Show me the Cache: Optimizing Cache-Friendly Recommendations for Sequential Content Access

Theodoros Giannakas[1], Pavlos Sermpezis[2], and Thrasyvoulos Spyropoulos[1]
[1] EURECOM, France, first.last@eurecom.fr
[2] FORTH, Greece, sermpezis@ics.forth.gr

***Abstract*—Caching has been successfully applied in wired networks, in the context of Content Distribution Networks (CDNs), and is quickly gaining ground for wireless systems. Storing popular content at the edge of the network (e.g. at small cells) is seen as a "win-win" for both the user (reduced access latency) and the operator (reduced load on the transport network and core servers). Nevertheless, the much smaller size of such edge caches, and the volatility of user preferences suggest that standard caching methods do not suffice in this context. What is more, simple popularity-based models commonly used (e.g. IRM) are becoming outdated, as users often consume multiple contents in sequence (e.g. YouTube, Spotify), and this consumption is driven by recommendation systems. The latter presents a great opportunity to bias the recommender to minimize content access cost (e.g. maximizing cache hit rates). To this end, in this paper we first propose a Markovian model for recommendation-driven user requests. We then formulate the problem of biasing the recommendation algorithm to minimize access cost, while maintaining acceptable recommendation quality. We show that the problem is non-convex, and propose an iterative ADMM-based algorithm that outperforms existing schemes, and shows significant potential for performance improvement on real content datasets.**

I. Introduction

At the dawn of the era of 5G, we witness dramatic changes in *network architecture* and *user traffic demand*, two main elements that shape the ecosystem of modern and future mobile networks. New architectures for cellular networks, comprising densification of the access network (small-cells, SCs) and integration of computing and caching capabilities in base stations (mobile edge caching & computing, MEC), have been proposed [1], to cope with the recent boom in traffic demand [2] and the envisioned increase in devices/traffic density in the near future ($\times 10k$ more traffic and $\times 10-100$ more devices [3]). At the same time, the higher data rates offered by mobile operators, lead to changes in the consuming habits of users as well. Users nowadays consume more video and music traffic through their mobile devices (video will be 82% of the total traffic by 2021 [2]).

Video (and music) traffic is highly skewed [4], [5], [6], with a relatively small number of popular videos creating a large part of the demand. This skewness favors caching efficiency, and has made Content Distribution Networks (CDNs) a major component of today's wired Internet. Nevertheless, these ideas are not immediately applicable to future wireless networks. Even putting aside the interference-prone nature of the wireless channel, several factors limit the gains from mobile edge caching, in practice: (i) The size of content catalogues (even of only the most popular contents) is typically much larger than the storage capacity of a small base station. As a result, a large number of requests will always have to be redirected through the backhaul no matter what caching policy is used [7], [8]. (ii) The smaller coverage area of SCs implies frequent changes in the set of associated users in a SC, and this introduces larger variation in content request patterns. Thus, it becomes more difficult for a caching algorithm to accurately predict the "local content popularity" [7]. (iii) Viral videos routinely receive hundreds of thousands of views within a few hours after their upload [9] (and usually fade in equally fast). To accommodate for such trending videos, frequent updates of cached content is needed, which imposes a heavy load for the backhaul.

A large number of *cooperative caching* policies have been recently proposed aiming to improve wireless caching efficiency, by exploiting coverage overlaps of small-cells [10], [11], coded transmissions [12], caching on user equipment [13], [14], [15] and vehicles [16], [17], and others [18]. While these policies do offer considerable benefits *in theory*, and under certain conditions some practical gains as well, they also face the above obstacles (as well as some additional ones, e.g., subpacketization complexity [19], limited resources in the case of device-side caching, etc.)

The above discussion suggests that caching policy alone is limited in the amount of performance gain it can bring at a small edge cache. Increasing cache capacity (to improve hit rates) or backhaul capacity (to allow for more frequent updates) seem like the only way to cope with this problem, but these are "hardware" solutions involving significant CAPEX/OPEX costs, when considering the very large number of small base stations envisioned in future heterogeneous and ultra-dense networks. The following question then arises: *Are there any practical "software-based" solutions that can improve caching efficiency, at a low cost?*

Our answer to this question follows from two key observations: (i) the performance of a caching algorithm is dependent on user request patterns; (ii) user requests are increasingly driven by recommendation algorithms [20], [21], [22]. For example, Netflix reports that around 80% of its video views are through recommendations [23], while the corresponding percentage for YouTube's related video is 50% [22]. Our proposal is therefore to not try to further improve what is stored at each cache, but rather *to better exploit the already*

978-1-5386-4725-7/18/$31.00 © 2018 IEEE

cached content by taking advantage of the recommendation algorithms, integrated in many of the content services that dominate the traffic consumption (e.g., YouTube, Netflix, Spotify). For instance, upon peak hours, when the backhaul is congested, a recommendation system could put higher preference on recommending content pre-cached in the vicinity of a user (e.g., at the base station with which she is associated, or to a nearby cache).

However, recommendations cannot be arbitrarily manipulated in practice. The recommendation system still needs to accomplish its primary goal, i.e., recommend to users contents of their interest. Hence, the goal of this paper is to *leverage the recommendation system to increase the caching efficiency, while at the same time ensuring high quality recommendations*. Note also that, recommending a content that is almost as interesting to the user but locally cached might not just be “acceptable to the user, better for the network”, but even *beneficial to both the user and the network*, if that content can be streamed for example at better quality (since congested links are avoided).

The idea of improving edge caching by taking into account recommendations has been recently proposed [24], [25], [26], [27], by optimizing caching policies based on [24], [25] or jointly with recommendation systems [26], [27]. These solutions require control over the caching policy (e.g., if content and network provider are the same entities or collaborate), which might not be feasible for every scenario. What is more, these works assume simple models of content access, such as the Independent Reference Model (IRM), where consecutive requests by a user are independent. In this paper, we go beyond the state of the art, by making the following contributions:

- To our best knowledge, we propose the first model for recommendation-driven sequential user requests, that better fits real users behavior in a number of popular applications (e.g. YouTube, Vimeo, personalized radio). We then formulate the problem of maximizing the impact of recommendations on cache performance, while maintaining high quality for related contents (Sections II and III).
- We show that the problem is non-convex, and propose an ADMM-like iterative algorithm that fully exploits the structure of the sequential content access statistics (Section IV).
- Using simulations on a number of real datasets for different content types, we show that our algorithm significantly outperforms baseline approaches and previous work, improving caching performance in a large range of setups, while maintaining the desired recommendation quality (Section V).

As a final note, while in this paper we consider the problem in the simple(r) context of a layer of caches with increasing cost, the proposed approach can be applied on top of a number of the cooperative wireless caching policies discussed earlier, e.g., [10], [12], [24]. We plan to explore the interplay of such advanced caching policies and the proposed cache-friendly recommender as part of future work.

II. Problem Setup

Content Traffic. We consider a content catalogue $\mathcal{K}$ of cardinality K, corresponding to a specific application (e.g. YouTube). A user can request a content from this catalogue either by asking *directly* for the specific content (e.g., in a search bar) or by following a *recommendation* of the provider. In practice, users spend on average a long time using such applications, e.g., viewing several related videos (e.g., 40 min. at YouTube [28]), or listening to personalized radio while travelling.

Recommendation System. Recommendation systems have been a prolific area of research in the past years, and often combine content features, user preferences, and context with one or more sophisticated methods to predict user-item scores, such as collaborative filtering [29], matrix factorization [30], deep neural networks [31], etc. We will assume for simplicity that the *baseline* recommender system (RS) for applications where the user consumes multiple contents works as follows:

(i) The RS calculates a similarity score u_{ij} between every content $i, j \in \mathcal{K}$, based on some state-of-the-art method; this defines a similarity matrix $U \in \mathbb{R}^{K \times K}$. Without loss of generality, let $u_{ij} \in [0, 1]$, where we normalize values so that $u_{ij} = 0$ denotes unrelated contents and $u_{ij} \to 1$ “very related contents”. W.l.o.g. we set $u_{ii} = 0, \forall i \in \mathcal{K}$ for all contents. Note also that this U might differ per user.

(ii) After a user has just watched content i, the RS recommends the N contents with the highest u_{ij} value [22], [31]. N is usually a small number (e.g. values of $3-5$ are typical for the default YouTube mobile app) or sometimes $N = 1$, as in the case of personalized radio (Spotify, last.fm) or “AutoPlay” feature in YouTube where the next content is simply sent to the user automatically by the recommender.

Caching Cost. We assume that fetching content i is associated with a cost $x_i \in \mathbb{R}$, which is known to the content provider. This cost might correspond to the delay experienced by the user, the added load in the backhaul network, or even monetary cost (e.g. for an Over-The-Top content provider leasing the infrastructure). It can also be used to capture different caching topologies. For example, to simply maximize the cache hit rate, we could set $x_i = 0$ for cached content, and $x_i = 1$ for non-cached. For hierarchical caching [11], [32], the cost increases if the content is cached deeper inside the network.

Finally, as mentioned earlier, the specific wireless setup is relatively orthogonal to our approach and beyond the scope of this work. However, as a simple example, consider the well-known femto-caching setup [10]. The proposed algorithm there would first decide what will be cached at each base station. Then, x_i would have a low value for all content that the user in question can fetch from some BS in range (possibly dependent on the SINR of the BS, as well [10]), and a high value otherwise.

User Request Model. Based on the above setup, we assume the following content request model.

Definition 1 (User Request Model)**.** *After a user has consumed a content i, then*

- *(recommended request) with probability a the user picks one of the N recommended contents with equal probability $\frac{1}{N}$.*
- *(direct request) with probability $1 - a$ it ignores the recommender, and picks any content j from the catalogue with probability p_j, where $p_j \in [0, 1]$ and $\sum_{j=1}^{K} p_j = 1$.*

p_j above represents an underlying (long-term) popularity of content j, over the entire content catalogue. For short, we denote the vector $\mathbf{p_0} = [p_1, \ldots, p_K]^T$. Note that the above model can easily generalized to consider different probabilities to follow different recommended contents (e.g. based on their ranking on the recommended list). Note also the assumption that a is fixed: for instance, for applications where the user cannot evaluate the content quality before she actually consumes the content, this assumption is realistic, at least in the "short term". In the remainder of the paper, we assume that if the recommendation quality is above a threshold, then the user's trust in the recommender (i.e. the value of a) remains fixed. We plan to explore scenarios where a changes at every step, as a function of recommendation quality, in future work.

Recommendation Control. Our goal is to modify the user's choices through the "recommended request" part above, by appropriately selecting the N recommended items. Specifically, let an indicator variable z_{ij} denote whether content j is in the list of N recommended contents, after the user has watched content i. If $z_{ij} \in \{0, 1\}$, the problem would be combinatorial and in most cases NP-hard. We can relax this assumption by letting $z_{ij} \in [0, 1]$, and $\sum_j z_{ij} = N, \forall i$. z_{ij} can be interpreted now as a probability. For example, if $z_{13} = 0.5$, then content 3 will be recommended half the times after the user consumes content 1. To facilitate our analysis we can further normalize this by defining variables $y_{ij} = \frac{z_{ij}}{N}$, $y_{ij} \in [0, \frac{1}{N}]$. It is easy to see that y_{ij} define a stochastic matrix Y. Putting everything together, the above user request model can be defined by a Markov chain, whose transition matrix P is given by

$$P = a \cdot Y + (1 - a) \cdot P_0, \tag{1}$$

where $P_0 = \mathbf{1} \cdot \mathbf{p_0^T}$ is a rank-1 matrix ($P_0 \in \mathbb{R}^{K \times K}$), equal to the *outer product* of a vector with K unit values and the direct request vector $\mathbf{p_0}$. The above model of content requests, and the corresponding Markov Chain, is reminiscent of the well-known "PageRank model" [33], where a web surfer either visits an arbitrary webpage i (with a probability p_i) or is directed to a webpage j through a link from a webpage i (with probability p_{ij}).

Table I summarizes some important notation.

TABLE I: Important Notation

$\mathcal{K}$	Content catalogue (of cardinality K)
u_{ij}	Similarity score for content pair $\{i, j\}$
N	Number of recommended contents after a viewing
x_i	Cost for fetching content i
a	Prob. the user requests a recommended content
p_j	Average a priori popularity of content j
$\boldsymbol{p_0}$	A priori popularity distribution of contents, $\in \mathbb{R}^K$
z_{ij}	Prob. the RS recommends content i after viewing j
y_{ij}	Normalized prob. $y_{ij} = \frac{z_{ij}}{N}$
$\boldsymbol{\pi}$	Stationary distribution of contents, $\in \mathbb{R}^K$
$\mathcal{C}$	Set of cached content (of cardinality C)

III. Problem Formulation

Given the above setup, our general goal in this paper is *to reduce the total cost of serving user requests by choosing matrix Y, while maintaining a required recommendation quality.*

Consider a user that starts a session by requesting a content $i \in \mathcal{K}$ with probability p_i (i.e., we assume her initial choice is not affected by the recommender), and then proceeds to request a sequence of contents according to the Markov chain P of Eq.(1). Assume that the user requests M contents in sequence. Then the associated access cost would be given by

$$\sum_{m=0}^{M} \mathbf{p_0}^T \cdot P^m \cdot \mathbf{x}, \tag{2}$$

where $\mathbf{x} = [x_1, ..., x_K]^T$ is the vector of the costs per content (see Section II).

M is a random variable though, and the various powers of transition matrix P, which contains the control variable Y, would greatly complicate the problem. However, the above Markov chain is strongly connected and ergodic under very mild assumptions for $\mathbf{p_0}$. It thus has a stationary distribution $\boldsymbol{\pi} = [\pi_1, ..., \pi_K]^T$, which is also equal to the long-term percentage of total requests for content i. Consequently, for M large enough we can approximate the average cost *per request* with

$$\boldsymbol{\pi}^T \cdot \mathbf{x} \tag{3}$$

where $\boldsymbol{\pi}$ can be calculated from the following lemma.

Lemma 1. *The stationary distribution $\boldsymbol{\pi}$ is given by*

$$\boldsymbol{\pi}^T = (1 - a) \cdot \mathbf{p_0}^T \cdot (I - a \cdot Y)^{-1} \tag{4}$$

where I the $K \times K$ identity matrix.

Proof. The stationary distribution above can be derived through the standard stationary equality [34]

$$\boldsymbol{\pi}^T = a \cdot \boldsymbol{\pi}^T \cdot Y + (1 - a) \cdot \mathbf{p_0}^T, \tag{5}$$

by observing that matrix $(I - a \cdot Y)$ has strictly positive eigenvalues (in measure). See also [35], for more details. □

We are therefore ready to formulate cache-friendly recommendations as an optimization problem.

Optimization Problem 1 (Cache-Friendly Recommendations)**.**

$$\underset{Y}{\textit{minimize}} \quad \mathbf{p_0}^T \cdot (I - aY)^{-1} \cdot \mathbf{x}, \tag{6}$$

$$0 \leq y_{ij} \leq \frac{1}{N}, \ \ \forall \ i \ and \ j \ \in \ \mathcal{K}. \tag{6a}$$

$$\sum_{j=1}^{K} y_{ij} = 1, \ \ \forall i \ \in \mathcal{K} \tag{6b}$$

$$y_{ii} = 0, \ \ \forall \ i \ \in \mathcal{K} \tag{6c}$$

$$\sum_{j=1}^{K} y_{ij} u_{ij} \geq q_i, \ \ \forall i \ \in \mathcal{K} \tag{6d}$$

Objective. The objective is to minimize the expected cost to access any content, and follows directly from Eq. (3) and Lemma 1. Note that we have dropped the constant $(1-a)$ from Eq. (4), as it does not affect the optimal solution.

Control Variables. The variables y_{ij} (K^2 in total), deciding what is recommended after each content i, constitute the control variables.

Constraints. The first three constraints make sure that y_{ij} forms a stochastic transition matrix that can be translated to N recommendations per item i. Specifically, the "box" constraints of Eq. (6a) ensures that all entries are positive, and smaller or equal to $1/N$. The latter is necessary as matrix Y is a *normalized* version of the recommendations. Together with Eq. (6b) these ensure that exactly N contents are recommended for every i (see also Section II, "Recommendation Control"). Eq. (6c) simply ensures that the same content cannot be recommended when it was just consumed.

Quality Constraint. Eq. (6d) ensures that that the "quality" of recommended contents for each i is above a desired threshold. Observe that, without this constraint, the optimal solution to the above problem is trivial, namely to always recommend the same N contents j with the minimum cost x_j. However, these contents will probably be unrelated (i.e. $u_{ij} \rightarrow 0$) essentially "breaking" the recommender. Hence, this constraint forces variables y_{ij} to select high u_{ij} values to ensure the recommender keeps doing its primary job, namely finding related contents. Note that, if there are at least N strongly related contents for each i (i.e., $u_{ij} = 1$), then the maximum value for q_i is 1. W.l.o.g., in the remainder we will assume the same quality constraint for all i ($q_i = q$).

The remaining quantities are constants, and inputs to the problem. While some of them might still vary over time (e.g., u_{ij}), we assume this occurs at a larger time scale, compared to our problem.

Unfortunately, the objective function (Eq. (6)) is non-convex, unless Y is positive semidefinite and symmetric. In that case, the problem could be cast into an SDP (Semi-Definite Program) using Schur's complement [36]. However, forcing Y to be symmetric in our problem leads to trivial solutions, as every symmetric Markov chain has a uniform invariant measure. What is more, the inverse in the objective further complicates solving this problem, as the gradient of this expression is rather complex.

IV. Optimization Algorithm

Given that Optimization Problem 1 is non-convex, there are no polynomial-time algorithms that can guarantee to converge to the optimal solution. This leaves us with two options for solving the problem: to apply (i) an exponential-time "global" optimization algorithm (e.g., Branch-and-Bound), or (ii) a heuristic algorithm for an approximate solution. The former is infeasible for all practical scenarios, due to the large problem size (K^2 control variables). Therefore, we will consider two heuristic approaches: in Section IV-A we consider a "myopic" algorithm, essentially a greedy approach that solves a simpler objective than Optimization Problem 1; this algorithm will be our baseline, as it resembles some recent state-of-the-art [26]); in Section IV-B, we propose a more sophisticated algorithm, inspired from ADMM type of schemes [37].

A. Myopic Algorithm

The non-convexity of Optimization Problem 1 is due to the expression of the stationary distribution $\boldsymbol{\pi}$ that appears in the objective function. As mentioned, the stationary distribution captures the long-term behavior of a system where users sequentially consume many contents. To simplify the objective, one can could consider a coarse approximation where the recommendation impact is there, but the algorithm "greedily" optimizes the access cost *only for the next content access*. In other words, it is as if a user initially requests a content i, then requests another content j (recommended or not), and then leaves the system. In this case, the objective becomes

$$(\mathbf{p_0}^T \cdot P) \cdot \mathbf{x}, \tag{7}$$

where the first term of Eq. (2) is dropped (because it is independent of the control variables), and we keep only the second term. This gives rise to the following optimization problem.

Optimization Problem 2 (Myopic Cache-Friendly Recommendations)**.**

$$\underset{Y}{\textit{minimize}} \quad \mathbf{p_0}^T \cdot (a \cdot Y + (1-a) \cdot P_0) \cdot x, \tag{8}$$
$$\textit{s.t.} \quad \textit{Eqs. (6a)–(6d)}$$

In the above problem, the constraints remain intact as the Problem 1. However, now the objective is linear in Y. This is an Linear Problem (LP) with affine and box constraints, which can be solved efficiently in polynomial time, using e.g. interior-point methods [36].

Remark. The single-step approach can be interpreted as a projection of the recent work of [26] to our framework. Specifically, the authors solve a similar "single-step" problem, jointly optimizing the caching and recommendation policy (which is formulated as a Knapsack problem). Omitting the caching decisions of [26], for the recommendations the authors solve a similar problem to Optimization Problem 2.

B. Cache-Aware Recommendations for Sequential content access (CARS)

The above "myopic" approach does not exploit the full structure of the Markov chain P. For example, assume there are two contents A and B that are both cached and both have high similarity with a content currently consumed, but B has slightly higher similarity. The Myopic scheme will choose to recommend B. However, assume that A is similar to many contents that happen to be cached, while B does not. This suggests that, if B is recommended, then in the next step there will be very few good options (hence the algorithm's name): the myopic algorithm will either have to recommend cached contents with low quality or high quality contents which lead to cache misses. To be able to foresee such situations and take the right decisions, we need to go back to the objective of Optimization Problem 1.

To circumvent the problem of having the inverse of the control matrix in the objective, we formulate an *equivalent* optimization problem by introducing the stationary vector $\boldsymbol{\pi}$ as an explicit ("auxiliary") control variable.

Optimization Problem 3 (Cache-Friendly Recommendations: Equivalent Problem)**.**

$$\underset{\boldsymbol{\pi},Y}{\textit{minimize}} \quad \boldsymbol{\pi}^T \cdot \mathbf{x}, \tag{9}$$

$$\textit{s.t.} \quad \textit{Eqs. (6a)–(6d)}$$

$$\boldsymbol{\pi}^T = \boldsymbol{\pi}^T \cdot (a \cdot Y + (1-a) \cdot \mathbf{p_0}^T) \tag{9a}$$

$$\sum_{j=1}^{K} \pi_j = 1 \tag{9b}$$

$$\pi_j \geq 0, \quad \forall\ j\ \in \mathcal{K}. \tag{9c}$$

Optimization Problem 3 constraints three new (sets of) constrains. Eq. (9b) and Eq. (9c) simply ensure that $\boldsymbol{\pi}$ is a probability distribution. Eq. (9a) is an important constraint that ensures that the two problems are equivalent, by forcing $\boldsymbol{\pi}$ to be a stationary distribution related to the transition matrix $P = a \cdot Y + (1-a) \cdot P_0$. It is easy to see that the two problems have the same set of optimal solutions.

The objective function is now linear in the control variables $\boldsymbol{\pi}$. However, constraint Eq. (9a) is a quadratic equality constraint, and thus the problem remains non-convex. Nevertheless, observe that the problem is now *bi-convex* in the variables Y and $\boldsymbol{\pi}$. Bi-convex problems can often be efficiently tackled with Alternating Convex Search (ACS) methods, that iteratively solve the convex sub-problems for each set of control variables. Unfortunately, such approaches fail here, as the Y subproblem is simply a feasibility problem (Y does not appear in the objective), and ACS would not converge (our implementation confirms this observation). What is more, having the quadratic equality constraint as a hard constraint does not facilitate such an iterative solution.

Instead, we propose to use a Lagrangian relaxation for that constraint, moving it to the objective. To ensure the strong convexity of the new objective, we form the *Augmented Lagrangian* [37]. Let us first define the function $c(\boldsymbol{\pi}, Y)$ as

$$c(\boldsymbol{\pi}, Y) = \boldsymbol{\pi}^T - \boldsymbol{\pi}^T \cdot (a \cdot Y - (1-a) \cdot P_0) \tag{10}$$

so that the constraints of Eq. (9a) can be written as

$$c(\boldsymbol{\pi}, Y) = 0 \tag{11}$$

The augmented Lagrangian is then given by:

$$f_\rho(\boldsymbol{\pi}, Y) = \boldsymbol{\pi}^T \cdot x + c(\boldsymbol{\pi}, Y) \cdot \boldsymbol{\lambda} + \frac{\rho}{2} \cdot (||c(\boldsymbol{\pi}, Y)||_2)^2 \tag{12}$$

where $\boldsymbol{\lambda}$ is the column vector of length K of the Lagrangian multipliers (one multiplier per quadratic equality), ρ a positive constant scalar, and $||\cdot||_2$ the euclidean norm. This objective is still subject to the remaining constraints of Optimization Problem 3, all of which are now affine. What is more, the problem remains bi-convex in the control variables Y and $\boldsymbol{\pi}$. We can thus apply an ADMM-like method, where we iteratively solve the convex subproblems with respect to Y and $\boldsymbol{\pi}$, but now with the above augmented objective, so that when $c(\boldsymbol{\pi}, Y)$ diverges a lot from 0, the subproblem solutions in the inner loop are penalized. We also update the Lagrangian multipliers λ_i at each iteration. Our detailed algorithm is described in Algorithm 1.

Algorithm 1 CARS (Cache-Aware Recommendations for Sequential content access)

```
    $Input: N, U, q, \mathbf{x}, a, \mathbf{p_0}$                    ▷ (system parameters)
    $Input: Acc_1, Acc_2, maxIter, \rho, \lambda_0, Y_0$
                                        ▷ (algorithm tuning parameters)
 1: $i \leftarrow 1$
 2: $COST_0 \leftarrow \infty$
 3: $V \leftarrow True$
 4: while $V$ do
 5:     $\boldsymbol{\pi}_i = \underset{\boldsymbol{\pi} \in C_\pi}{\text{argmin}}\{f_\rho(\boldsymbol{\pi}, Y_{i-1})\}$
 6:     $Y_i = \underset{Y \in C_Y}{\text{argmin}}\{f_\rho(\boldsymbol{\pi}_i, Y)\}$
 7:     $\lambda \leftarrow \lambda + (\frac{\rho}{2}) \cdot c(\boldsymbol{\pi}_i, Y_i)$
 8:     $COST_i \leftarrow (1-a) \cdot \mathbf{p_0^T} \cdot (I - a \cdot Y_i)^{-1} \cdot \mathbf{x}$
 9:     $\epsilon_1 \leftarrow (|c(\pi_i, Y_i)|_2)^2$
10:     $\epsilon_2 \leftarrow |COST_i - COST_{i-1}|$
11:     $V = ((\epsilon_1 > Acc_1) \wedge (\epsilon_2 > Acc_2)) \vee (i \leq maxIter)$
12:     $i \leftarrow i + 1$
13: end while
14: $j \leftarrow \underset{\ell=1,\ldots,i-1}{\text{argmax}}\ \{COST_\ell\}$
15: $return\ \ Y_j$
```

Algorithm 1 receives as input the system parameters $N, U, q, \mathbf{x}, a, \mathbf{p_0}$, and the desired accuracy levels and initialization parameters $Acc_1, Acc_2, maxIter, \rho, \lambda_0, Y_0$. It initializes the objective ($COST_0$) to infinity and starts an iteration for solving the convex subproblems (lines 4–13). In the first leg of the loop (line 5), the augmented Lagrangian $f_\rho(\boldsymbol{\pi}, Y)$ is minimized over π, considering as constant the variables Y (equal to their prior value). Then, considers the returned value of $\boldsymbol{\pi}$ from line 5 as constant and minimizes the Lagrangian over the variables Y. Both minimization sub-problems are convex and can be efficiently solved. The solution space of the sub-problems C_Y and C_π is given by Eqs. (6a)–(6d) and Eqs.(9b)–(9c), respectively. After calculating in line 8 the long

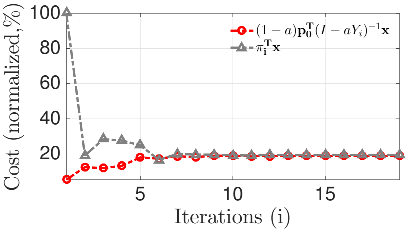


Fig. 1: Convergence of CARS.

term $COST$ we get from Y_i, the status of the current iteration is computed in the (a) primal residual of the problem (line 9) and (b) the difference of returned $COST$ compared to the previous step (line 10). The algorithm exits the while loop, when the value of the primal residual and improvement in the $COST$ are smaller than the required accuracy, or when the maximum allowable iterations are reached (as described in line 11).

As a final note, the above problem can also be cast into a non-convex QCQP (quadratically constrained quadratic program). State-of-the-art heuristic methods for approximate solving generic QCQP problems [38] are unfortunately of too high computational complexity for problems of this size. It is worth mentioning that we transformed the problem to a standard QCQP formulation and we applied methods based on [38] but the algorithms were only capable of solving small instances of the problem (a few 10s of contents).

Convergence of CARS. Finally, we investigate the performance of CARS (Algorithm 1) as a function of its computational cost, i.e., the maximum number of iterations needed. Fig. 1 shows the achieved *actual cost* (red line, circle markers) at each iteration, and the *virtual cost* (gray line, triangle markers) calculated from the current value of the auxiliary variable $\boldsymbol{\pi}$, in a simulation scenario (see details in Section V). It can be seen that within 5 iterations, CARS converges to its maximum achieved cache hit ratio. This is particularly important for cases with large content catalogue sizes that require an online implementation of CARS.

V. Performance Evaluation

In this section, we investigate the improvements in caching performance by the proposed cache-aware recommendation algorithm on top of a preselected caching allocation. We perform simulations using real datasets of related contents (movies and songs), collected from online databases. We first briefly present the datasets (Section V-A) and the simulation setup (Section V-B), and then present simulation results in a wide range of scenarios and parameters and discuss the main findings and implications (Section V-C)

A. Datasets

We collect two datasets that contain ratings about multimedia content. We use this information to build similarity matrices U, which are later used in the selection of recommendations, e.g., to satisfy a minimum recommendation quality q (as defined in Section III).

MovieLens. We use the $100k$ subset from the *latest Movielens* movies-rating dataset from the MovieLens website [39], containing 69162 ratings (from 0.5 to 5 stars) of 671 users for 9066 movies. To generate the matrix U of movie similarities from the raw information of user ratings, we apply a standard collaborative filtering method [40]. Specifically, we first apply an item-to-item collaborative filtering (using 10 most similar items) to predict the missing user ratings, and then use the cosine-distance ($\in [-1, 1]$) of each pair of contents based on their common ratings

$$sim(i,j) = \frac{\sum_{n=1}^{\#users} r_n(i)\cdot r_n(j)}{\sqrt{\sum_{n=1}^{\#users} r_n^2(i)}\cdot\sqrt{\sum_{n=1}^{\#users} r_n^2(j)}}$$

where we normalized the ratings r_i, by subtracting from each rating the average rating of that item. We build the matrix U by saturating to values above 0.6 to 1, and zero otherwise, so that $u_{nk} \in \{0,1\}$.

Last.fm. We use the subset of *The Million Song Dataset* from the Last.fm database [41], containing $10k$ song IDs. The dataset was built based on the method "getSimilar", and thus it contains a $K \times K$ matrix with the similarity scores (in [0,1]) between each pair of songs in the dataset, which we use as the matrix U. As the Last.fm dataset is quite sparse and we set the non zero values u_{ij} to one to make a binary U in that dataset as well.

To facilitate simulations, we process both datasets, by removing rows and columns of the respective U matrices with $\sum_{j\in\mathcal{K}} u_{ij} \le N$ (where number $N = 4$ is the number of total recommendations). After the preprocessing, we ended up with a content catalogue of size $K = 1060$ and $K = 757$ for MovieLens and Last.fm traces respectively.

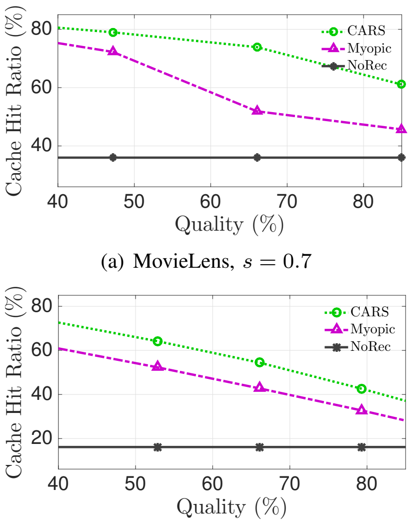
B. Simulation Setup

Content Demand. The users generate 40000 requests for contents in a catalogue $\mathcal{K}$; requests are either *direct* with probability $\mathbf{p_0} \sim Zipf(s)$ (s the exponent of the Zipf law) for any content, or *recommended* with probability $\frac{1}{N}$ for each of the recommended contents. We consider scenarios with exponent $s \in [0.4, 0.8]$ and $N = 4$. Unless otherwise stated, we set the default value $\alpha = 0.8$, similarly to the statistics in [23].

Caching Policy. We consider a popularity based caching policy, where the C most popular (w.r.t. $\mathbf{p_0}$) contents are locally cached in the base station. This policy is optimal in a single cache network, when no recommendation system is employed.

Recommendation policy. We simulate scenarios under the following three recommendation policies:

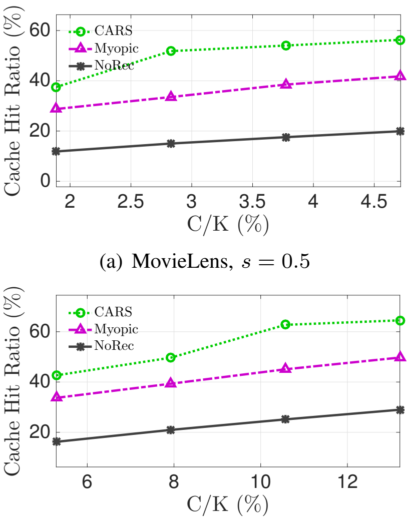
- *No Recommendation*: This is is also a baseline scenario, where users request contents only based on $\mathbf{p_0}$ (or, equivalently $a = 0$).
- *Myopic policy*: Cache-aware recommendations using the algorithm of Section IV-A, which optimizes recommendations



(a) MovieLens, $s = 0.7$

(b) Last.fm, $s = 0.4$

Fig. 2: Cache Hit Ratio vs Quality $N = 4,\ C/K = 5\%$.



(a) MovieLens, $s = 0.5$

(b) Last.fm, $s = 0.4$

Fig. 3: Cache Hit Ratio vs Relative Cache size, $Q = 80\%,\ N = 4$.

assuming single-step content requests. This policy relates to the previous works of [24], [26].

- *Proposed Policy/CARS*: Cache-aware recommendations using *CARS*, which optimizes recommendations for sequential content consumption.

C. Results

We compare the three recommendation policies in scenarios with varying q (minimum quality of recommendations - see Section III), cache size C, probability to request a recommended content a, and N recommended contents. For simplicity, we assume costs $x = 0$ for cached contents and $x = 1$ for non-cached. Hence, the cost becomes equivalent to the cache hit ratio ($CHR = (1 - a) \cdot \mathbf{p_0}^T \cdot (I - aY)^{-1} \cdot \mathbf{x}$), which we use as metric to measure the achieved performance in our simulations.

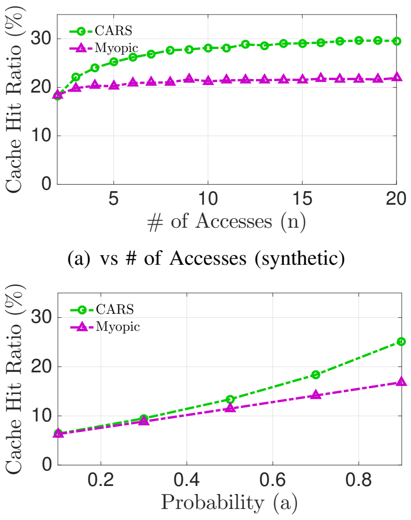
Impact of Quality of Recommendations. Recommending cached contents becomes trivial, if no quality in recommendations is required. However, the primary goal of a content provider is to satisfy its users, which translates to high quality recommendations. In the following, we present results that show that the proposed *CARS can always achieve a good trade-off between cache hit ratio and quality of recommendation, significantly outperforming baseline approaches*.

In Figures 2(a) and 2(b) we present the achieved cache hit ratio (y-axis) of the four recommendation policies for the MovieLens and Last.fm, datasets, respectively, in scenarios where the recommender quality is imposed to be above a predefined threshold q (x-axis). The first observation is that *Myopic* and *CARS* achieve their goal to increase the CHR compared to the baseline case of *NoRec*. The absolute gains for both policies increases for lower values of q, because for lower q there is more flexibility in recommendations. For high values of q, close to 100%, less recommendations that "show the cache" are allowed, and this leads to lower gains. However, even when the quality reaches almost 100%, the gains of *CARS* remain significant. In fact, the relative performance of *CARS* over the *Myopic* increases with q, which indicates that non-*Myopic* policies are more efficient when high recommendation quality is required.

Moreover, comparing Figures 2(a) and 2(b) reveals that the achievable gains depend also on the similarity matrix U. While in Fig. 2(b) both cache-aware recommendation policies follow a similar trend (for varying q), in Fig. 2(a) for the larger dataset of MovieLens, the performance of *CARS* decreases much less compared to *Myopic* with q.

Impact of Caching Capacity. In Figures 3(a) and 3(b) we investigate the performance of the recommendation policies with respect to the cache size, for a fixed value of the recommender quality q. The proposed algorithm, outperforms significantly the other two policies. For example, in Fig. 3(b), for $C/K = 8\%$ it achieves a 25% improvement over the *Myopic* algorithm. Even in the case of the MovieLens dataset (Fig. 3(a)), where the *Myopic* algorithm can only marginally improve the cache hit ratio, *CARS* still achieves significant gains. In total, in all scenarios we considered, *the relative caching gains from the proposed cache-aware recommendation policy (over the no-recommendation case) are consistent and even increase with the caching size.*

Impact of Sequential Content Consumption. *CARS* takes into account the fact that users consume more than one content sequentially, and optimizes recommendations based on this. On the contrary the *Myopic* algorithm (similarly to previous works [24], [26]) considers single content requests. Therefore, Algorithm 1 is expected to perform better as the average number of consecutive requests by a user increases. The simulation results in Fig. 4(a) validate this argument. We simulate a scenario of a small catalogue $K = 100, C = 4, N = 3, s = 0.6, q = 90\%$ and a U matrix with an $\overline{R} = 4$



(a) vs # of Accesses (synthetic)

(b) vs Probability (a) (synthetic)

Fig. 4: CHR for $N = 3$, (synthetic scenario)
(a) $q = 85\%,\ s = 0.2,\ C/K = 4\%$
(b) $q = 90\%,\ s = 0.6,\ C/K = 2.5\%$

related contents per content on average, where we vary the number of consecutive requests by each user. It can be seen that the *Myopic* algorithm increases the cache hit ratio when the users do a few consecutive requests (e.g., 3 or 4); after this point the cache hit ratio remains constant. However, under *CARS*, not only the increase in the cache hit ratio is higher, but it increases as the number of consecutive requests increase. This is a promising message for real content services (such as YouTube, Netflix, Spotify, etc.) where users tend to consume sequentially many contents.

Impact of Probability a. The probability a represents the *frequency* that a user follows a recommendation rather than requesting for an arbitrary content (restart probability, e.g., through the search bar in YouTube). The value of a indicates the influence of the recommendation system to users; in the cases of YouTube and Netflix it is approximately 0.5 and 0.8 respectively [22], [23]. In Fig. 4(b) we present the performance of the two cache-aware recommendation policies for varying values of a. The higher the value of a, the more frequently a user follows a recommendation, and thus the higher the gains from the cache-aware recommendation policies. However, while the gain from the *Myopic* algorithm increases linearly with a, the gains from the proposed *CARS* increase superlinearly. This is due to the fact that Algorithm 1 takes into account the effect of probability a when selecting the recommendations (e.g., see the objective function of Optimization Problem 3).

VI. Related Work

Mobile Edge Caching. Deploying small cells (SCs) over the existing macro-cell networks infrastructure, has been extensively studied and is considered a promising solution that could handle the existing and predicted massive data demands [42], [43], [44]. However, this densification of the cellular network will undoubtedly impose heavier load to the backhaul network. Taking advantage of the skewness in traffic demand, it has been suggested that caching popular content at the "edge" of the network, at SCs [10], user devices [14], [13], [15], or vehicles [16], [17] can significantly relieve the backhaul. Our work proposes an orthogonal and complementary approach for increasing the caching efficiency. We modify the recommendation algorithm to point the users towards the cached content, when this is possible and satisfies the quality of user experience. This can bring further gains in cache hit ratio, on top of existing caching algorithms/architectures.

Caching and Recommendations. The interplay between recommendation systems and caching has been only recently considered in literature, e.g., for peer-to-peer networks [45], CDNs [46], [47], or mobile/cellular networks [24], [26], [27], [25]. Closer to our study, are the works in [47], [26], [27], [24] that consider the promotion/recommendation of contents towards maximizing the probability of hitting a local cache. Leveraging the high influence of YouTube recommendations to users, the authors of [47] propose a reordering method for the *related list* of videos; despite its simplicity, this method is shown to improve the efficiency of CDNs. [26] considers the joint problem of caching and recommendations, and proposes a heuristic algorithm that initially places contents in a cache (based on content relations) and then recommends contents to users (based on cached contents). At the selection of the recommendations, [26] considers a single request per user, whereas our work considers a sequential content consumption model, which is closer to user behavior in services such as YouTube, Netflix, Spotify, etc. Similarly to [26], in [27], a single access user is considered. The caching policy in [27] is based on machine learning techniques, the users' *behavior* is estimated through the users' interaction with the recommendations and this knowledge is being exploited at the next BS cache updates. Finally, [24] studies the problem of recommendation-aware caching (in contrast to cache-aware recommendations in this paper). Assuming a content provider/service that offers alternative content recommendation or delivery, [24] proposes near-optimal approximation algorithms for content placement in mobile networks with single-cell and multi-cell (e.g., similarly to [10]) user association.

VII. Conclusions

In this paper we studied the problem of cache-friendly content recommendations in the context of mobile edge caching. We first introduced a model for sequential content requests over a recommendation system, which captures the user behavior in popular services such as YouTube, Netflix, Spotify, etc. Then, we proposed *CARS*, a cache-aware recommendation algorithm that can increase the caching efficiency (for the network operator), without losing in quality of recommendations (for the user). Our simulation results showed that *CARS* outperforms methods that do not take into account sequential content consumption. A promising future research direction work is to consider the joint problem of caching and recommendations –under sequential requests– in order to fully exploit the potential of modern communication networks.

References

[1] E. Bastug, M. Bennis, and M. Debbah, "Living on the edge: The role of proactive caching in 5g wireless networks," *IEEE Communications Magazine*, vol. 52, no. 8, pp. 82–89, 2014.

[2] C. V. networking Index, "Forecast and methodology, 2016-2021, white paper," *San Jose, CA, USA*, vol. 1, 2016.

[3] "What is 5g and why should lawmakers care?." https://www.washingtonpost.com/news/innovations/wp/2015/10/26/what-is-5g-and-why-should-lawmakers-care/?utm_term=.b66c4efb89b6.

[4] P. Gill, M. Arlitt, Z. Li, and A. Mahanti, "Youtube traffic characterization: A view from the edge," in *Proc. ACM IMC*, 2007.

[5] J. Erman, A. Gerber, K. K. Ramadrishnan, S. Sen, and O. Spatscheck, "Over the top video: The gorilla in cellular networks," in *Proc. ACM IMC*, 2011.

[6] Z. Li, J. Lin, M.-I. Akodjenou, G. Xie, M. A. Kaafar, Y. Jin, and G. Peng, "Watching videos from everywhere: A study of the pptv mobile vod system," in *Proc. ACM IMC*, 2012.

[7] M. Leconte, G. Paschos, L. Gkatzikis, M. Draief, S. Vassilaras, and S. Chouvardas, "Placing dynamic content in caches with small population," in *Proc. IEEE Infocom*, 2016.

[8] S. Elayoubi and J. Roberts, "Performance and cost effectiveness of caching in mobile access networks," in *Proceedings of International Conference on Information-Centric Networking, ICN '15, San Francisco, USA,*, 2015.

[9] https://en.wikipedia.org/wiki/List_of_most_viewed_online_videos_in_the_first_24_hours, accessed Feb 2018.

[10] N. Golrezaei, K. Shanmugam, A. G. Dimakis, A. F. Molisch, and G. Caire, "Femtocaching: Wireless video content delivery through distributed caching helpers," in *Proc. IEEE INFOCOM*, 2012.

[11] 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.

[12] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *IEEE Trans. on Information Theory*, vol. 60, pp. 2856–2867, May 2014.

[13] B. Han, P. Hui, V. S. A. Kumar, M. V. Marathe, J. Shao, and A. Srinivasan, "Mobile data offloading through opportunistic communications and social participation," *IEEE Trans. on Mobile Computing*, 2012.

[14] P. Sermpezis and T. Spyropoulos, "Effects of content popularity on the performance of content-centric opportunistic networking: An analytical approach and applications," *IEEE/ACM Transactions on Networking*, vol. 24, no. 6, pp. 3354–3368, 2016.

[15] P. Sermpezis and T. Spyropoulos, "Offloading on the edge: Performance and cost analysis of local data storage and offloading in HetNets," in *Proc. IEEE WONS*, pp. 49–56, 2017.

[16] J. Whitbeck, M. Amorim, Y. Lopez, J. Leguay, and V. Conan, "Relieving the wireless infrastructure: When opportunistic networks meet guaranteed delays," in *Proc. IEEE WoWMoM*, 2011.

[17] L. Vigneri, T. Spyropoulos, and C. Barakat, "Storage on Wheels: Offloading Popular Contents Through a Vehicular Cloud," in *Proc. IEEE WoWMoM*, 2016.

[18] T. X. Tran, P. Pandey, A. Hajisami, and D. Pompili, "Collaborative multi-bitrate video caching and processing in mobile-edge computing networks," in *Proc. IEEE WONS*, pp. 165–172, 2017.

[19] K. Shanmugam, M. Ji, A. M. Tulino, J. Llorca, and A. G. Dimakis, "Finite-length analysis of caching-aided coded multicasting," *IEEE Transactions on Information Theory*, vol. 62, no. 10, pp. 5524–5537, 2016.

[20] X. Cheng and J. Liu, "Nettube: Exploring social networks for peer-to-peer short video sharing," in *Infocom 2009, Ieee*, pp. 1152–1160, IEEE, 2009.

[21] S. Gupta and S. Moharir, "Modeling request patterns in vod services with recommendation systems," in *International Conference on Communication Systems and Networks*, pp. 307–334, Springer, 2017.

[22] R. Zhou, S. Khemmarat, and L. Gao, "The impact of youtube recommendation system on video views," in *In Proc. of ACM IMC 2010*.

[23] 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.

[24] P. Sermpezis, T. Spyropoulos, L. Vigneri, and T. Giannakas, "Femtocaching with soft cache hits: Improving performance with related content recommendation," in *Proc. IEEE GLOBECOM*, 2017.

[25] 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.

[26] L. E. Chatzieleftheriou, M. Karaliopoulos, and I. Koutsopoulos, "Caching-aware recommendations: Nudging user preferences towards better caching performance," in *Proc. IEEE INFOCOM*, 2017.

[27] D. Liu and C. Yang, "A learning-based approach to joint content caching and recommendation at base stations," *arXiv preprint arXiv:1802.01414*, 2018.

[28] "Google spells out how YouTube is coming after TV." http://www.businessinsider.fr/us/google-q2-earnings-call-youtube-vs-tv-2015-7/.

[29] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th international conference on World Wide Web*, pp. 285–295, ACM, 2001.

[30] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, 2009.

[31] P. Covington, J. Adams, and E. Sargin, "Deep neural networks for youtube recommendations," in *Proceedings of the 10th ACM Conference on Recommender Systems*, pp. 191–198, ACM, 2016.

[32] S. Borst, V. Gupta, and A. Walid, "Distributed caching algorithms for content distribution networks," in *Proc. IEEE INFOCOM*, pp. 1–9, 2010.

[33] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web.," tech. rep., Stanford InfoLab, 1999.

[34] M. Harchol-Balter, *Performance Modeling and Design of Computer Systems: Queueing Theory in Action*. Cambridge University Press, 2013.

[35] K. Avrachenkov and D. Lebedev, "Pagerank of scale-free growing networks," *Internet Mathematics*, vol. 3, no. 2, pp. 207–231, 2006.

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

[37] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, *et al.*, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[38] J. Park and S. Boyd, "General heuristics for nonconvex quadratically constrained quadratic programming," *arXiv preprint arXiv:1703.07870*, 2017.

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

[40] X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," *Adv. in Artif. Intell.*, pp. 4:2–4:2, 2009.

[41] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere, "The million song dataset," in *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011.

[42] A. Ghosh, N. Mangalvedhe, R. Ratasuk, B. Mondal, M. Cudak, E. Visotsky, T. A. Thomas, J. G. Andrews, P. Xia, H. S. Jo, H. S. Dhillon, and T. D. Novlan, "Heterogeneous cellular networks: From theory to practice," *IEEE Comm. Magazine*, vol. 50, no. 6, pp. 54–64, 2012.

[43] J. G. Andrews, H. Claussen, M. Dohler, S. Rangan, and M. C. Reed, "Femtocells: Past, present, and future," *IEEE Journal on Selected Areas in Communications*, vol. 30, no. 3, pp. 497–508, 2012.

[44] J. G. Andrews, "Seven ways that hetnets are a cellular paradigm shift," *IEEE Communications Magazine*, vol. 51, no. 3, pp. 136–144, 2013.

[45] D. Munaro, C. Delgado, and D. S. Menasché, "Content recommendation and service costs in swarming systems," in *Proc. IEEE ICC*, 2015.

[46] D. K. Krishnappa, M. Zink, and C. Griwodz, "What should you cache?: a global analysis on youtube related video caching," in *Procc. ACM NOSSDAV Workshop*, pp. 31–36, 2013.

[47] 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.