The Order of Things: Position-Aware Network-friendly Recommendations in Long Viewing Sessions

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

Abstract—Caching has recently attracted a lot of attention in the wireless communications community, as a means to cope with the increasing number of users consuming web content from mobile devices. Caching offers an opportunity for a win-win scenario: nearby content can improve the video streaming experience for the user, and free up valuable network resources for the operator. At the same time, recent works have shown that recommendations of popular content apps are responsible for a significant percentage of users requests. As a result, some very recent works have considered how to nudge recommendations to facilitate the network (e.g., increase cache hit rates). In this paper, we follow up on this line of work, and consider the problem of designing cache friendly recommendations for long viewing sessions; specifically, we attempt to answer two open questions in this context: (i) given that recommendation position affects user click rates, what is the impact on the performance of such network-friendly recommender solutions? (ii) can the resulting optimization problems be solved efficiently, when considering both sequences of dependent accesses (e.g., YouTube) and position preference? To this end, we propose a stochastic model that incorporates position-aware recommendations into a Markovian traversal model of the content catalog, and derive the average cost of a user session using absorbing Markov chain theory. We then formulate the optimization problem, and after a careful sequence of equivalent transformations show that it has a linear program equivalent and thus can be solved efficiently. Finally, we use a range of real datasets we collected to investigate the impact of position preference in recommendations on the proposed optimal algorithm. Our results suggest more than 30\% improvement with respect to state-of-the-art methods.

I. INTRODUCTION

Storing content close to wireless users is recognized as a promising method to (i) reduce the network cost to serve a request, and (ii) improve user experience (e.g., better playout quality). As a result, a number of studies suggest to install tiny caches (e.g., hard drives) at every small-cell or femto-node [1], bringing ideas from hierarchical caching [2] into the wireless domain.

Nevertheless, the rapidly growing catalog sizes, smaller sizes per cache (e.g., at femto-nodes or user devices) compared to traditional CDNs, and volatility of user demand when considering smaller populations, make the task of caching algorithms increasingly challenging [3], [4]. For example, installing say 1TB in every small cell in an ultra-dense network (already a pretty expensive investment) would still fit less 1\% or less of the content catalogue of even one provider (e.g., the Netflix catalogue is reportedly in the order of few PBs). Things are even more stringent for UE-side caching solutions [5], where it is reported that up to 10-20 files could be pre-fetched realistically [6].

To overcome such challenges, a radical approach has been recently proposed [7], [8], [9], [10], [11], [12], based on the observation that user demand is increasingly driven today by recommendation systems of popular applications (e.g., Netflix, YouTube). Instead of simply recommending interesting content, recommendations could instead be “nudged” towards interesting content with low access cost (e.g., locally cached) [7], [12]; the recommendation quality remains unaltered, and the new content might in fact become accessible at better quality (e.g., HD). This idea is appealing, potentially presenting a win-win situation for all involved parties.

Nevertheless, due to the very recent research interest in the topic, a number of key questions remain unanswered. First, it has been shown that the users have the tendency to click on recommended contents (or products in the case of e-commerce) according to the position they find them, e.g., contents higher up in the recommendation list [13], [14]. However, several of the aforementioned studies tend to ignore this aspect [9], [11], [10] in their analysis, assuming that an equally good recommendation will be clicked equally frequently, regardless of the position in the application GUI that it appears. The work in [7], while taking into account the ranking of the recommendations in the modeling and their proposed algorithm, in the simulation section they assume that the boosting of the items is equal. So an interesting question arising then is: Does the performance of network-friendly recommendation schemes improve, deteriorates, or is unaffected by such position preference?

A second important question has to do with the computational complexity of optimizing network-friendly recommendations. In settings where each user requests one content (or equivalently requests many contents in an I.I.D. manner), the caching-side of the problem [11] or the recommendation-side of the problem, can be efficiently approximated. However, the joint caching and recommendation problem is NP complete [7], without any known approximation guarantees or optimal decompositions [7]. Things get worse, when one considers a user accessing multiple contents during a session in a structured manner, due to the inherent memory this system has (the content recommended and/or accessed at step $n$ has
an impact beyond step $n + 1$). Even without position preference, the problem of network-friendly recommendations for long (markovian) sequences of content accessed seems to be hard (non-convex) [9]. A second question of interest then is: Can the problem of network-friendly recommendations even be solved efficiently, in a context where there is both position preference and dependence in consecutive content requests?

To this end, in this paper we make the following contributions towards answering the above questions:

(i) Sequential request analysis based on absorbing Markov chain theory. We propose an analytical framework based on absorbing Markov chain theory, to model a user accessing a sequence of contents, driven by a recommender (Sections II and III). The sequential request model with preference to top recommendations better fits real users behavior in a number of popular applications (e.g. YouTube, Vimeo, Spotify) compared to Independent Reference Models (IRM) used in previous work [11] and/or models neglecting the position of recommendations [9], [11].

(ii) Optimal solution. We formulate a generic optimization problem for high quality but network-friendly recommendations. While the original problem in non-convex (similarly to previous formulations [9]), we prove an equivalent convex problem for high quality but network-friendly recommendations (Section V). Our findings reveal that the more skewed the preference towards top recommendations better fits the baseline recommender. W.l.o.g. we also assume $p_0$ governs the first content accessed, when a user starts a session. This model captures a number of everyday scenarios (e.g., watching clips on YouTube, personalized radio, etc).

The last point in the definition is a key differentiator of this work, compared to some previous ones on the topic [9], [7], [10]. A variety of recent studies [13], [14] has shown that the web-users have the tendency to click on contents (or products in the case of e-commerce) according to the position they find them. For example, in the PC interface of YouTube, they show a preference for the contents that are higher in the list of the recommended items. Hence, the probability of picking content in position 1 ($v_1$), might be quite higher than the probability to pick the content in position $N$ ($v_N$). In contrast, [9], [7], [10] explicitly or implicitly assume that $v_i = \frac{1}{N}, \forall i$.

Remark - Position Entropy: A key goal of this paper is to understand the additional impact of position preference on the achievable gains of network-friendly recommendations. A natural way to capture position preference is with the entropy of the probability mass function $v = [v_1, v_2, ..., v_N]$, namely

$$H_v = H(v_1, ..., v_N) = - \sum_{n=1}^{N} v_n \cdot \log(v_n). \quad (1)$$

The original case of no position preference, corresponds to a uniformly distributed $v$, which is well known to have maximum entropy. Any position preference will lead to lower entropy, with the extreme case of a “1-hot vector” (i.e., only one $v_i = 1$) having zero entropy.

Content Retrieval Cost. We assume that fetching content $i$ is associated with a generic cost $c_i \in \mathbb{R}, c = [c_1, c_2, ..., c_K]^T$, which is known to the content provider, and might depend on access latency, congestion overhead, or even monetary cost.

Maximizing cache hits: Can be captured by setting $c_i = 1$ for all cached content and to $c_i = 0$, for non-cached content.

Hierarchical caching: Can be captured by letting $c_i$ take values out of $n$ possible ones, corresponding to $n$ cache layers: higher values correspond to layers farther from the user [16], [2].

B. Baseline Recommendations.

Recommendation systems (RS) are an active area of research, with state-of-the-art RS using collaborative filtering [17], and recently deep neural networks [18]. For simplicity, we assume that the baseline RS works as follows:

1In fact, a Zipf-like relation has been observed [14].
**Definition 2 (Baseline Recommendations and Matrix U).**

(i) For every pair of contents \(i, j \in K\) a score \(u_{ij} \in [0, 1]\) is calculated, using a state-of-the-art method. Note that these scores can be personalized, and differ between users.2

(ii) After a user has just consumed content \(i\), the RS recommends contents according to these \(u_{ij}\) values (e.g., the \(N\) contents \(j\) with the highest \(u_{ij}\) value [14], [18]).3

C. Network-friendly Recommendations.

Our goal is to depart from the baseline recommendations (Def. 2) that are based only on \(U\), and let them consider the access costs \(c\) as well. We define recommendation decisions as follows.

**Definition 3 (Control Variables \(R^1, \ldots, R^N\)).** Let \(r_{ij}^n \in [0, 1]\) denote the probability that content \(j\) is recommended after a user watches content \(i\) in the position \(n\) of the list. For the \(n\)-th position in the recommendation list, these probabilities define a matrix \(K \times K\) recommendation matrix, which we call \(R^0\).

Defining recommendations as probabilities provides us more flexibility, as it allows to not always show a user the same contents (after consuming some content \(i\)). For example, assume \(K = 4\) total files, a user just watched item 1, and \(N = 2\) items must be recommended. Let the first row of the matrix \(R^1\) be \(r_{ij}^1 = [0, 1, 0, 0]\) and that of \(R^2\) be \(r_{ij}^2 = [0, 0, 0.5, 0.5]\). In practice, this means that in position 1 the user will always see content 2 being recommended (after consuming content 1), and the recommendation for position 2 will half the time be for content 3 and half for content 4.

Our objective is to choose what to recommend in which position, i.e., choose \(R^1, \ldots, R^N\), to minimize the average content access cost. However, we still need to ensure that the user remains generally happy with the quality of recommendations and does not abandon the streaming session.

**Recommendation Quality Constraint.**

Let \(r_{ij}^{(n)(B)}\) denote the baseline recommendations of Def .2. We can define the recommendation quality of this baseline recommender for content \(i\), \(q_{ij}^{max}\) as follows

\[
q_{ij}^{max} = \sum_{j=1}^{K} \sum_{n=1}^{N} v_n \cdot r_{ij}^{(n)(B)} \cdot u_{ij}.
\]

This quantity will act as another figure of merit for other (network-friendly) RS.

**Definition 4 (Quality of Network-Friendly Recommendations).** Any other (network-friendly) RS that differs from the

---

2\(u_{ij}\) could correspond to the cosine similarity between content \(i\) and \(j\), in a collaborative filtering system [17], or simply take values either 1 (for a small number of related files) and 0 (for unrelated ones). These scores might also depend on user preferences and past history of that user, as is often the case when users are logged into the app.

3\(N\) depends on the scenario. E.g., in YouTube \(N = 2, \ldots, 5\) in its mobile app, and \(N = 20\) in its website version.

---

Fig. 1: Comparison of baseline (left) and network-friendly (right) recommenders. Gray and white boxes denote cached and non-cached contents, respectively. Recommending after content 3 a slightly less similar content (i.e., content 4 instead of 6), leads to lower access cost in the long term.

**baseline recommendations** \(r_{ij}^0\) can be assessed in terms of its recommendation quality \(q \in [0, 1]\) with the constraint:

\[
\sum_{j=1}^{K} \sum_{n=1}^{N} v_n \cdot r_{ij}^n \cdot u_{ij} \geq q \cdot q_{ij}^{max}, \forall i \in K.
\]

where \(q_{ij}^{max}\) is the quantity defined in Eq.(2).

This equation weighs each recommendation with: (a) its quality \(u_{ij}\), and (b) the importance of the position \(n\) it appears at, \(v_n\). Note however that this constraint is not a restrictive choice. One could conceive a more “aggressive” recommender that removes the weight \(v_n\) from the left-hand side. In fact, our framework can handle any quality constraint(s) that are convex in \(r_{ij}^n\).

Based on the above discussion, a network-friendly recommender could favor at each step contents \(j\) (i.e., give high \(r_{ij}^n\) values) that have low access cost \(c_j\) but also are interesting to the user (i.e., have high \(u_{ij}\) value). However, as we show in later sections, such a greedy approach is suboptimal, as the impact of \(r_{ij}^n\) goes beyond the content \(j\) accessed next, affecting the entire *sample path* of subsequent contents in that session. The example in Fig. 1 depicts such a scenario: after content 3, instead of recommending content 6 (related value \(u_{36} = 1\)) content 4 is recommended (\(u_{34} = 0.8\)), because 4 is more related to cached contents (9 and 7) that can be recommended later (whereas 6 is related to the non-cached contents 5 and 4).4

**Remark on Recommendation Personalization.** As hinted at earlier, content utilities \(u_{ij}\) and recommendations \(r_{ij}^n\) can be user-specific (e.g. \(u_{ij}^n\) for user \(u\)), since different users might have different access patterns that can be leveraged. Nevertheless, to avoid notation clutter we do not use superscript \(u\) in the remainder of the paper, and will assume that these quantities and the respective optimization algorithm is done per user.

**Remark on Recommendation Quality.** Cache-friendly recommendations might also improve user QoE, in addition to network cost, a “win-win” situation. Today’s RS, measure their performance (QoR) without taking into account where the recommended content is stored. Assuming two contents equally interesting to the user where the one is stored locally while the other is not: it is obvious that the cached one could be streamed in much better quality (e.g., HD, so higher QoS), thus leading to \(q > 1\). Hence, more sophisticated QoE (= QoR + QoS) metrics could combine these effects: e.g., a content’s effective utility \(\hat{u}_{ij} = f(u_{ij}, c_j)\) that increases if \(j\) is highly

4The reason is that many contents \(j\) will have high enough relevance \(u_{ij}\) to the original content \(i\), and are thus interchangeable [14]
related to \( i \) but also if it is locally cached (i.e., \( c_j \) is low). Such a metric could be immediately integrated into our framework, simply by replacing \( u \) with \( \bar{u} \).

Table I summarizes some important notation. Vectors and matrices are denoted with bold symbols.

### III. Average Session Cost

Having defined the content access model, our first step towards “optimizing” the (network-friendly) recommendations, is to better understand what we are trying to optimize. To this end, in this section we derive the expected content access cost for a typical user session, as a function of recommendation variables \( r_{ij}^u \). This will serve as the objective of our problem. (Section IV).

#### Definition 5

Let \( S = \{i_1, i_2, \ldots, i_n\}, i_n \in \mathcal{K} \) be a sequence of contents accessed by a user according to Def. 1 during a viewing session. Then \( S \) is a discrete-time Markov process with transition matrix

\[
P = \alpha \sum_{n=1}^{N} v_n R^n + (1 - \alpha)1 \cdot p_0^T,
\]

where \( 1 = [1, 1, \ldots, 1]^T \) is a column vector of all 1s.

When the user has just consumed content \( i \), then she might next consume content \( j \) if all the following occur: she decides to follow a recommendation (probability \( \alpha \) according to Def. 1), \( j \) appears in the position \( n \) (probability \( r_{ij}^u \)), and she picks the content at the \( n \)-th position (probability \( v_n \)). These probabilities are by definition independent, hence the probability of these three events is their product, \( \alpha \cdot v_n \cdot r_{ij}^u \). Note that the user might consume \( j \), if she finds it in positions other than \( n \) (for example in position \( m \)) and will then click it with \( v_m \). Moreover, the user might also consume \( j \) after \( i \), if she ignores the recommendations (with probability \( 1 - \alpha \) according to Def. 1) and picks content \( j \) from the entire catalog (with probability \( p_j \)). Putting all these together gives the transition probability from \( i \) to \( j \), \( P_r(i \rightarrow j) = \alpha \cdot \sum_{n=1}^{N} v_n \cdot r_{ij}^n + (1 - \alpha) \cdot p_j \), which written in matrix notation gives Eq. (4).

#### Lemma 1 (Content Access as Renewal-Reward)

A content access sequence \( S = \{S^1_R, S^2_R, \ldots\} \) defines a renewal process, with subsequences \( S^i_R \), where the user follows recommended content, each ending with a jump outside of the recommender. The cost \( c_i \) incurred at each state is the reward.

#### Lemma 2 (Recommendation-Driven Cost)

The content access cost \( C(S_R) \) during a renewal cycle \( S_R \) is given by

\[
E[C(S_R)] = p_0^T \cdot G \cdot c,
\]

and the expected length of such a cycle is

\[
|S_R| = p_0^T \cdot G \cdot 1 = \frac{1}{1 - \alpha},
\]

where \( G = \left( I - \alpha \cdot \sum_{n=1}^{N} R^n \right)^{-1} \) is the Fundamental Matrix of an AMC with \( K \) transient states and 1 absorbing state, corresponding to a jump outside recommended states.

Proof. Let a user start a sub-sequence by retrieving content \( i \). The expected number of retrievals of content \( j \) (or, number of times visiting state \( j \)) until the end of the sub-sequence is given by \( g_{ij} \), where \( g_{ij} \) is the \((i\text{-row}, j\text{-column})\) element of the fundamental matrix \( G \) of the AMC [19].

The fundamental matrix is defined as

\[
G = \sum_{n=0}^{\infty} Q^n = (I - Q)^{-1}
\]

where \( Q \) the matrix with the transition probabilities \( q_{ij} \) between the transient states of the AMC \((i, j) \in \mathcal{K})\). Following the same arguments as in Def. 5, we get that \( g_{ij} = \alpha \cdot \sum_{n=1}^{N} v_n \cdot r_{ij}^n \), or, in a matrix format \( Q = \alpha \cdot \sum_{n=1}^{N} v_n \cdot R^n \). Substituting this into Eq. (7) gives the expression for \( G \) that appears in Lemma 2. Now, the total cost of retrieving a content \( j \) is \( c_j \). Since each content \( j \) is retrieved on average \( g_{ij} \) times during a sub-sequence that starts from \( i \), the total cost is given by

\[
E[C(S_R) | i] = \sum_{j \in \mathcal{K}} g_{ij} \cdot c_j
\]

The probability that a sub-sequence starts at content \( i \) is equal for all sub-sessions and is given by \( p_i \). Thus, taking the expectation over all the possible initial states \( i \), gives Eq. (9).

\[
E[C(S_R)] = \sum_{i \in \mathcal{K}} E[C(S_R) | i] \cdot p_i = \sum_{i \in \mathcal{K}} \sum_{j \in \mathcal{K}} g_{ij} \cdot c_j \cdot p_i
\]

Expressing the above summation as the product of the vectors \( p_0 \) and \( c \), and the matrix \( G \), gives Eq. (5). Similarly, if \( g_{ij} \) is the amount of time spent on state \( j \) before absorption, starting from state \( i \), then \( \sum_j g_{ij} \) must be
equal to the total time spent at any state before absorption. Weighing this with the probability \( p_i \) of starting at each state \( i \), gives the expected time to absorption, which is the expected duration of a sub-sequence \( E[\|S_n\|] = \sum_i p_i \sum_j g_{ij} \). Writing this in matrix notation, gives the first part of Eq.(6).

However, observe that the probability of absorption at any state \( i \) is equal to \( 1 - \alpha \), independent of \( i \). Hence, the number of steps till absorption is a geometric random variable with parameter \( 1 - \alpha \), and thus the mean time (i.e., number of steps) to absorption is \( \frac{1}{1-\alpha} \). □

The following Theorem, which gives the expected retrieval cost for a user session, follows immediately from Lemmas 1, 2, and the Renewal-Reward theorem [20]

**Theorem 1.** The expected retrieval cost per content, for a user session \( S \), given a recommendation matrix \( R \) is

\[
E[C(S) \mid R^{1}, \ldots, R^{N}] = \frac{\mathbf{p}_0^T \left( I - \alpha \sum_{n=1}^{N} v_n R^n \right)^{-1} \mathbf{c}}{1 - \alpha} \tag{10}
\]

### IV. OPTIMIZATION PROBLEM AND METHODOLOGY

In this section, we use the results of the previous section to formulate the problem of minimizing the expected access cost until absorption under a set of modeling constraints.

**A. The Problem and its Constraints**

**OP 1 (Nonconvex formulation).**

\[
\begin{align*}
\text{minimize} & \quad \mathbf{p}_0^T \left( I - \alpha \sum_{n=1}^{N} v_n \cdot R^n \right)^{-1} \cdot \mathbf{c} \quad \tag{11} \\
\text{subject to} & \quad \sum_{j=1}^{K} \sum_{n=1}^{N} v_n \cdot r_{ij} \cdot u_{ij} \geq q \cdot q_{\text{max}}^n, \quad \forall i \in K \\
& \quad \sum_{j=1}^{K} r_{ij} = 1, \quad \forall i \in K \text{ and } n = 1, \ldots, N \quad \tag{12} \\
& \quad \sum_{n=1}^{N} r_{ij} \leq 1, \quad \forall \{i, j\} \in K \quad \tag{13} \\
& \quad 0 \leq r_{ij} \leq 1 \ (i \neq j), \quad r_{ii} = 0 \ \forall \ i, n. \quad \tag{14}
\end{align*}
\]

The constraint in Eq.(12), is responsible for keeping the quality of the recommendations above a pre-specified (and given) threshold. The pair of constraints in Eqs.(13,15), defines a probability simplex for every row of all the \( R^n \) matrices. Note that we also prohibit self-recommendations (\( r_{ii}^n = 0 \ \forall \ i \) and \( n \)) (see Eq.(15)). Importantly, Eq.(14) is necessary in the position-aware setup, to ensure that the same content will not be recommended in two different positions. As an example assume that \( r_{i1}^1 = [0, 1, 0, 0] \) and \( r_{i1}^2 = [0.2, 0.3, 0.5] \); in that case we clearly see that content 2 would always be shown in position 1 (after watching content 1), but 20% of those times it would be shown in position 2 as well. Hence, Eq.(14) ensures that such decision vectors would be infeasible.

Evidently, our feasible space consists of either linear (equalities or inequalities) or box constraints with respect to the decision variables \( r_{ij}^n \). However, the objective is non-convex in general.

**Lemma 3.** The problem described in OP 1 is nonconvex.

**Proof.** The problem OP 1 comprises \( N \cdot K^2 \) variables \( r_{ij}^n \), and a set of \( K^2 \cdot (N + 2) + K \) linear (equality and inequality) constraints, thus the feasible solution space is convex. However, assume w.l.o.g that \( p_0 = c = w \), \( N = 1 \), and \( r_{11} = 1 \); the objective now becomes \( f(R) = w^T (I - \alpha \cdot R)^{-1} w \). Unless \( R \) is symmetric positive semi-definite (PSD), \( f(R) \) is non-convex [21]. Forcing \( R \) to be symmetric would require additional constraints that lead to suboptimal solutions of this problem [22]. Therefore, our objective as is, is nonconvex and there are no exact methods that can solve it in polynomial time. □

**B. The Journey to Optimality**

In addition to non-convexity, a key difficulty in solving OP 1 is the inverse matrix in the objective. Any gradient-based algorithm would require a matrix inversion at each gradient step (an operation of complexity \( O(K^3) \). To circumvent this, we introduce \( K \) auxiliary variables \( z^T \), for which we will demand \( z^T = \mathbf{p}_0^T \left( I - \alpha \sum_{n=1}^{N} v_n R^n \right)^{-1} \). This introduces \( K \) new equality constraints, leading to the following equivalent problem.\(^5\)

**Intermediate Step (Equivalent formulation).**

\[
\begin{align*}
\text{minimize} & \quad c^T \cdot z, \quad \tag{16} \\
\text{subject to} & \quad z^T - \alpha \cdot z^T \cdot \sum_{n=1}^{N} v_n \cdot R^n = \mathbf{p}_0^T \\
& \quad \text{Eqs. (12, 13, 14, 15)} \quad \tag{17}
\end{align*}
\]

The new objective is now convex (in fact, linear) in the new variable \( z \). However, as the set of constraints Eq.(17) are all quadratic equalities, the problem remains nonconvex. The above formulation falls under the umbrella of non-convex QCQP (Quadratically Constrained Quadratic Program), where it is common to perform a convex relaxation of the quadratic constraints, and then solve an approximate convex problem (e.g., SDP or Spectral relaxation, see [23] for more details). The problem can also be seen as bi-convex in variables \( R^n \) and \( z \), respectively. Alternating Direction Method of Multipliers (ADMM) can be applied to such problems, iteratively solving convex subproblems [24], [9]. Nevertheless, none of these methods provides any optimality guarantees, and even convergence for non-convex ADMM is an open research topic.

To further deal with this additional complication, we define another set of variables as \( f_{ij}^n = z_i \cdot r_{ij}^n \). Since the \( j \)-th element of the \( n \)-th vector \( z^T \cdot R^n \) can be written as \( \sum_j z_i \cdot r_{ij}^n \), we can write now \( z^T \cdot R^n = I^T \cdot F^n \), and the new variables are \( z \) and \( F^1, \ldots, F^N \), which are a \( K \times 1 \) vector, and \( N \times K \) matrices respectively.

This new transformation leads to the following problem.

\(^5\)Two problems are equivalent if the solution of the one, can be uniquely obtained through the solution of the other [21]; introducing auxiliary variables preserves the property.
OP 2 (LP formulation).

\[
\begin{align*}
\text{minimize} \quad & e^T \cdot z, \\
\text{subject to} \quad & \sum_{j=1}^{K} \sum_{n=1}^{N} v_n \cdot f^n_{ij} \cdot u_{ij} \geq z_i \cdot q \cdot d_i^{\max}, \quad \forall \ i \in \mathcal{K} \\
& \sum_{j=1}^{K} f^n_{ij} = z_i, \quad \forall \ i \in \mathcal{K} \text{ and } n = 1, \ldots, N \\
& \sum_{n=1}^{N} f^n_{ij} \leq z_i, \quad \forall \ {i, j} \in \mathcal{K} \\
& f^n_{ij} \geq 0 \quad (i \neq j), \quad f^n_{ii} = 0, \quad \forall \ i, j \in \mathcal{K} \\
& z_j - \alpha \cdot \sum_{n=1}^{N} v_n \cdot \sum_{i=1}^{K} f^n_{ij} = p_j, \quad \forall j \in \mathcal{K} \\
\end{align*}
\]

Lemma 4. The change of variables \( f^n_{ij} = z_i \cdot r^n_{ij}, \) is a bijection (one-to-one mapping) between \((z_i, r^n_{ij})\) and \((z_i, f^n_{ij})\).

Proof. This follows immediately, as we can readily obtain \( r^n_{ij} = \frac{z_i}{f^n_{ij}} \) from \((z_i, f^n_{ij})\). Note that, since \( z_j = \sum_{i=1}^{K} f^n_{ij} + p_j, \) and \( p_i \in (0,1) \quad \forall \ i, \) i.e. nonzero (see Def. 1), this forces \( z > 0 \) and thus \( r^n_{ij} \) are always uniquely defined. \( \square \)

Corollary. OP 1 can be solved efficiently as an LP.

Proof. Equivalency due to Lemma 4. \( \square \)

We have therefore transformed the nonconvex OP 1 to a convex (LP) one OP 2, and can now solve it optimally.

C. A Myopic Approach

A natural way to tackle the OP 1 is to try minimizing the cost of content retrieval in a single-content session (i.e., only one transition in the Markov chain). This is equivalent to minimizing the scalar quantity

\[
p_0^T \cdot (\alpha \cdot \sum_{n=1}^{N} v_n \cdot R^n + (1 - \alpha) \cdot 1^T \cdot p_0) \cdot c
\]

Ignoring the terms that do not depend on the control variables \( R^n \), yields the following.

OP 3 (Greedy Aware Recommendations).

\[
\begin{align*}
\text{minimize} \quad & p_0^T \cdot \left( \sum_{n=1}^{N} v_n \cdot R^n \right) \cdot c, \\
\text{subject to} \quad & \text{Eqs. (12, 13, 14, 15)}
\end{align*}
\]

Unlike the multi-step problem, this is already an LP, and can be solved directly without the earlier transformation steps. This solution of OP 3 will serve in the upcoming results section as a baseline approach, to solving the hard basis problem OP 1. Interestingly, the solution of OP 3 (we will call Greedy from now), resembles the policies proposed in [7], [8]. Although the algorithm of [7] targets a different context, i.e., the joint caching and single access content recommendation, the Greedy algorithm could be interpreted as applying the recommendation part of [7] for each user, along with a continuous relaxation of the control (recommendation) variables. In doing so, the recommendation problem is simply an LP of the type of Eq. (26), when the recommendations are allowed to be probabilistic. Due to this relaxation, the greedy algorithm is an upper bound for [7], looking at the recommendation problem only.

V. Validation Results

A. Warm Up

In this section we evaluate the performance of the proposed algorithm and provide insights regarding the behavior of the network-friendly recommendations schemes. For a realistic evaluation, we use three collected datasets from video/audio services. Before diving into the details, we need to state the following

Performance metric: Cache Hit Rate (CHR), as computed by the objective of Eq.(10), here we will minimize the cache miss.

Relative Gain: computed as \( \frac{CHR_{(proposed)}}{CHR_{(baseline)}} \cdot 100\% \), \( p_0: \) drawn from Zipf [25] of parameter \( s \), \( \nu: \) drawn from Zipf [14] of parameter \( \beta \), \( \alpha: \) will vary from 0.7 to 0.8, c: \( c_i = 0 \) for the C (cache capacity) most popular contents according to \( p_0 \), and 1 to the rest.

Solving OP 2, OP 3: carried out using IBM ILOG CPLEX in Python. We note that since CPLEX is designed to receive LPs in the standard form, we had to vectorize our matrices in order to bring the problem in the format \( \min_{x \geq 0, A \cdot x \leq b} \{ e^T \cdot x \} \) with linear and bound constraints over the variables. Regarding OP 3, it is easy to see that the problem’s objective Eq.(26) decomposes into \( K \) independent minimization problems, of size \( N \cdot K \) each, as the variables per content \( i \) are not coupled. Finally note that for the simulations in Figs. 3, 4, we will quote the cache-hit rate \textit{without} recommendations for reference, (i.e. storing the most popular contents that fit in the cache \( C \), based on \( p_0 \)) and, which we denote as MPH (Most Popular Hit - No Recommendations). This information along with the simulation parameters are included in Table II.

B. Schemes we compare with

We refer to our algorithm (OP 2) as Optimal.

Greedy Aware: We consider as baseline algorithm for network-friendly recommendations (OP 3 [7]), which is a position-aware scheme, but does not take into account that requests are sequential.

CARS: algorithm [9], a position-unaware scheme for sequential content requests proposed in, will serve as our second baseline. The CARS algorithm optimizes (with no guarantees) the recommendations for a user performing multiple sequential requests, but assumes that the user selects \textit{uniformly} one of the recommendations regardless of the position they appear.

Note on CARS. In our framework, this translates to solving OP 1 for uniform \( v \). The algorithm will then return \( N \) identical stochastic recommendation matrices. Importantly, whichever \( v \) we choose, the parenthesis of the Eq.(11) will be \( (I - \alpha \cdot \{v_1 \cdot R + \ldots + v_N \cdot R \}) = (I - \alpha \cdot R) \). This explains why the hit rate of CARS in the plots, remains constant regardless of the click distribution \( v \).
C. Datasets

**YouTube FR.** \( (K = 1054) \) We used the crawling methodology of [12] and collected a dataset from YouTube in France. We considered 11 of the most popular videos on a given day, and did a breadth-first-search (up to depth 2) on the lists of related videos (max 50 per video) offered by the YouTube API. We built the matrix \( U \in \{0, 1\} \) from the collected video relations.

**last.fm.** \( (K = 757) \) We considered a dataset from the last.fm database [26]. We applied the “getSimilar” method to the content IDs’ to fill the entries of the matrix \( U \) with similarity scores in \([0,1]\). We then set scores above 0.1 to \( u_{ij} = 1 \) to obtain a dense \( U \) matrix.

**MovieLens.** \( (K = 1066) \) We consider the MovieLens movies-rating dataset [27], containing 69162 ratings (0 to 5 stars) of 671 users for 9066 movies. We apply an item-to-item collaborative filtering (using 10 most similar items) to extract the missing user ratings, and then use the cosine distance (maximum entropy). In our simulations, we have used a zipf distribution \([14]\) over the positions and by decreasing its exponent, the entropy on the \( -ax \)-is axis is increased. As an example, in Fig. 3(a), lowest \( H_v \) corresponds to a vector of probabilities \( v = [0.8, 0.2] \) (recall that \( N = 2 \)), while the highest one on the same plot to \( v = [0.58, 0.42] \).

**Observation 1.** Our first observation is that the lower the entropy, the higher the optimal result. In the extreme case where the \( H_v \rightarrow 0 \) (virtually this would mean \( N = 1 \), the user clicks deterministically), the optimal hit rate becomes maximum. This can be validated in Fig. 5(b), where for increasing entropy the hit rate decreases and its max is attained for \( N = 1 \).

**Optimal vs Greedy.** The second question we study is: *How would a simpler greedy/myopic, yet position-aware, algorithm fare against our proposed method?* Fundamentally, the Greedy algorithm solves a less constrained problem than OP 1, and is therefore a more lightweight option in terms of execution time. However, the merits of using the proposed optimal method are noticeable in Figs. 4(a), 4(b) (parameters in Table II). In all three datasets, we see an impressive improvement, between 20 – 60%.

**Observation 2.** The constant relative gain of the two aware algorithms hints that both, as the entropy increases, seem to do the right placement in the positions. However, as Greedy decides with a small horizon, it cannot build the correct long paths that lead to higher gains in the following requests (clicks) of the user.

Lastly, we investigate the sensitivity of the three methods against the number of recommendations \( N \). In Fig. 5(b), we present the CHR curves of all three schemes for increasing \( N \), where we keep constant the distribution \( v \sim \text{zipf}(0.9) \). As expected, for \( N = 1 \) (e.g., YouTube autoplay scenario) CARS and the proposed scheme coincide, as there is no flexibility in having only one recommendation. However, as \( N \) increases, CARS and Greedy decay at a much faster pace than the proposed scheme, which is more resilient to the increase of \( N \). This leads to the following observation.

**Observation 3.** For large \( N \), CARS may offer the “correct” recommendations (cached or related or both), but it cannot place them in the right positions, as there are now too many available spots. In contrast, our algorithm Optimal recommends the “correct” contents, and places the recommendations in the “correct” positions. Fig. 5(a), strengthens

---

**TABLE II: Parameters of the simulation**

<table>
<thead>
<tr>
<th>Dataset</th>
<th>( q % )</th>
<th>( \text{zipf}(s) )</th>
<th>( \alpha )</th>
<th>( N )</th>
<th>MPH %</th>
</tr>
</thead>
<tbody>
<tr>
<td>MovieLens</td>
<td>80</td>
<td>0.8</td>
<td>0.7</td>
<td>2</td>
<td>23.26</td>
</tr>
<tr>
<td>YouTube FR</td>
<td>95</td>
<td>0.6</td>
<td>0.8</td>
<td>2</td>
<td>12.17</td>
</tr>
<tr>
<td>last.fm</td>
<td>80</td>
<td>0.6</td>
<td>0.7</td>
<td>3</td>
<td>11.74</td>
</tr>
</tbody>
</table>

---

Fig. 3: Cache Hit Rate vs \( H_v (C/K \approx 1.00\%) \) for Greedy Aware.

---

Fig. 4: Cache Hit Rate vs \( H_v (C/K \approx 1.00\%) \) for Greedy Aware.

---

Fig. 5: (a) Relative Gain vs \( (N, \beta) \) and (b) Cache Hit Rate vs \( N \) (\( C/K \approx 1.00\% \), \( \alpha = 0.7 \))
even more the Observation 3: its key conclusion is that with high enough enough $\beta$ (i.e. low $H_\omega$) and more than 2 or 3 recommendations, while CARS aims to solve the multiple access problem, its position preference unawareness leads to suboptimal recommendation placement, and thus severe drop of its CHR performance compared to the Optimal.

VI. RELATED WORK

RS and Caching Interplay. The relation between RS and caching has only recently been considered [8], [7], [10], [11], [12], [28], [29], [30], [31]. Closer to our study, [7] considers the joint problem of caching and recommendations, placing the most popular contents (among all users) in a cache and then trying to bias recommendations to favor cached contents, taking into account position preference in their model. However, this is applied to a different setup than ours (no markovian traversal of content graph); furthermore, they do not provide any simulation results on the impact of position preference. The work in [9] tackles recommendations for users consuming multiple contents in a row, as we do. However, [9] formulates a nonconvex problem, and proposes a heuristic algorithm, and does not have optimality guarantees.

Optimization Methodology. The problem of optimal recommendations for multi-content sessions, bares some similarity with PageRank manipulation [32], [33], [22]. The idea there is to choose the links of a subset of pages (the user has access to) with the intention to increase the PageRank of some targeted web page(s). Although that problem is generally hard, some versions of the problem can also be convexified [32].

VII. CONCLUSIONS

This work has proposed the optimal solution for network-friendly position aware recommendations. This technique can be used offline in a data-center of the content provider for network cost minimization.

Nevertheless, the area is still in its infancy. Theoretical and experimental research is needed to refine user behavior models and metrics, as well as dynamic learning and optimization of system parameters (e.g., user’s reactivity to modified recommendations) or even where the “network-friendly” content appears in the recommendation list. Finally, jointly optimizing recommendations together with caching decisions (as in previous works [7], but now for multi-content sessions) is a key future step.

REFERENCES


[27] “https://groupens.org/datasets/movielen/”


