $q_{im} \in [0,1]$ the probability of finding user i in the range of SC m. Each SC is equipped with a cache memory of capacity C, i.e. that can store up to C contents[1]. We use variables $x_{km} \in \{0,1\}$ to indicate whether content k is stored in the cache of helper m. A user can only retrieve contents from the SCs she is connected to and only if the cache contains the file.

B. Content Model

We denote with $\mathcal{K}$ the catalogue of contents from which the users make their requests. Since many pairs of contents in $\mathcal{K}$ may be related to each other, the catalogue can be represented by a graph where each content is a node and related contents are linked by a weighted edge indicating the level of relevance of one content with respect to the other. For example, if $\mathcal{K}$ is a catalogue of music videos, two songs of the same artist will be linked with a high weight and two songs from different artists but of the same genre will be linked with a low weight. Thus the relations between the contents in $\mathcal{K}$ can be represented with an adjacency matrix $U = \{u_{kn} \in [0,1]\}, k,n = 1,\dots,K$ indicating the relative value that a content has to replace another. The values are normalized to have $u_{kn} = 1$ if $n = k$.

[1]We assume for simplicity that each content has roughly equal size (e.g., video chunks), as is commonly assumed in related work [8], [15], [17]. However, our method can be applied to variable size content as well, giving rise to 'knapsack-type' constraints [18].

C. Recommendation-driven Content Access Model

In our model each user generates random i.i.d requests for contents in catalogue $\mathcal{K}$ where content k is requested with probability p_k that might be different for each user (in which case we will have p_k^i). In our simulations the p_k follow a Zipf distribution as shown in related literature [19].

When a new request for a content k arrives from a user i three things can happen:

1) Direct hit ($q_{im} = 1, x_{km} = 1$): The content is found in one or more caches that the user is connected to.

2) Soft hit ($x_{km} = 0\ \forall m / q_{im} = 1;\ \exists\ n,m : u_{kn} > 0, x_{nm} = 1, q_{im} = 1$): The caches do not contain the requested content k but the user is offered a maximum of N alternative contents n (the recommendations) that are related to the request and found in the caches. If the user accepts to take any of the recommendations instead, a soft hit occurs whose value depends on the u_{kn}.

3) Cache miss ($x_{km} = 0, u_{kn} = 0\ \forall i,k,n,m / q_{im} = 1, x_{nm} = 1$): Neither the requested item nor any related content is found in the accessible caches.

Our model assumes that given a request for content k that is not locally available, the user will be sequentially offered alternative contents n that she might accept with a certain probability $v_{kn} \in [0,1]$. It is reasonable to assume that the probability of accepting a substitute depends directly on how related the substitute is to the content originally requested (and this is what standard recommenders do, e.g. through collaborative filtering). Thus, in the following we will assume $v_{kn} = u_{kn}$, and discard the notation v_{kn}. This model for soft cache hits was first introduced in [8]. However, here we make the assumption that the number of contents accessible by the user is limited by the number of recommendations. While in [8] it was assumed that *any* content stored in the cache could be offered to the user as a substitute of her request, this is actually unrealistic, since the number of contents in the cache may be much larger than what is practical to recommend through an application interface. This new assumption, however, introduces an additional variable to optimize over (the y_{kn}) and an additional constraint ($\Sigma_n y_{kn} \leq N$).

We also consider the possibility that even if two contents k and n are related, the utility of getting one of them when the other is requested depends on the user's particular preferences and thus it may differ between users. Thus in the multi-user case we will consider individual matrices $U^i = \{u_{kn}^i\}$ for each user i. Since the utility of each content has a direct impact on whether it is recommended or not, we will define the user-specific recommendation matrices $Y^i = \{y_{kn}^i\}$ analogously.

Table I summarizes the notation of the variables described above.

III. Problem Formulation and Algorithms

Based on the previous problem description, it is clear that there are two sets of variables to optimize: what is cached where (variables x_{km}) and what to recommend to each user (variables y_{kn}^i). We formalize our objective below.

TABLE I
IMPORTANT NOTATION

$\mathcal{M}$	Set of SCs ($\|\mathcal{M}\| = M$)
$\mathcal{I}$	Set of users ($\|\mathcal{I}\| = I$)
q_{im}	User i is in range of SC ($q_{im} = 1$) or not ($q_{im} = 0$)
C	Storage capacity of a SC
x_{km}	Content k is stored in SC m ($x_{km} = 1$) or not ($x_{km} = 0$)
$\mathcal{K}$	Set of contents ($\|\mathcal{K}\| = K$)
u^i_{kn}	Utility of content n for a user i requesting content k
p_k	Probability of content k being requested
y^i_{kn}	Recommend n when i requests k ($y^i_{kn} = 1$) or not ($y^i_{kn} = 0$)

A. Joint Optimization for a Single Cache

In order to better illustrate the problem's hardness, our methodology, and intuition behind it, we will first tackle the problem of designing the caching and recommendation variables in the single-user single-cache scenario, and we will then extend this result to the multi-user femto-caching setting.

Optimization Problem 1 (Joint Caching and Recommendation - Single User Single Cache)**.**

$$\underset{x,Y}{\textit{maximize}} \quad \sum_{k=1}^{K} p_k \left[1 - \prod_{n=1}^{K} (1 - x_n \cdot u_{kn} \cdot y_{kn})\right] \tag{1}$$

$$\textit{s.t.} \quad \sum_{k=1}^{K} x_k \leq C \tag{2}$$

$$\sum_{n=1}^{K} y_{kn} \leq N, \forall k \tag{3}$$

$$x_n, y_{kn} \in \{0, 1\} \tag{4}$$

In the problem above the objective function (1) computes the expectation of a cache hit over all contents in the catalog. A (potentially soft) hit occurs when a content k is requested and a related ($u_{kn} > 0$) content n that is cached ($x_n = 1$) is recommended ($y_{kn} = 1$). In case of a direct hit, we assume that $u_{kk} = 1$ for all k, and we set $y_{kk} = 1$, i.e. a requested content is always recommended if it is cached (and the user accepts it with probability 1, since she got what she wanted). It is easy to see that the product term in (1) corresponds to the probability that no content n leads to a cache hit: this is clear if the content is not cached ($x_n = 0$), or it is not recommended ($y_{kn} = 0$); if it is cached and recommended, then the user rejects it with probability $1 - u_{kn}$. Constraint (2) states that we cannot cache more contents than the capacity of the cache and constraint (3) limits the number of recommended items to N. Altogether, our general objective could be loosely described as "maximizing the cache hit rate for the operator, while at the same time making sure that the recommendations to each user are relevant".

Previous works considering the possibility of enhancing the cache hits through recommendations [8]–[10] solve a different or partial version of this problem. In [9], [10] the model captures the "distortion" by alternative recommendations in a different manner (as a constraint outside the objective), and the algorithm solves the problem for x_n only and amends the recommendations in a posterior step. In [8] only the caching problem (choosing x_n) is solved without accounting for the selection of the limited number of recommendations to show to the user. Problem 1 can be reduced to that of [8] by setting $N = C$ and making $y_{kn} = 1\ \forall k, n$.

Note that the above problem is already NP, even for a single cache and one user, since the simpler problem in [8] considering variables x_n only is already NP. However, the objective can be decomposed to optimize over each variable separately and iteratively, which allows to define a polynomial-time algorithm with approximation guarantees. We remark that this is not always the case, and submodularity methods such as those in [8], [15] cannot always be extended directly on both sets of variables (this is easy to see with a counterexample and was proved formally in [10]).

Our method solves a primal decomposition of the joint problem in (1): An *outer problem* that maximizes its objective respect to x and an *inner problem* that, for a given x, maximizes the objective in (1) respect to y. This decomposition is equivalent to the original problem and allows to define a "nested" algorithm where the outer loop tries to find the best content to add to the cache, and the inner loop selects the best recommendations for the user for each potential cache configuration. We provide the details in the following.

B. Submodularity-based Approximation Algorithm for Problem 1

Let us first assume that the caching vector (variables x_n) are given and consider the subproblem of maximizing (1) with respect to variables y_{kn}. We will show first that this can be done in polynomial time, so that it can be used as a subroutine next.

Let us denote the terms in the objective as

$$f_k(x, Y) = 1 - \Pi_n \left(1 - x_n \cdot u_{kn} \cdot y_{kn}\right). \tag{5}$$

Let us further denote as

$$f^*_k(x) = \max_{Y} \{1 - \Pi_n \left(1 - x_n \cdot u_{kn} \cdot y_{kn}\right)\}. \tag{6}$$

Lemma 1. *Let*

$$F^*(x) = \max_{Y} \{\Sigma_k p_k \left[1 - \Pi_n \left(1 - x_n \cdot u_{kn} \cdot y_{kn}\right)\right]\},$$

then $F^*(x) = \sum_k p_k \cdot f^*_k(x)$.

Proof. This follows easily from the fact that the constraints for Y (i.e. Eq. (3)) are decoupled per line, so finding the optimal recommendations, given a caching vector, decouples to K independent subproblems of maximizing one term in the sum (corresponding to a row k of Y) subject to the respective constraint (3) for line k. □

Hence, we can focus on optimizing each term in the sum (corresponding to each line k of matrix Y) separately. Let us denote set S as $S = \{j \in 1, K : x_j = 1\}$, i.e. the set of cached elements, and use notation $f^*_k(S)$ and $f^*_k(x)$, interchangeably.

Lemma 2. *Let $u^{(j)}_k(S)$ denote the j^{th} order statistic of elements $\{u_{kn} : n \in S\}$, i.e., the j^{th} largest element in that set. Then,*

$$f^*_k(S) = 1 - (1 - u^{(1)}_k(S)) \cdot (1 - u^{(2)}_k(S)) \ldots (1 - u^{(N)}_k(S)). \tag{7}$$

Proof. Our choice per line consists of choosing at most N contents to recommend, so as to maximize $1 - \Pi_{n \in S} \left(1 - u_{kn} \cdot y_{kn}\right)$. Assume that choosing the N elements with the highest u_{kj} is not optimal, as claimed

above, and recommending another element $l \in S$ leads to a better objective. Let us further assume that l replaces the N^{th} highest content. Then,

$$1-(1-u_k^{(1)}(S))\dots(1-u_k^{(N-1)}(S))\cdot(1-u_{kl}) > 1-(1-u_k^{(1)}(S))\dots(1-u^{(N)}(S))$$
$$\Rightarrow (1-u^{(N)}(S)) > (1-u_{kl}) \Rightarrow u^{(N)}(S) < u_{kl}$$

which is clearly a contradiction, since we assumed that l is *not* among the N-highest u_{kj} values in set S.

It is easy to see that the proof holds (in fact strengthens) when replacing any other order statistic(s) in Eq.(7). □

Lemma 3. *$f_k^*(S)$ is a monotonically non-decreasing function of $|S|$, the cardinality of set S.*

Proof. The proof can be found in Appendix A. □

Theorem 1. *The function $f_k^*(S)$ is submodular in S, for any k.*

Proof. The proof can be found in Appendix B. □

Corollary 2. *The objective of Problem 1, i.e., eq.* (1) *is monotone submodular in x.*

Proof. The objective of Problem 1 is equivalent to

$$\underset{x}{\text{maximize}}\ F^*(x) \equiv \underset{x}{\text{maximize}} \sum_k p_k \cdot f_k^*(x)$$

since it holds that [20]:

$$\max_{X,Y} g(X,Y) = \max_X \Big(\max_Y g(X,Y) \Big)$$

Therefore the objective is a positive weighted sum of monotone submodular functions $f_k^*(S)$, and hence the sum is monotone submodular as well [18]. □

The submodularity and monotonicity properties of the objective function of Problem 1 allow us to define a primal decomposition algorithm with performance guarantees. We propose Algorithm 1 to design the cache vector x and recommendation matrix Y defined in the problem.

The Algorithm works as follows: in an outer loop (lines 4-13) it looks for the content j that maximizes the marginal gain obtained from adding that content to the cache, as is standard with greedy algorithms for submodular problems. However, for each candidate content it performs an inner loop that resorts the utility values for the new cache configuration and chooses the recommendations in accordance with Lemma 2 (function ChooseRecommendations() in line 9), i.e. for each content k it selects the N cached items with highest u_{kn} and sets $y_{kn} = 1$. The procedure is then repeated until the cache is full.

Theorem 3 states the approximation guarantees of Algorithm 1 for the Joint Caching and Recommendation problem for a single cache:

Theorem 3 (Approximation Guarantee of Greedy)**.** *Let x_{OPT} be the optimal caching vector to Problem 1 and x^* the caching vector returned by Algorithm 1.*[2] *It holds that*

$$F^*(x^*) > \left(1-\frac{1}{e}\right) F^*(x_{OPT})$$

Proof. As stated by Corollary 2, the objective function of Problem 1 is monotone submodular, and the problem has a cardinality constraint. It is known that for this class of problems the greedy algorithm achieves in the worst case a $(1-\frac{1}{e})$-approximation solution [18]. □

[2]Note that the optimal recommendation matrices Y_{OPT} and Y^* are immediately obtained from x_{OPT} and x^* respectively by recommending the N cached elements with the highest utility (see Lemma 2)

Algorithm 1 Joint Caching and Recommendation Design

Input: U, N, C, p
Output: x, Y

1: $x = 0_K, Y = 0_{K\times K}$
2: **while** $t < C$ **do**
3: $x^{\text{aux}} = x, R = 0_K$
4: **for** $j = 1, \dots, K$ **do**
5: **if** $x_j == 1$ **then** ▷ If content j is already cached
6: **continue**
7: **else**
8: $x_j^{\text{aux}} = 1$
9: Y = ChooseRecommendations(x^{aux}, N, U)
10: $R_j = \Sigma_k p_k (1 - \Pi_n (1 - x_n^{\text{aux}} \cdot y_{kn} \cdot u_{kn}))$
11: $x_j^{\text{aux}} = 0$
12: **end if**
13: **end for**
14: $j^* = \arg\max_j R$
15: $x_j = 1$
16: **end while**
17: Y = ChooseRecommendations(x, N, U)
18: **return** x, Y

C. Joint Optimization: Multi-User Multi-Cache

The results obtained above for a single cache can be extended to prove submodularity and monotonicity also in the multi-user femtocaching (i.e. multi-cache) scenario. In this new setting, a user $i \in \mathcal{I}$ may be connected to one or more helpers $m \in \mathcal{M}$. We will indicate this association with the indicator matrix q_{im}. We also consider now that the utility of a content to replace another may be user-dependent, and thus now we have a per-user content relation matrix $U^i = \{u_{kn}^i\}$ and an associated recommendation matrix $Y^i = \{y_{kn}^i\}$. We will use $\mathbf{U}$ and $\mathbf{Y}$ to denote the 3-dimensional arrays resulting from the concatenation of the U^i and Y^i matrices, respectively. The problem of maximizing the cache hit ratio with soft cache hits and limited number of recommendations in this new scenario is stated in Problem 2.

Optimization Problem 2 (Joint Network Caching and User-specific Recommendation)**.**

$$\underset{X,\mathbf{Y}}{\textit{maximize}} \sum_{k=1}^{K}\sum_{i=1}^{U} p_k^i \Bigg[1 - \prod_{n=1}^{K}\Bigg[\prod_{m=1}^{M}(1-x_{nm}\cdot q_{im}) + \Big(1-\prod_{m=1}^{M}(1-x_{nm}\cdot q_{im})\Big)(1-u_{kn}^i \cdot y_{kn}^i)\Bigg]\Bigg] \tag{8}$$

$$\textit{s.t.} \quad \sum_{k=1}^{K} x_{km} \le C, \qquad \forall m \in \mathcal{M} \tag{9}$$

$$\sum_{k=1}^{K} y_{kn}^i \le N, \qquad \forall k \in \mathcal{K}, \forall i \in \mathcal{U} \tag{10}$$

$$x_{nm}, y_{kn}^i \in \{0,1\} \tag{11}$$

The steps in the proof of Theorem 1 allow us to easily extend the submodularity and monotonicity results to the

multi-user femtocaching scenario of Problem 2, as shown next.

Corollary 4. *The problem of joint "femto-caching" (i.e. multiple co-dependent caches) and recommendation is also submodular (subject to matroid constraint).*

Proof. The proof can be found in appendix C. □

A greedy algorithm analogous to Algorithm 1 can be defined for Problem 2, where now the outer loop goes through all potential (content, cache) assignments (see appendix C). This algorithm can guarantee a $\frac{1}{2}$-approximation of the optimal solution, since Problem 2 is a maximization problem with a submodular objective and a matroid constraint [21].

IV. Experimental Results

We performed simulations with both synthetic and real-world data to test the performance of our algorithms for jointly designing the caching and the recommendation (from now onward denoted JCR) against two alternative approaches. These are two suboptimal decomposition algorithms that correspond to existing approaches for which the caching and recommendation decisions are taken separately:

Caching policies:

1) Soft Cache Hits (SCH): Chooses the contents to store taking into account the possibility of having soft cache hits but *without* considering the limited number of recommendations. This method is identical to that in [8] and implicitly assumes that all cached contents can be recommended.

2) Popularity (POP): Caches the most popular contents until the cache is full. This approach is completely blind to the possibility of the user acquiring benefit through accepting related content.

Recommendation policy: The recommendations in the two approaches above are chosen *after* having filled the cache with their respective criteria, and these are chosen in accordance with Lemma 2 (for each content k, the N contents with highest u_{kn} are recommended).

For our implementation of the greedy-based approaches (JCR and SCH) we used the *lazy evaluations method*, which exploits the submodularity property of the objective to significantly accelerate the greedy search process (due to lack of space we redirect the interested reader to [18] for a nice explanation of this method).

We performed experiments with synthetic data to observe the effect of the graph structure and the network parameters on the performance of the algorithms. This allowed us to appreciate the relative performance of the schemes in a controlled environment for different setup parameters before testing them on the real-world data. Figure 1 shows these results for a synthetic content graph generated with the Barabassi-Albert model and two different scenarios: (1) catalog size K=500, cache size C=50 and number of links added by each new node n=10, and (2) K=1000, C=75 and n=2. In both cases the recommendations were N=1, the popularity Zipf exponent α=1 and the u_{kn}=0.5.

Figure 1 shows that JCR can beat the other two algorithms in both scenarios and achieve over a 10% of difference with the weakest, but there is a trade-off that (with this particular graph structure) does not allow it to significantly beat both at once. In Scenario 1 the high C/K ratio and the large number of edges favor the "confusion" of SCH, which adopts the strategy of trying to accumulate many soft hits but in the end its performance is constrained by the limited number of recommendations. On the contrary, POP can accrue a lot of direct hits thanks to the large C, and its CHR is further boosted by the soft hits obtained thanks to the high connectivity of the graph. The situation is reversed in Scenario 2: low connectivity and low C/K ratio harm the performance of POP but push SCH towards adopting a strategy more similar to that of JCR, where the fractions of soft and direct hits are more balanced than in Scenario 1. The differences between JCR and SCH will be more notorious when $N \ll C \ll K$, as in real scenarios. However, to appreciate them in our limited-size datasets we had to set it to $N = 1$. As can be expected, increasing N with C fixed makes SCH perform more similarly to JCR, and identically in the limit $N = C$.

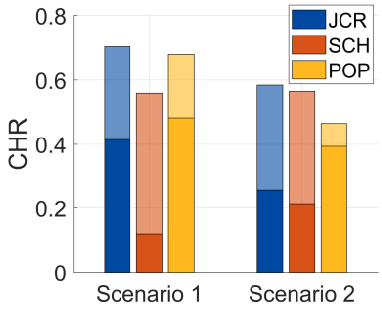

Fig. 1. Expected cache hit ratio (CHR) of the three algorithms tested in the two synthetic scenarios. The fraction of CHR earned by direct and soft hits have both been indicated in each bar in dark and light color, respectively.

For our experiments with real data we considered four datasets:

MovieLens (K=3306): for this dataset we built **U** from the user ratings using collaborative filtering (see details of preprocessing in [8]).

YouTube (K=2098) [22]: here we built matrix **U** setting u_{kn} to non-zero if k is in the list of recommended videos for n or vice versa.

We also considered two datasets not related to video content to further compare the algorithms:

Amazon Videogames (Azn-VG, K=5614) and **Amazon for Android applications** (Azn-Apps, K=8229) [23]: For these we built **U** from the "also bought" list of the datasets, setting $u_{kn} \neq 0$ when items k and n were bought together.

In all cases we kept the largest component of the dataset graphs, and we set the off-diagonal elements of U to u_{kn} = 0.5. Since the Amazon datasets did not contain the content popularity distribution, we synthetically generated random popularity values for each content following a Zipf distribution with exponent α = 2. In our experiments we set the cache size C to be 5% the size of the dataset catalog and N = 1 recommendation was shown to the user for each content request.

Figure 2 shows the performance of our algorithms in the single-cache scenario for all datasets. We have indicated the fraction of cache hits coming from direct and soft hits in dark and light colors, respectively. Our experiments with the femto scenario (not included here) showed very similar trends. Below we make some remarks on these results.

The joint approach outperforms the other two in all cases. In all cases JCR achieves a higher CHR that the other two approaches (both in real and synthetic datasets). This demonstrates the importance of not only accounting for the possibility of soft cache hits, but also for the limited number of recommendations. It is worth noticing that even if greedy has a worst case approximation in theory, in practice it performs very close to optimal.

The magnitude of the difference between JCR and the other algorithms depends very much on the source of the greatest fraction of hits (direct or soft). Whether JCR can beat one or both of the other algorithms by a large difference depends on *how* it does so. In the YouTube dataset, for example, all approaches perform similarly and they go for a lot of direct hits. This suggests that for this dataset the gains are dominated by the popularity values and even JCR and SCH tend to adopt the strategy of POP. Conversely, for MovieLens it seems that the graph structure benefits the acquisition of soft hits. Thus in this case the approaches accounting for soft hits (JCR and SCH) prioritize this source of CHR and adopt a similar strategy that significantly outperforms POP. The greatest gains of JCR respect to the other two are better appreciated when similar gains can be obtained either by caching popular items or by exploiting the soft hits, as observed in the Azn-VG and Azn-Apps datasets.

The overall performance of each algorithm and the differences between them highly depend on the graph structure. As shown in the experiment with synthetic data, by just modifying the graph properties we can invert "who JCR beats by a large difference". The real traces provide further insight into this effect, but due to their complex structure interpreting what is happening in these cases is harder. In future work we will further explore how graph-specific properties affect the algorithm performance when cache hits can be accrued through recommendations.

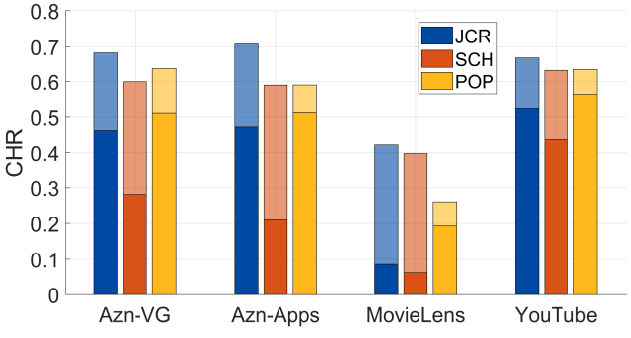

Fig. 2. Expected CHR of the three tested algorithms in all four datasets.

V. Conclusion

Caching and recommendation are two distinct technologies that impact on each other but that are currently optimized separately. We have proposed a simple algorithm with provable approximation guarantees that can jointly design the caching and recommendation strategies in very general scenarios and consistently outperform schemes that take these choices separately. We observed how the magnitude of the gains highly depends on the graph structure of the dataset. Future work will be dedicated to better understand how the graph properties of the content matrix impact on the algorithm performance when soft cache hits are allowed.

Acknowledgements

This work was partially funded by the IMT F&R program, by the ANR-17-CE25-0001 Jeunes Chercheuses et Jeunes Chercheurs project 5C-for-5G, and by Greece and EU (European Social Fund) in the framework of the Operational Programme "Human Resources Development, Education and Lifelong Learning" within the Action "Support to Postdoctoral Researchers, 2nd round" (MIS 5033021) of the Greek State Scholarships Foundation.

Appendix A
Proof of Lemma 3

According to Lemma 2, for set S, it holds that $f_k^*(S) = 1-(1-u_k^{(1)}(S))\cdot(1-u_k^{(2)}(S))\dots(1-u_k^{(N)}(S))$. Now, assume that we add some element i in S. Then, there exist two cases:

(a) If $u_{ki} \leq u_k^{(N)}(S)$, then $f_k^*(S\cup\{i\}) = f_k^*(S)$, the objective remains unchanged.

(b) If $u_{ki} > u_k^{(N)}(S)$, then

$$
\begin{aligned}
&f_k^*(S\cup\{i\}) = \\
&= 1-(1-u_k^{(1)}(S))\dots(1-u_k^{(N-1)}(S))\cdot(1-u_{ki}) \\
&= 1-(1-f_k^*(S))\frac{(1-u_{ki})}{(1-u_k^{(N)}(S))} \\
&\overset{(a)}{\geq} f_k^*(S)\left(1-\frac{(1-u_{ki})}{(1-u_k^{(N)}(S))}\right)+f_k^*(S)\frac{(1-u_{ki})}{(1-u_k^{(N)}(S))} \\
&= f_k^*(S),
\end{aligned}
$$

where in (a) we used the fact that $f_k^*(S) \leq 1$. This completes the proof.

Appendix B
Proof of Theorem 1

Consider the powerset $\mathcal{F}$ of $\{1,2,\dots,K\}$. Now, assume that we add element i to some set $S\in\mathcal{F}$. We denote

$$\Delta f(S,i) = f_k^*(S\cup\{i\}) - f_k^*(S). \tag{12}$$

Then, we can consider the following cases:

$$
\begin{aligned}
&\left(u_{ki} \leq u_k^{(N)}(S)\right) \Rightarrow \Delta f(S,i) = 0. \\
&\left(u_{ki} > u_k^{(N)}(S)\right) \Rightarrow \Delta f(S,i) = (1-u_k^{(1)}(S))\dots \\
&\qquad \dots(1-u_k^{(N-1)}(S))\cdot\left[u_{ki}-u_k^{(N)}(S)\right]
\end{aligned} \tag{13}
$$

To show submodularity, we need to prove that $\Delta f(A,i) - \Delta f(B,i) \geq 0$, for any sets $A,B\in\mathcal{F}$, such that $A\subset B$, and any $i\in\{1,2,\dots,K\}$. There are three separate cases to consider:

(Case 1) $\Delta f(A,i) = \Delta f(B,i) = 0$: Adding element i to either set A or B does not improve the objective (i.e., the new content i added in the cache, has a value u_{ki} that is lower than the top N values of contents already in A (or B)).

(Case 2) $\Delta f(A,i) > 0, \Delta f(B,i) = 0$: Then, $\Delta f(A,i) - \Delta f(B,i) = (1-u_k^{(1)}(S)) \dots (1-u_k^{(N-1)}(S)) \cdot [u_{ki} - u_k^{(N)}(S)]$, according to eq. (13), which is strictly higher than 0.

(Case 3) $\Delta f(A,i) > 0, \Delta f(B,i) > 0$: In this case, it is easy to see according to Lemma 3, that the highest order statistics of sets A and B will coincide up to some order m, and will differ from order $m+1$ up to N Then, we can write

$$\Delta f(A,i) = (1-u_k^{(1)}(B)) \dots (1-u_k^{(m)}(B))(1-u_k^{(m+1)}(A)) \dots \dots (1-u_k^{(N-1)}(A)) \left[u_{ki} - u_k^{(N)}(A)\right] \quad (14)$$

$$\Delta f(B,i) = (1-u_k^{(1)}(B)) \dots (1-u_k^{(m)}(B))(1-u_k^{(m+1)}(B)) \dots \dots (1-u_k^{(N-1)}(B)) \cdot \left[u_{ki} - u_k^{(N)}(B)\right] \quad (15)$$

Let's denote the common term in the product as C, and as C_A and C_B the different terms in each product, respectively. Then,

$$\begin{aligned} \Delta f(A,i) - \Delta f(B,i) &= \\ &= C\left[C_A\left(u_{ki} - u_k^{(N)}(A)\right) - C_B\left(u_{ki} - u_k^{(N)}(B)\right)\right] \\ &\geq C \cdot C_B \cdot \left[\left(u_{ki} - u_k^{(N)}(A)\right) - \left(u_{ki} - u_k^{(N)}(B)\right)\right] \quad (16) \\ &= C \cdot C_B \cdot \left(u_k^{(N)}(B) - u_k^{(N)}(A)\right) \geq 0. \quad (17) \end{aligned}$$

Eq. (16) follows from the fact that $f_k^*(S)$ is monotonically increasing and thus $f_k^*(B) \geq f_k^*(A) \Rightarrow 1 - C \cdot C_B \geq 1 - C \cdot C_A \Rightarrow C_A \geq C_B$.

Eq. (17) also follows easily from Lemma 3, as all order statistics are monotonically increasing in the cardinality of the considered set.

Finally, the case $\Delta f(A,i) = 0, \Delta f(B,i) > 0$ cannot occur due to f_k^* being monotonically increasing in the cardinality of the chosen set of elements. This concludes the proof that $\Delta f(A,i) - \Delta f(B,i) \geq 0$ for all possible cases, and thus that f_k^* is submodular.

APPENDIX C
PROOF OF COROLLARY 4

We may denote the terms between the outer square brackets as $f_{ki}(X, \mathbf{Y}^i)$ and $f_{ki}^*(X) = \max_{\mathbf{Y}^i} f_{ki}(X, \mathbf{Y}^i)$. We may then repeat the argument of Lemma 1 to show that $F^*(X) = \sum_k \sum_i p_k^i f_{ki}^*(X)$, since we can maximize the terms in the sum for each k and i independently.

Let us denote $S = \{(\ell, m), \ell \in \{1,K\}, m \in \{1,M\} : x_{\ell m} = 1\}$ the pairs (content, helper) such that content ℓ is stored in helper m. We will use the notation $f_{ki}^*(S)$ and $f_{ki}^*(X)$ interchangeably. Furthermore, let $S_i \subset S$ be the subset of pairs in S such that user i has access to a helper $m \in S$, i.e. $q_{im} = 1$. Note that since a user cannot get any benefit (cache hit) from a helper that she is not connected to, $f_{ki}^*(S) = f_{ki}^*(S_i)$, and furthermore $f_{ki}^*(S_i) = 1 - (1-u_k^{(1)}(S_i)) \cdot (1-u_k^{(2)}(S_i)) \dots (1-u_k^{(N)}(S_i))$ by the same arguments used in Lemma 2. Therefore if we decouple the problem *per user* and take the elements of S to be the pairs (ℓ, m) we can repeat the steps of Lemma 3 and Theorem 1 to show submodularity and monotonicity for $f_{ki}^*(S)$. Since the objective $F^*(X)$ of Problem 2 is a positive weighted sum of $f_{ki}^*(S)$, it is monotone submodular as well.

REFERENCES

[1] T. V. Doan, L. Pajevic, V. Bajpai, and J. Ott, "Tracing the path to youtube: A quantification of path lengths and latencies toward content caches," *IEEE Communications Magazine*, vol. 57, no. 1, pp. 80–86, 2019.

[2] C. A. Gomez-Uribe and N. Hunt, "The netflix recommender system: Algorithms, business value, and innovation," *ACM Transactions on Management Information Systems (TMIS)*, vol. 6, no. 4, p. 13, 2016.

[3] R. Zhou, S. Khemmarat, and L. Gao, "The impact of youtube recommendation system on video views," in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pp. 404–410, ACM, 2010.

[4] "Netflix open connect." https://openconnect.netflix.com/en/.

[5] "Google global cache." https://peering.google.com/#/.

[6] D. K. Krishnappa, M. Zink, C. Griwodz, and P. Halvorsen, "Cache-centric video recommendation: an approach to improve the efficiency of youtube caches," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 11, no. 4, p. 48, 2015.

[7] T. Giannakas, P. Sermpezis, and T. Spyropoulos, "Show me the cache: Optimizing cache-friendly recommendations for sequential content access," in *19th IEEE International Symposium on "A World of Wireless, Mobile and Multimedia Networks", WoWMoM 2018, Chania, Greece, June 12-15, 2018*, pp. 14–22, 2018.

[8] P. Sermpezis, T. Giannakas, T. Spyropoulos, and L. Vigneri, "Soft cache hits: Improving performance through recommendation and delivery of related content," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 6, pp. 1300–1313, 2018.

[9] L. E. Chatzieleftheriou, M. Karaliopoulos, and I. Koutsopoulos, "Caching-aware recommendations: Nudging user preferences towards better caching performance," in *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*, pp. 1–9, IEEE, 2017.

[10] L. E. Chatzieleftheriou, M. Karaliopoulos, and I. Koutsopoulos, "Jointly optimizing content caching and recommendations in small cell networks," *IEEE Trans. Mob. Comput.*, vol. 18, no. 1, pp. 125–138, 2019.

[11] Z. Lin and W. Chen, "Joint pushing and recommendation for susceptible users with time-varying connectivity," in *GLOBECOM*, pp. 1–6, 2018.

[12] L. Song and C. Fragouli, "Making recommendations bandwidth aware," *IEEE Trans. Information Theory*, vol. 64, no. 11, pp. 7031–7050, 2018.

[13] H. Nam, K.-H. Kim, and H. Schulzrinne, "Qoe matters more than qos: Why people stop watching cat videos," in *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*, pp. 1–9, IEEE, 2016.

[14] T. Spyropoulos and P. Sermpezis, "Soft cache hits and the impact of alternative content recommendations on mobile edge caching," in *Proc. ACM Workshop on Challenged Networks (CHANTS)*, pp. 51–56, 2016.

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

[16] R. Zhou, S. Khemmarat, and L. Gao, "The impact of youtube recommendation system on video views," in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pp. 404–410, ACM, 2010.

[17] K. Poularakis, G. Iosifidis, and L. Tassiulas, "Approximation algorithms for mobile data caching in small cell networks," *IEEE Transactions on Communications*, vol. 62, no. 10, pp. 3665–3677, 2014.

[18] A. Krause and D. Golovin, "Submodular function maximization," *Tractability: Practical Approaches to Hard Problems*, vol. 3, no. 19, p. 8, 2012.

[19] L. A. Adamic and B. A. Huberman, "Zipf's law and the internet.," *Glottometrics*, vol. 3, no. 1, pp. 143–150, 2002.

[20] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[21] M. L. Fisher, G. L. Nemhauser, and L. A. Wolsey, "An analysis of approximations for maximizing submodular set functions—ii," in *Polyhedral combinatorics*, pp. 73–87, Springer, 1978.

[22] "Dataset for statistics and social network of youtube videos." http://netsg.cs.sfu.ca/youtubedata/, 2008.

[23] J. McAuley, C. Targett, Q. Shi, and A. Van Den Hengel, "Image-based recommendations on styles and substitutes," in *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 43–52, ACM, 2015.