Network-aware Recommendations in the Wild: Methodology, Realistic Evaluations, Experiments

Savvas Kastanakis, Pavlos Sermpezis, Vasileios Kotronis, Daniel Menasché, Thrasyvoulos Spyropoulos

Abstract—Joint caching and recommendation has been recently proposed as a new paradigm for increasing the efficiency of mobile edge caching. Early findings demonstrate significant gains for the network performance. However, previous works evaluated the proposed schemes exclusively on simulation environments. Hence, it still remains uncertain whether the claimed benefits would change in real settings. In this paper, we propose a methodology that enables to evaluate joint network and recommendation schemes in real content services by only using publicly available information. We apply our methodology to the YouTube service, and conduct extensive measurements to investigate the potential performance gains. Our results show that significant gains can be achieved in practice; e.g., 8 to 10 times increase in the cache hit ratio from cache-aware recommendations. Finally, we build an experimental testbed and conduct experiments with real users; we make available our code and datasets to facilitate further research. To our best knowledge, this is the first realistic evaluation (over a real service, with real measurements and user experiments) of the joint caching and recommendations paradigm. Our findings provide experimental evidence for the feasibility and benefits of this paradigm, validate assumptions of previous works, and provide insights that can drive future research.

Index Terms—Mobile networks, recommendation systems, multimedia services, network measurements, user experiments.

1 Introduction

Multi-access Edge Computing (MEC) is identified as one of the key technologies for 5G networks [1]. MEC architectures enable the extension of the successful paradigm of Content Delivery Networks (CDNs) and content caching to the edge of the mobile networks, thus reducing latency of content delivery and offloading of the backhaul links. However, a key difference to CDNs is that caches in MEC are located at the edge of the mobile network (e.g., base stations), and unavoidably have limited capacity and serve small –and frequently changing– user populations [2]. These factors, despite the advances in caching policies [2]–[4] or delivery techniques [5], [6], limit the possible gains from MEC: capacity is a tiny fraction of today's content catalogs, and traffic is highly variable; hence, a large number of user requests is for non-cached contents, i.e., not served in the edge.

A recently proposed solution for increasing the efficiency of MEC is jointly considering caching and recommending content [7]–[17]. Recommendation Systems (RS) are integrated in many popular services (e.g., YouTube, Netflix) and significantly affect the user demand [18], [19]. Therefore, leveraging recommendations to steer content demand towards cached contents can significantly increase the cache hit ratio (network performance) and content delivery latency (user experience), even under the challenging conditions of small caches or populations in MEC.

As a toy example of the proposed paradigm, consider the following (depicted in Fig. 1): Assume a user watching a video A over a streaming service, whose recommendation system would suggest to the user to watch next a video B. Also assume that video B is not locally cached or needs to be fetched from a congested link, which does not allow a high quality streaming of B (low video quality, high start up delay, rebufferings, etc.). In the proposed paradigm, the recommendation system could instead suggest to the user to watch next a video C, which is still relevant to A (e.g., C is similar with B) and can be delivered in high quality (e.g., is stored in a MEC cache). This network-aware recommendation for video C can be a win-win situation for the network, which consumes less resources for the video delivery, and the user, who enjoys a better streaming experience.

Previous works generalize the above example [7]–[17], [20]–[26], by considering more general recommendation techniques and parameters (e.g., number and order of recommendations [23]), more general content delivery schemes (e.g., multiple caches [9], [17], [24], broadcasting [20], [21], [26]), and more general user demand models (e.g., acceptance of recommendations [22], sequential requests [10]).

Early findings demonstrate that the potential gains for the network performance can be significant, e.g., by increasing up to an order of magnitude the caching efficiency [9]. However, these promising results are based exclusively on evaluations on *simulation environments* and mainly consider *small content catalogs* (e.g., a few thousands contents) of *synthetic or public datasets that are not collected from real content delivery services* (e.g., MovieLens [27]). While the contribution of previous works to the understanding of the involved challenges, benefits and tradeoffs, is indisputable, it still remains uncertain *if and how these findings would change in real settings*.

Deviations from such real setting aspects may affect the expected performance. For instance: (i) Real content delivery services typically have huge content catalogs (e.g., YouTube

• *S. Kastanakis is with FORTH-ICS, Greece, and Lancaster University, UK.*
P. Sermpezis is with the Aristotle University of Thessaloniki, Greece (corresponding author; e-mail: sermpezis@csd.auth.gr).
V. Kotronis is with FORTH-ICS, Greece.
D. Menasché is with the Federal University of Rio de Janeiro, Brazil.
T. Spyropoulos is with EURECOM, France.

Manuscript received May 14, 2020; revised November 23, 2020.

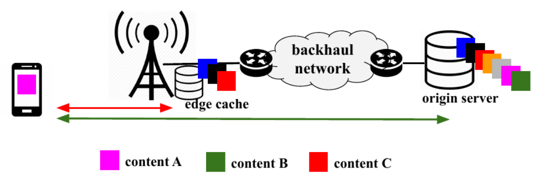


Fig. 1: Example of network-aware recommendations: Contents B and C are relevant to a content A currently consumed by a user. A baseline RS would recommend content B, while a network-aware RS recommends content C that can be served by the edge cache in a higher delivery quality.

and Netflix video catalogs are reported to be in the order of petabytes), from which recommendations are selected and within which the users can navigate. Considering only a tiny subset of these options for user actions may overestimate the gains. (ii) While the employed user models take into account the quality of recommendations and willingness of users to follow a "nudged" list of recommendations, we still lack experimental evidence whether this would indeed approximate well the real user behavior.

In this paper, we aim to address these issues and take the next step in the evaluation of the joint network and recommendations paradigm: we propose a methodology that enables evaluation under realistic settings, apply it in a real service (YouTube), and conduct measurements and experiments with real users; to our best knowledge this is the first study of this kind in the field[1]. Specifically, our contributions are summarized as follows.

Methodology. We propose a methodology that enables realistic evaluations for the joint network and recommendations paradigm (Section 2). We exploit information made publicly available from the recommendation systems of content providers, and incorporate it to the existing frameworks used for the evaluation of joint network and recommendation schemes. In this way, we circumvent the problem of the content similarity data that is required by the majority of existing works, but is not disclosed by the content providers. In fact, we claim that detailed content/user information is not necessary, and only the output of a RS may suffice for the design of joint policies. This allows to face RSs as black boxes (without disclosing sensitive/private data and algorithms), and as a side-effect, it could enable joint caching and recommendation approaches, without requiring tight collaboration between network operators and CPs (as is considered by previous works).

We apply the proposed approach, and design a network-aware algorithm (named `CABaRet`) that leverages available information provided by a RS, and returns cache-aware recommendations (Section 2.2).

Measurements and a realistic evaluation. We apply our methodology and perform extensive measurements and evaluation over the YouTube service (Section 3). Our results show that significant caching gains can be achieved in practice; even in conservative scenarios of the considered setup, our approach increases the cache hit ratio by a factor of $\times 8$ to $\times 10$. To our best knowledge, this is the first evaluation of a joint caching and recommendation approach in a real service and with realistic traffic.

Experiments with real users. We build an experimental testbed and conduct experiments with *real users* to (i) test the performance of `CABaRet` in practice, and (ii) verify to what extent the assumptions made hold in practice with real users (Section 4). We are the first to test the concept of joint caching and recommendations with real user experiments. Moreover, we publish the collected dataset and open-source the code of the testbed, which is generic and enables researchers to design and conduct their own experiments either with `CABaRet` or with any other algorithm they implement.

The experimental findings (i) are in agreement with the measured performance of `CABaRet` in Section 3, which further supports the usefulness of the proposed methodology for realistic evaluations; (ii) validate key assumptions made in related literature, and provide useful quantitative results that can drive future models, parameter selection and assumptions; and (iii) provide valuable evidence for the feasibility and benefits of the network-aware recommendations in practice, namely, users are willing to follow "nudged" recommendations and they do not perceive this as a significant compromise in recommendation quality.

Finally, we provide an overview of related work (Section 5) and conclude our paper (Section 6).

2 Methodology

2.1 Overview

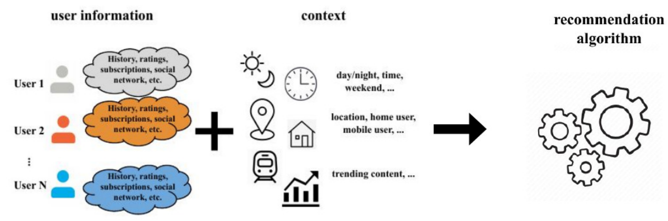
2.1.1 Motivation: the need for realistic evaluations.

We consider a communication system where a set of contents can be delivered with lower cost for the network and/or in higher QoS. To simplify our discussion, in the remainder we refer to a caching system as in the example of Fig. 1, where the set of cached contents can be delivered in low-cost/high-QoS. However, our methodology applies to generic communication setups, by simply assigning a cost value to each content, where the cost of delivering a content may depend on the network, the wireless channel conditions, the transmission (coded, broadcast, unicast), etc.

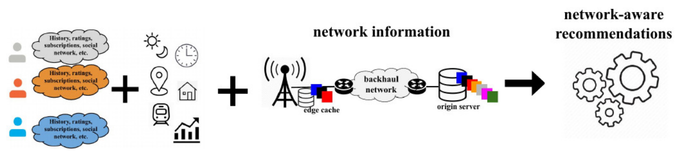
The main approach in literature for the joint design of network and recommendations, assumes that a system (for recommendations and/or caching) has full knowledge of the content similarities or user preferences. Under this assumption, it considers that the "baseline RS" (see Fig. 2(a)) would recommend to the user the contents with the highest similarity/relevance, while the "network-aware RS" (see Fig. 2(b)) could also recommend contents with lower similarity, as soon as the similarity is above a threshold (e.g., quality of recommendations [10], user preference window [8]). This relaxation in the recommendation quality is the key behind the joint network and recommendations paradigm, since it allows to recommend contents of lower similarity/relevance but which can be delivered more efficiently and/or in higher QoS by the network.

The full knowledge of content similarities/relevance can be a reasonable assumption when the network-aware RS is operated by the content provider, which holds all the information about contents and user activity. However, this approach comes with the following limitations.

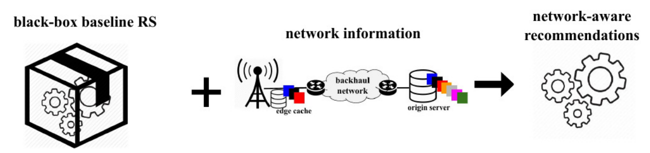
1. A preliminary version of our work, containing some parts of this paper, appears in [28].


user information
context
recommendation algorithm
User 1
User 2
User N
History, ratings, subscriptions, social network, etc.
day/night, time, weekend, ...
location, home user, mobile user, ...
trending content, ...

(a) Baseline RS



(b) Network-aware RS (previous approaches)



(c) Network-aware RS (our approach)

Fig. 2: Information that is taken into account by RS: (a) *baseline RS*: detailed user information and context; (b) *network-aware RS of previous approaches*: detailed user information (and context possibly) and network information; (c) *the proposed approach for network-aware RS*: output of the baseline RS and network information.

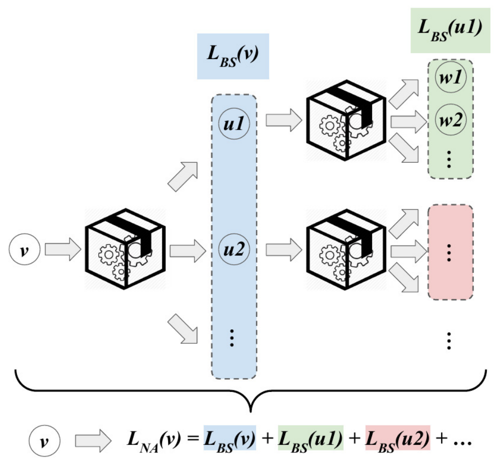


Fig. 3: Obtaining a list of "relaxed" recommendations by triggering the baseline RS in a breadth-first-search way.

The information about content similarity and user preferences is not made public by content providers, due to its sensitive nature and business value. Hence, when it comes to the evaluation, previous works rely on synthetically generated datasets of content similarity or user preferences. It is unknown whether these datasets represent well real content catalogs, content similarity, user preferences, etc. In case they significantly deviate from the structure and characteristics of the real content similarity matrices (which are shown to be determinant factors for the performance [7], [9]), the evaluation may lead to erroneous or inaccurate findings and insights about the proposed solutions.

Some works (e.g., [9], [10], [12], [13], [22]) have considered publicly available data, such as the MovieLens dataset [27] that contains real user ratings for movies, and allow to calculate content similarities (e.g., using collaborative filtering techniques [12]). However, this approach also deviates from real setups, as: (i) the sparsity of available data can lead to very different content similarities, (ii) the datasets do not correspond to content catalogs of real services and/or neglect the fact that content similarity/relevance is affected by time (trending content, transient interest in contents, etc.), and (iii) modern RS do not make recommendations only based on content similarity (collaborative filtering), but employ more complex mechanisms that take into account also user subscriptions, content diversity, trends, etc. [29].

2.1.2 The proposed methodology

To overcome the aforementioned limitation, we propose a methodology that leverages information made publicly available by the RS of content services, and enables realistic evaluations on the real content catalogs and operation of these services. Instead of using the "raw" information about content similarity or user preferences, i.e., the *input* of the baseline RS which is typically not publicly available, our methodology uses the *output* of the baseline RS that is already provided by the content service. For example, in the case of YouTube, the output of the baseline RS (i.e. YouTube's RS) is the set of recommendations displayed on the screen of the user's device. Figure 2 presents the conceptual difference between our approach (Fig. 2(c)) and previous network-aware approaches (Fig. 2(b)).

In particular, our methodology is detailed below and depicted in Fig. 3. Consider a content service (video streaming, online radio, etc.) that uses a baseline RS. When a user consumes a content v, the baseline RS provides a list $\mathcal{L}_{BS}(v)$ of contents related to v and to the current session of the user (the subscript "BS" denotes the baseline RS). The recommendations in $\mathcal{L}_{BS}(v)$ are the most relevant according to the (network-oblivious) baseline RS. A network-aware RS (denoted with subscript "NA") would need a larger set of related contents $\mathcal{L}_{NA}(v) \supseteq \mathcal{L}_{BS}(v)$, including also less relevant contents, in order to be able to exploit a relaxation in the recommendation quality for the favor of network-friendly recommendations. We construct such a set of "relaxed" recommendations $\mathcal{L}_{NA}(v)$, by recursively triggering the baseline RS in a Breadth-First Search (BFS) manner: we start by obtaining the list $\mathcal{L}_{BS}(v)$, then $\forall u \in \mathcal{L}_{BS}(v)$ we request from the baseline RS the lists $\mathcal{L}_{BS}(u)$ and append them to $\mathcal{L}_{NA}(v)$, and so on. In the end of the process, the list $\mathcal{L}_{NA}(v)$ contains contents directly and indirectly related to v, which is the key information for joint network and recommendation algorithms.

Hence, compared to previous approaches, the main advances of the proposed method are: (i) it can be applied to a real service, without being limited to samples or fractions of the content catalog; (ii) it leverages the real output of the baseline RS, thus integrating the best performing recommendation mechanisms used in production (e.g., [29]) and avoiding simplifications for the baseline recommendation mechanisms (e.g., naive item-item similarity, or collaborative filtering); (iii) it is based only on publicly available information and considers the baseline RS as a black-box, which enables to apply the approach on any real service, without requiring access to private/sensitive data.

2.1.3 Benefits beyond realistic evaluations

Apart from enabling realistic evaluations, we believe that the proposed approach brings additional benefits for the joint network and recommendations paradigm, which we briefly discuss below.

Modular architectures. The black-box approach yields a modular design, minimizing the dependence between network and RS components, e.g., the RS can be replaced or upgraded while the remainder of the system remains unchanged. This enables modular system architectures with higher robustness and increased privacy. Furthermore, it brings higher scalability; in contrast, most previous approaches require the knowledge of the entire catalog, which is huge in practice. Our initial investigation shows that while this may come at a cost of performance (i.e., knowing the entire catalog could lead to optimal performance), a good trade-off between performance and scalabitity is feasible in practice (Appendix A.2).

Techno-economic feasibility. The existing approaches in joint network and recommendations paradigm mainly assume that the content provider (CP) and the network operator (NO) are the same entity or closely collaborate and exchange information. The convergence between CPs and NOs is enabled due to the architectural developments of MEC and RAN Sharing [30], while CPs increasingly deploy their own infrastructure to bring content closer to the user, e.g., Netflix OpenConnect, Google Global Cache, or bring their equipment inside the network of NOs [31]. However, this collaboration requires some investment in infrastructure and technology, and changes in business strategy. In this context, our modular/black-box approach could lower the barrier for techno-ecomonic feasibility of the joint network and recommendations paradigm: it does not rely on the exchange of private/sensitive information, and thus can (i) be applicable even when the network and the RS are not controlled by the same entity, and (ii) cope with potential tussles between CPs and NOs.

2.2 Network-aware Recommendations with `CABaRet`

We now proceed to apply the proposed methodology in the real service of YouTube. To this end, we design a network-aware recommendation algorithm (`CABaRet`) that leverages information provided by the YouTube RS (Section 2.2.1), and discuss the related design implications (Section 2.2.2).

Remark: We would like to stress that while we focus on the YouTube service, our approach and the `CABaRet` algorithm are generic and can be applicable to other video/radio services. For example, the majority of popular content services provide public APIs, such as Vimeo [32], Twitch [33], Dailymotion [34], Spotify [35], or in case APIs are not available, indirect methods (e.g., web-based parsing) exist for retrieving content recommendations, e.g., for Netflix [36] or for Facebook through the *Tracking Exposed* project [37].

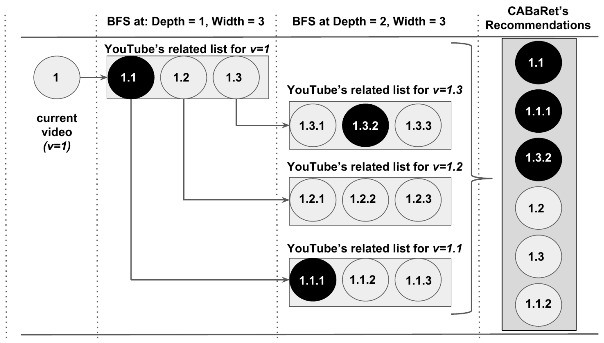

BFS at: Depth = 1, Width = 3
BFS at Depth = 2, Width = 3
CABaRet's Recommendations
YouTube's related list for v=1
1
1.1
1.2
1.3
current video (v=1)
YouTube's related list for v=1.3
1.3.1
1.3.2
1.3.3
YouTube's related list for v=1.2
1.2.1
1.2.2
1.2.3
YouTube's related list for v=1.1
1.1.1
1.1.2
1.1.3
1.1
1.1.1
1.3.2
1.2
1.3
1.1.2

Fig. 4: CABaRet: example with $D_{BFS} = 2$, $W_{BFS} = 3$, $N = 6$. Cached videos are denoted with black color.

2.2.1 The `CABaRet` Algorithm

`CABaRet` overview. `CABaRet` receives information about content relations from the YouTube RS through its API. In particular, when a user watches a video v, `CABaRet` requests from the YouTube API a list of video IDs $\mathcal{L}$ related to v, i.e., the videos that YouTube would recommend to the user. Then it requests the related video IDs for every video in $\mathcal{L}$ and adds them in the end of $\mathcal{L}$, and so on, in a Breadth-First Search (BFS) manner. In the end of the process, the list $\mathcal{L}$ contains IDs of videos directly and indirectly related to v; of these videos, the top N that are cached and/or highly related to v are finally recommended to the user. An example is depicted in Fig. 4.

Input. The recommendation algorithm receives as input:

- v: the video ID (or URL) which is currently watched
- N: the number of videos to be recommended
- $\mathcal{C}$: the list with the IDs of the cached videos
- D_{BFS}: the depth to which the BFS proceeds
- W_{BFS}: the number of related videos that are requested *per content* from the YouTube API (i.e., the “width” of BFS)

Output. The recommendation algorithm returns as output:

- $\mathcal{R}$: ordered list of N video IDs to be recommended.

Workflow. `CABaRet` searches for videos related to video v in a BFS manner as follows (*line 1* in Algorithm 1). Initially, it requests the W_{BFS} videos related to v, and adds them to a list $\mathcal{L}$ in the order they are returned from the YouTube API. For each video in $\mathcal{L}$, it further requests W_{BFS} related videos, as shown in Fig. 4, and adds them in the end of $\mathcal{L}$. It proceeds similarly for the newly added videos, until the depth D_{BFS} is reached; e.g., if $D_{BFS} = 2$, then $\mathcal{L}$ contains W_{BFS} video IDs related to v, and $W_{BFS} \cdot W_{BFS}$ video IDs related to the related videos of v.

Then, `CABaRet` searches for video IDs in $\mathcal{L}$ that are also included in the list of cached videos $\mathcal{C}$ and adds them to the list of video IDs to be recommended $\mathcal{R}$, until all IDs in $\mathcal{L}$ are explored or the list $\mathcal{R}$ contains N video IDs, whichever comes first (*lines 4–9*). If after this step, $\mathcal{R}$ contains less than N video IDs, $N - |\mathcal{R}|$ video IDs from the head of the list $\mathcal{L}$ are added to $\mathcal{R}$; these are the top $N - |\mathcal{R}|$ non-cached video IDs that are directly related to video v (*lines 10–15*).

Extension: different costs per video. In more generic setups, each video i may have a different delivery cost c_i. `CABaRet` can be easily modified for this case, by selecting the N video with the lowest costs c_i (and prioritizing videos found earlier in the BFS among those with equal costs).

Algorithm 1
CABaRet: Cache-Aware & BFS-related Recommendations

```
    $Input: v, N, \mathcal{C}, D_{BFS}, W_{BFS}$
 1: $\mathcal{L} \leftarrow BFS(v, D_{BFS}, W_{BFS})$        ▷ ordered set of video IDs
 2: $\mathcal{R} \leftarrow \emptyset$        ▷ ordered set of video IDs to be recommended
 3: $i \leftarrow 1$
 4: for $c \in \mathcal{L}$ do
 5:     if $i \leq N$ and $c \in \mathcal{C}$ then
 6:         $\mathcal{R}.append(c)$
 7:         $i \leftarrow i + 1$
 8:     end if
 9: end for
10: for $c \in \mathcal{L} \setminus \mathcal{R}$ do
11:     if $i \leq N$ then
12:         $\mathcal{R}.append(c)$
13:         $i \leftarrow i + 1$
14:     end if
15: end for
16: $return\ \mathcal{R}$
```

2.2.2 Implications and Design Choices

High-quality recommendations. Using the baseline RS (here, the YouTube recommendations) ensures strong relations between videos that are directly related to v (i.e., BFS at depth 1). Moreover, typically the baseline RS provides only a subset of the relevant recommendations to the user; for example, while the YouTube RS finds hundreds of videos highly related to v, only a few of them (e.g,. 5 or 20, depending on the end device) are finally communicated to the user [29]. The rationale behind our methodology and CABaRet is to explore the related videos that are not communicated to the user. To this end, based on the fact that related videos are similar and have high probability of sharing recommendations (i.e., if video a is related to b, and b to c, then it is probable that c relates to a) [38], [39], CABaRet tries to infer these latent video relations through BFS. Hence, videos found by BFS in depths > 1 are also (indirectly) related to v and probably good recommendations.

To further support the above claim, we collect and analyze datasets of related YouTube videos. Specifically, we consider the set of most popular videos, denoted as $\mathcal{P}$, in a region, and for each $v \in \mathcal{P}$ we perform BFS by requesting the list of related videos (similarly to *line 1* in CABaRet). We use as parameters $W_{BFS} = \{10, 20, 50\}$ and $D_{BFS} = 2$, i.e., considering the directly related videos (depth 1) and indirectly related videos with depth 2. We denote as $\mathcal{R}_1(v)$ and $\mathcal{R}_2(v)$ the set of videos found at the first and second depth of the BFS, respectively. We calculate the fraction of the videos in $\mathcal{R}_1(v)$ that are also contained in $\mathcal{R}_2(v)$, i.e., $I(v) = \frac{|\mathcal{R}_1(v) \cap \mathcal{R}_2(v)|}{|\mathcal{R}_1(v)|}$. High values of $I(v)$ indicate a strong similarity of the initial content v with the set of indirectly related contents at depth 2.

Table 1 shows the median values of $I(v)$, over the $|\mathcal{P}| = 50$ most popular contents in the region of Greece (GR), for different BFS widths. As it can be seen, $I(v)$ is very high for most of the initial videos v. For larger values of W_{BFS}, $I(v)$ increases, and when we fully exploit the YouTube API capability, i.e., for W_{BFS}=50, which is the maximum number of related videos returned by the YouTube API, the median value of $I(v)$ becomes larger than 0.9. Finally, we measured the $I(v)$ in other regions as well, and observed that even in large (size/population) regions, the $I(v)$ values remain high, e.g., in the United States (US) region, $I(v)$=0.8 for W_{BFS}=50.

TABLE 1: $I(v)$ vs. W_{BFS} for the region of GR.

W_{BFS} :	10	20	50
$I(v)$:	0.70	0.85	0.92

Tuning CABaRet. Typically, users prefer videos in the top of the recommendation list, hence, CABaRet puts in the top of the list $\mathcal{R}$ the cached videos found in the BFS[2].

Moreover, the parameters D_{BFS}, W_{BFS} can be tuned to achieve a desired performance, e.g., in terms of probability of recommending a cached or highly related video. For large D_{BFS}, the similarity between v and the videos *at the end of the list* $\mathcal{L}$ is expected to weaken, while for small D_{BFS} the list $\mathcal{L}$ is shorter and it is less probable that a cached content is contained in it. Hence, the parameter D_{BFS} can be used to achieve a trade-off between quality of recommendations (small D_{BFS}) and probability of recommending a cached video (large D_{BFS}). Note that while other methods (e.g., DFS, random walk) can be used by CABaRet to search for related/cached videos, BFS returns the most related videos, and thus is optimal with respect to the quality of recommendations (given a search budget).

The number of related videos requested per content W_{BFS}, can be interpreted similarly to D_{BFS}. A small W_{BFS} leads to considering only top recommendations per video, while a large W_{BFS} leads to a larger list $\mathcal{L}$. For the size of the list $\mathcal{L}$ it holds that

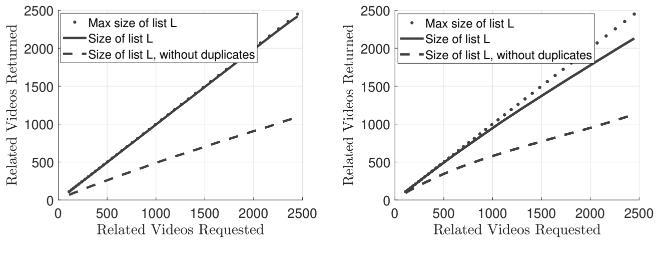
$$|\mathcal{L}| \leq \sum_{n=1}^{D_{BFS}} (W_{BFS})^n \qquad (1)$$

where the equality holds when all videos found by the BFS are unique.

Remark: The time and space complexity of CABaRet is $O(|\mathcal{L}|)$, or by using Eq. (1) it is $O((W_{BFS})^{D_{BFS}})$. While the complexity is exponential to D_{BFS}, in practice small values of D_{BFS} (e.g,. 1 or 2) are enough to achieve high efficiency, thus rendering CABaRet a lightweight algorithm that can run in real-time (see Section 4).

Remark: YouTube imposes quotas on the API requests per application per day, which prevents API users from setting the parameters W_{BFS} and D_{BFS} to arbitrarily large values. However, even with small number of API requests (for related contents), the exploration returns a large number of *unique* videos. Figure 5 shows how many relations are requested for parameters $W_{BFS} \in \{1, ..., 50\}$ and $D_{BFS} = 2$ (x-axis), versus the size of the returned list $|\mathcal{L}|$ (y-axis). Two settings are considered, where the BFS starts from a top trending YouTube video from the YouTube "front page" or from a video searched through the "search bar" (see details in Section 3). In both cases, and as already suggested by the results of Table 1, the BFS discovers several duplicates. On the one hand, this indicates a high-quality of recommendations. On the other hand, the number of unique video IDs in the list $\mathcal{L}$, increases linearly or almost linearly with the

2. Nevertheless, if for a service the patterns of users preferences is different (e.g., preference to the bottom of recommendation list), CABaRet could be tuned accordingly.



(a) "Front Page" video demand (b) "Search Bar" video demand

Fig. 5: Num. of requested related videos VS Num. of returned related videos from the API.

number of explored relations, thus indicating that the BFS achieves an efficient exploration (large $|\mathcal{L}|$).

Finally, N can be selected to fit different device or application settings (e.g., browser or mobile app), but also affects the performance; e.g., if N is small, the user has a few options, which can further promote cached videos but decrease the quality of recommendations.

In practice, `CABaRet` can be further fine-tuned through experimentation with real users, e.g., A/B testing [29].

Performance modeling. The performance of `CABaRet` can be measured by the fraction of requests made for the cached videos (i.e., in our setup, the *cache hit ratio* or *CHR*). This depends on how many cached videos are recommended to the user and the probability a user to select one of them; in practice, these quantities are intertwined and depend on complex user demand patterns. In the following sections, we conduct extensive measurements and experiments to quantify the achieved performance in the YouTube service and under realistic user demand patterns. However, here, we also provide an analytical model to predict the achieved CHR, which can be applied to any service and can be useful to obtain initial performance estimations (e.g., before proceeding to measurements for a more detailed evaluation).

Let us denote the probability that a user selects a video at the i^{th} position ($i = 1, ..., N$) of the recommendation list as p_i. Also let M be the number of cached contents that is found in the BFS, i.e., $M = |\mathcal{L} \cap \mathcal{C}|$, and thus the number of cached videos in the recommendation list of `CABaRet` is $\max\{M, N\}$. Then, the fraction of requests for cached videos (CHR) will be $\sum_{i=1}^{\max\{M,N\}} p_i$, and taking the expectation over M, gives:

$$CHR = \sum_{m=1}^{|\mathcal{L}|} \left(\sum_{i=1}^{\max\{m,N\}} p_i \right) \cdot P\{M = m\}$$

where M follows a Binomial distribution with $|\mathcal{L}|$ number of trials and success probability q_C, where q_C is the probability that a recommendation is for a cached content[3].

For the most common case of $p_i \geq p_j$ for $i < j$ (i.e., users preference is higher for top recommendations), the inner sum in the above expression is a concave function of m. Thus, Jensen's inequality allows us to upper bound the CHR by a simpler expression involving only the mean number of cached contents found by the BFS $\bar{M}$: [4] $CHR \leq \sum_{i=1}^{\max\{\lceil \bar{M} \rceil, N\}} p_i$. This bound is tight for our actual measurement results in Section 3.

Despite the assumptions made in the model (e.g., independence between p_i and the set of recommended videos), it can be generalizable to any service (e.g., to short-video services having considerably different user demand patterns than YouTube [16], [43]) given general user demand statistics, i.e., q_C and p_i.

Caching optimization under `CABaRet`. `CABaRet` receives as input a list of cached videos $\mathcal{C}$ (or, more general, videos that can be delivered by the network in high quality) and returns cache-aware recommendations to increase the caching efficiency. Depending on the considered scenario, it may be possible to control the list $\mathcal{C}$ as well. Carefully selecting the contents in the list $\mathcal{C}$ can lead to further increase of the caching efficiency [12], [13]. Under `CABaRet` recommendations it is possible to design the caching policy as well, so that it further increases the cache hit ratio as we showed in our preliminary work [28]. While a detailed investigation is out of the scope of this paper, we provide in Appendix A a formulation of the optimization problem, an approximation algorithm, as well as evaluation results for the extra increase that can be achieved by jointly selecting the caching policy under `CABaRet`.

3 Measurements and Evaluation

Using the proposed methodology and the `CABaRet` algorithm, we conduct extensive measurements and experiments over the YouTube service[5], to investigate the performance (in terms of cache hit ratios) of network-aware recommendations in MEC scenarios. The setup of the scenarios is presented in Section 3.1, and the results in Section 3.2 and Section 3.3 for two video demand types.

3.1 Setup

The YouTube API provides a number of functions to retrieve information about videos, channels, user ratings, etc. In our measurements, we request the following information:

- the most popular videos in a region (max. 50)
- the list of related videos (max. 50) for a given video

Remark: In the remainder, we present results for the region of Greece (GR). Nevertheless, our insights hold also in the other regions we tested, and, indicatively, we briefly state results for the region of United States (US).

Caching. We assume a MEC cache storing the most popular videos in a region. Unless otherwise stated, we populate the list of cached contents with the top C video IDs returned from the YouTube API.

Recommendations. We consider two classes of scenarios with (i) YouTube and (ii) `CABaRet` recommendations. In both cases, when a user enters the UI, the 50 most popular videos in her region are recommended to her (as in

3. Typically, (i) the most popular contents are cached and (ii) recommendations have bias towards popular contents (popularity bias [40]–[42]), which leads to high q_C, and thus high CHR.

4. If user preferences are for recommendations at the end of the list ($p_i \leq p_j$ for $i < j$), Jensen's inequality gives $CHR \geq \sum_{i=1}^{\max\{\lfloor \bar{M} \rfloor, N\}} p_i$

5. Our experiments and use of the YouTube API conform to the YouTube terms of service https://www.youtube.com/static?template=terms.

YouTube's front page). Upon watching a video v, a list of $N = 20$ videos is recommended to the user; the list is (i) composed of the top N directly related videos returned from the YouTube API (YouTube scenarios), or (ii) generated by `CABaRet` with parameters N, W_{BFS} and D_{BFS} (`CABaRet` scenarios).

Video Demand. In each experiment, we assume a user that enters the UI and selects an initial video to watch in one of the following ways: (a) "front-page recommendations": the user selects to watch one of the initially recommended (i.e., 50 most popular) videos recommended in the front page; or (b) "search bar": the user types in the search bar a keyword of her interest, and selects one of the returned video recommendations. These two types of initial requests represent the two most common ways of user behavior (note that the former captures also trending videos selections) [18]. We present the results for each of the aforementioned initial video demand types separately, in Section 3.2 and Section 3.3, respectively; the former is expected to have a more concentrated demand among the most popular videos (and thus, higher CHR, since those are assumed to be cached), while the latter a more varying demand that stresses the caching system.

After the initial video, the system recommends a list of N videos $(r_1, r_2, ..., r_N)$, and the user selects with probability p_i to watch r_i next. We set the probabilities p_i to depend on the order of appearance –and not the content– and consider *uniform* ($p_i = \frac{1}{N}$) and *Zipf* ($p_i \sim \frac{1}{i^\alpha}$) scenarios; the higher the exponent α of the Zipf distribution, the more preference is given by the user to the top recommendations (user preference to top recommendations has been observed in YouTube traffic [44]).

3.2 Results: "Front-Page" Video Demand

3.2.1 Single Requests

We first consider scenarios of single requests (similarly to [8], [45]). In each experiment i ($i = 1, ..., M$) a user watches one of the top popular videos, let $v_1(i)$, and then follows a recommendation and watches a video $v_2(i)$. We measure the Cache Hit Ratio (CHR), which we define as the fraction of the *second requests* of a user that are for a cached video (since the first request is always for a cached –top popular– video):

$$CHR = \frac{1}{M} \cdot \sum_{i=1}^{M} \mathbb{I}_{v_2(i) \in \mathcal{C}} \quad (2)$$

where $\mathbb{I}_{v_2(i) \in \mathcal{C}} = 1$ if $v_2(i) \in \mathcal{C}$ and 0 otherwise, and M the number of experiments[6].

CHR vs. BFS parameters. Fig. 6 shows the CHR achieved by `CABaRet` under various parameters, along with the CHR under regular YouTube recommendations, when caching all the most popular videos ($|\mathcal{C}|$=50). The efficiency of caching significantly increases with `CABaRet`, even when only directly related contents are recommended (D_{BFS}=1), i.e., *without loss in recommendation quality*. Just reordering the list of YouTube recommendations (as suggested in [44]), brings gains when p_i is not uniformly distributed. However, the added gains by our approach are significantly higher. As expected, the CHR increases for larger W_{BFS} and/or D_{BFS}; e.g., `CABaRet` for W_{BFS}=50 and D_{BFS}=2, achieves 8 to 10 times higher CHR than regular YouTube recommendations. Also, the CHR increases for more skewed p_i distributions, since top recommendations are preferred and `CABaRet` places cached contents at the top of the recommendation list.

6. We considered all possible experiments on the collected dataset.

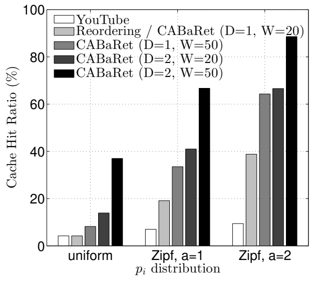


Fig. 6: CHR under different BFS parameters.

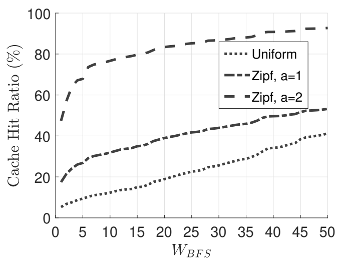


Fig. 7: CHR vs. W_{BFS} (D_{BFS}=2).

In experiments concerning the –larger– US region, the CHR values are lower for both regular YouTube ($< 0.5\%$) and `CABaRet` ($1\% - 43\%$) recommendations, due to the fact that the top popular videos appear with lower frequency in the related lists. However, the *relative gains* from `CABaRet` are consistent with (or even higher than) the presented results.

CHR vs. knowledge of content relationships. `CABaRet` uses only partial knowledge (i.e., black-box) of content relationships. This could bring some reduction in the maximum gains that can be achieved by a network-aware RS (knowledge vs. performance trade-off). For example, previously proposed algorithms that assume knowledge of the entire content relationships graph (which is equivalent to `CABaRet` with a large enough parameter W_{BFS} to explore the entire catalog) could achieve higher gains. To quantify this trade-off, we present in Fig.7 the CHR of `CABaRet` (performance) vs. the W_{BFS} parameter (knowledge of content relationships). As expected the CHR increases when more information about the content relationships is available (i.e., larger W_{BFS}). However, when p_i follows a Zipf distribution, which is more common in practice, the effect of W_{BFS} is less intense. This indicates that the benefits of the proposed black-box approach (see Section 2.1) can be combined with a performance that is comparable to approaches requiring more information about the content relationships.

CHR vs. number of cached videos. We further consider scenarios with varying number of cached contents $C = |\mathcal{C}|$. In each scenario, we assume that the C most popular contents are cached. Fig. 8 shows the CHR achieved by `CABaRet`, in comparison to scenarios under regular YouTube recommendations. The results are consistent for all considered values of C; the CHR under `CABaRet` is significantly higher than in the YouTube case. Moreover, even when caching a small subset of the most popular videos, `CABaRet` brings significant gains. E.g., by caching $C = 10$ out of the 50 top related contents `CABaRet` increases the CHR from 2% and 3.2% to 17% and 50%, for the uniform and Zipf(α=1) scenarios, respectively.

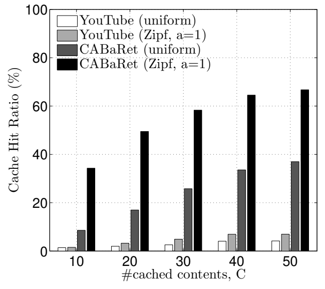


Fig. 8: CHR vs. # cached contents C (W_{BFS}=50, D_{BFS}=2).

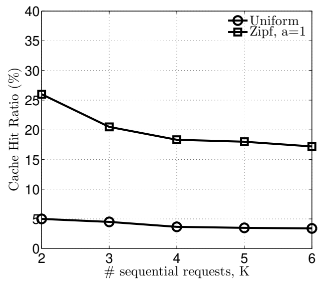


Fig. 9: CHR vs. # requests in sequence K (C=20, W_{BFS}=20, D_{BFS}=2). *Note*: y-axis up to 40%.

3.2.2 Sequential Requests

We now test the performance of our approach in scenarios where users enter the system and watch a sequence of K, $K > 2$, videos (similarly to [10], and in contrast to the previous case, where they watch only two videos, i.e., $K = 2$). At each step, the system recommends a list of videos to the user by applying `CABaRet` on the currently watched video. We denote as $v_k(i)$ the k^{th} video requested/watched by a user in experiment i. We measure the CHR, which is now defined as

$$CHR = \frac{1}{M} \cdot \sum_{i=1}^{M} \sum_{k=2}^{K} \mathbb{I}_{v_k(i)\in\mathcal{C}} \quad (3)$$

where $\mathbb{I}_{v_k(i)\in\mathcal{C}} = 1$ if $v_k(i) \in \mathcal{C}$ and 0 otherwise, over $M = 100$ experiments per scenario.

Moving "farther" from the initially requested video (which belongs to the list of most popular and cached videos) through a sequence of requests, we expect the CHR to decrease, due to lower similarity of the requested and cached videos. However, as Fig. 9 shows, the decrease in the CHR (under `CABaRet` recommendations) is not large. The CHR remains close to the case of single requests (i.e., for K=2 in the x-axis), indicating that our approach performs well even when we are several steps far from the cached videos. In fact, caching more than the top most popular videos appearing on the front page, would further reduce the CHR decrease.

3.3 Results: "Search Bar" Video Demand

Up to now, we have considered a user that starts his/her viewing session by selecting one of the trending videos recommended in the YouTube homepage. While this is a common behavior (in YouTube and similar services), we now consider the other popular option for a user, which is to enter the YouTube webpage/app and select a desired video (e.g., through the search bar or directly typing the video url) irrespectively of the current trends. In the following, we describe our measurements and experiments for realistic scenarios that capture this second class of user behavior. Since considering users to select arbitrary initial videos, dramatically increases the set of initial videos (i.e., from 50 top trending contents in Section 3.2 to -theoretically- the entire YouTube catalogue that counts more than 5 billion videos), the CHR achieved by `CABaRet` (and any algorithm) is expected to decrease. Our goal here is to quantify the CHR gains when users start their session by searching a video through the search bar, and test whether the proposed approach can still provide considerable benefits in this "worst-case" scenario.

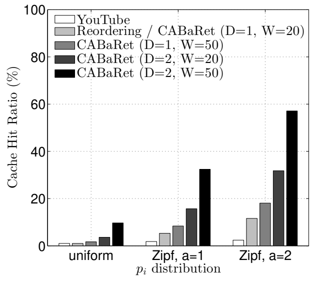
Remark: The cache stores the C top most popular YouTube videos, as in Section 3.1. Hence, our results are comparable to the results of Section 3.2, and demonstrate the performance of the same scheme for this second class of users (who have different initial video demand).

Initial video demand through the "search bar". We assume a user that enters the UI and searches though the search bar for a video according to her preferences (i.e., she does not watch one of the recommended trending videos as in Section 3.1). While for the initial demand we could select randomly a video from the entire YouTube catalogue, e.g., uniformly or with a probability proportional to the total number of views, this would not capture the user behavior observed in practice: not all contents are equally probable to be selected, total number of views is not necessarily proportional to current demand (e.g., recent videos attract more clicks than older videos), timely topics attract more attention, etc. Hence, to simulate realistic "video searches" we apply the following methodology.

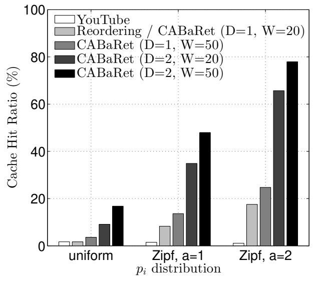
- The user types a keyword/phrase in the search bar. To obtain a dictionary of keywords that correspond to popular and recent interests, we use the Google Trends API [46]. For each region, we collect the top 10 keywords for seven consequent days within a week (some examples of keywords from our dataset are "NBA Top Plays", "Avengers Trailer", "Grammy Nominations", "How to boil an egg?", etc.).
- To map keywords (Google Trends) to YouTube videos, we pass each keyword to the YouTube API, which returns a list of video IDs, i.e., the list that would be returned if a user entered this keyword in the YouTube search bar. We select the first video ID from the list of each keyword. In total, we collect 70 video IDs, of which we use the first 50 (for consistency with the top 50 trending videos in Section 3.1). We call the list of these 50 video IDs, as "top Google trends".
- In each experiment, we assume a user that enters the UI, watches one of the 50 "top Google trends" videos, and then select one of the N recommended videos to watch next (as described in Section 3.1.

CHR vs. BFS parameters. Figure 10(a) shows the CHR (single requests - Eq. (2)) achieved with different `CABaRet` parameters in various scenarios (x-axis). The absolute values of CHR are in all scenarios 20%–65% lower compared to those in Fig. 6, which corresponds to users with initial requests for the top popular YouTube videos. While this decrease is expected in these more challenging scenarios (since the top YouTube videos are cached, but users start their viewing session from arbitrary videos), `CABaRet` can still effectively exploit the caching vectors and achieve, e.g., a CHR up to 32% and 57% in the Zipf(a=1) and Zipf(a=2) scenarios, respectively, in which otherwise we would observe a percentage within 1%–2.4% of cache hits under the original YouTube recommendations.

Moreover, we observe that the relative difference in performance between the different recommendation schemes



(a) Caching: top-50 most popular YouTube



(b) Caching: top-50 Google trends

Fig. 10: CHR under different BFS for "Search Bar" video demand and cache populated with the (a) top-50 most popular videos, and (b) top-50 Google trends

(e.g., YouTube vs. `CABaRet`) remains the same as in Fig. 6: `CABaRet` significantly increases the caching efficiency by 10 times (uniform) to more than 20 times (Zipf, a=2) even for the more diverse (and thus challenging for the caching system) "Search Bar" video demand patterns.

CHR vs. set of cached contents. Figure 10(b) shows the CHR in the same video demand scenarios presented in Fig. 10(a), but now under a different caching policy. We consider a cache that stores 50 videos of the "top Google trends". This makes the caching policy more targeted to the "Search Bar" traffic demand, and thus we expected an improved performance. Fig. 10(b) verifies this intuition: in all scenarios the CHR of `CABaRet` is higher than in Fig. 10(a). These results suggest that changing also the caching policy to better match the recommendations, i.e., joint selection of caching and recommendations, can further improve performance; indeed in Appendix A we show that when caching is optimized under `CABaRet` recommendations, an extra $\times 2$ increase in the cache hit ratio can be achieved.

Another interesting observation in the top Google trends caching scenarios is that even with $W_{BFS} = 20$ we can achieve high performance (CHR comparable to $W_{BFS} = 50$), which was not the case in the other scenarios we tested (e.g., Fig. 6 or Fig. 10(a)). This indicates that the main factor to improve performance is the depth of the BFS: for $D_{BFS} = 2$ the CHR becomes significantly higher; due to more diversity in "Search Bar" video demand and top Google trends, we need to explore deeper in the video relationship lists.

4 Experiments with Real Users

4.1 Experimental Testbed

We implemented an experimental platform with the architecture and main functionality of the framework presented in Section 2. Our goal is to conduct experiments with real users to (i) evaluate the performance that can be achieved in practice, and (ii) validate our assumptions, insights and measurement findings.

Overview. The UI is designed to accommodate our experiments (rather than resembling a real service or a prototype), and a screenshot is shown in Fig. 11 (more details in Section 4.2). For the back-end, we assume that a list of cached video IDs is available at the time of the experiment (see Section 3.1), and we use the YouTube API to embed a YouTube video player in our platform and serve video contents to the participants of the experiment. Finally, we generate recommendations using the `CABaRet` algorithm.

Open-source code. To facilitate future research on this topic, we open-source the code of the experimental testbed [47]. Moreover, our implementation is modular and easily extensible. Thus, researchers and practitioners can use (as well as configure, parametrize, modify, or extend) our testbed to conduct their own experiments. More specifically: (i) the UI can be easily configured to present a desired number of recommendations N, include a search bar (e.g., to conduct experiments similar to Section 3.3), add/remove rating questions, etc.; (ii) the list of cached video IDs in the back-end can be arbitrarily modified; (iii) the researcher can implement and use any other new algorithm (instead of `CABaRet`), by only modifying and calling a different method in the recommendation module.

Collected dataset. We conducted an experimental campaign recruiting participants through mailing lists and social media, and collected 742 samples from users in regions around the world. Adding to the open-source code, we also publish the dataset with the results of our experiments [47], which contains more information than those presented in this paper[7]. We believe that this dataset can be of interest and facilitate researchers, since recruiting users and conducting experiments is an arduous task.

4.2 Experiment Session

We invited users to visit our platform and participate in our experiment. We first summarize the steps of each experiment/session, and then elaborate on some key steps.

Action 1: The user enters the platform and is requested to select from a list his/her preferred *region*.

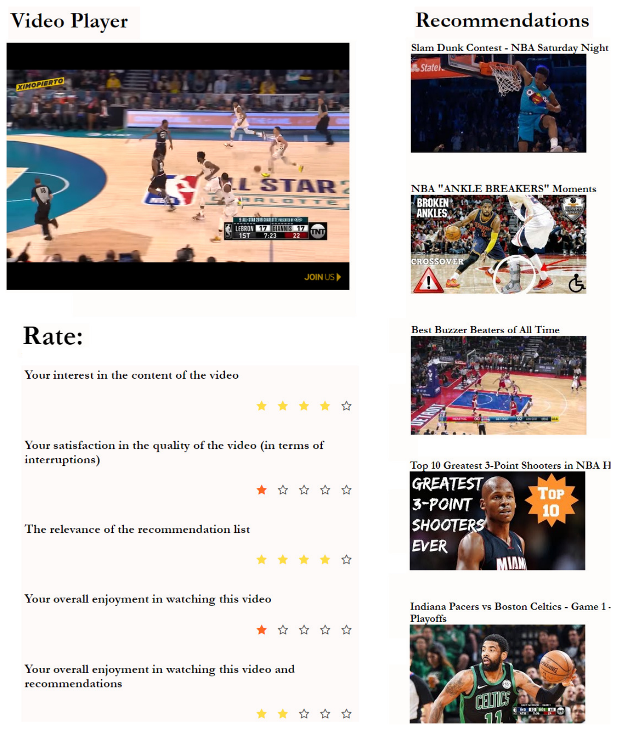
Action 2: After selecting a region, she is redirected to a page with instructions about the experiment. There, she is asked to start the viewing session by selecting a video from a list of 20 trending (in the selected region) videos.

Action 3: When selecting a video to watch, the user is redirected to a page as shown in Fig. 11, where: (a) The user watches the video (for as much time as she wants); (b) 5 videos are recommended to the user to watch next; (c) the user is requested to provide some ratings about her viewing experience, including the relevance of recommendations (*QoR*).

Action 4: The user selects one of the 5 recommended videos to watch next, and then step 3 is repeated. The maximum number of videos to watch is 5. After the fifth video, the experiment session ends.

The information that is communicated to the users (when they enter the experimental platform) is that they are going to select, watch, and rate a series of five YouTube videos for the purposes of a research study. No further information is revealed to users about how we select the videos to recommend, to avoid biasing their selections and ratings. We also inform the users that no personal information is collected.

7. We refer the interested reader to [48] for a more detailed analysis of the experimental results.


Video Player

Recommendations

Slam Dunk Contest - NBA Saturday Night

NBA "ANKLE BREAKERS" Moments

Best Buzzer Beaters of All Time

Top 10 Greatest 3-Point Shooters in NBA H

Indiana Pacers vs Boston Celtics - Game 1 - Playoffs

Rate:

Your interest in the content of the video ★★★★☆

Your satisfaction in the quality of the video (in terms of interruptions) ★☆☆☆☆

The relevance of the recommendation list ★★★★☆

Your overall enjoyment in watching this video ★☆☆☆☆

Your overall enjoyment in watching this video and recommendations ★★☆☆☆

Fig. 11: Experimental platform - instance of a user experiment: (i) a user watches a video (top/left), and is requested to (ii) rate her satisfaction from the watched video and recommendations (bottom/left) and (iii) select one of the recommendations to proceed to the following video (right).

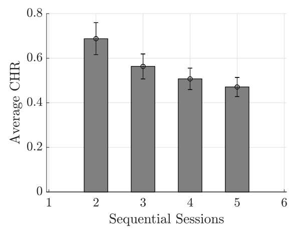


Fig. 12: CHR vs. #requests in sequence K (C=500, W_{BFS}=20, D_{BFS}=2).

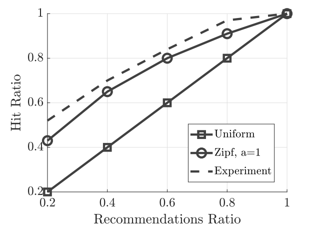


Fig. 13: CHR vs fraction of cached videos in recommendation list

4.3 Experiment Setup

Region (Action 1). We offer as options a subset of the regions provided by the YouTube API [49]; we selected 7 representative regions (different continents, diverse demographics, available video data).

Initial list of videos (Action 2). For each region, we retrieve from the YouTube API the list of 50 top trending videos. We randomly select 20 of them (for the selected region) to present to the user.

Caching. We compiled a list of 500 videos IDs that are assumed to be cached[8] ; we consider a different list per region. In each list, we select to first include the top 50 trending videos in this region. Then, for each of these 50 videos, we request its 50 recommendations / related videos provided by YouTube API. From these 2500 (50 $\times$ 50) total videos, we add in the list the 450 videos with the higher number of views ("most popular").

List of recommendations (Action 3b). The list of the 5 recommendations given to the user when watching a video are generated by `CABaRet`. We tuned the parameters of `CABaRet` as follows: the width of the BFS is 50 in the first depth, and for the first 10 of the item found in the first depth we search in second depth as well and retrieve a list of 50; in total we compile a list of $50 + 10 \cdot 50 = 550$ videos. This modification compared to the parameters used in Section 3 was done for scalability reasons (number of available credits, time needed by the YouTube API to respond, etc.).

Collected data (Action 3c). In each experiment session we collect the following data:

- ID of watched video
- IDs of the final recommendation list (i.e., the 5 videos presented in the right side in Fig. 11), and the positions of videos in this list
- ID of the initial YouTube recommendations; these videos were not presented to the user
- IDs of videos that are (assumed to be) cached
- User ratings

4.4 Results

Key finding: The CHR in the real-user experiments is 47%.

In our experiments with `CABaRet` recommendations, a percentage of 47% among the videos selected and watched by real users, was for cached videos[9]. While our experimental results are admittedly preliminary for a quantitative analysis, they qualitatively verify that we achieve in practice (i.e., with real users) the CHR values demonstrated in Section 3.

Moreover, in Fig. 12 we present how the CHR (calculated as in Eq. (3)) varies with the number of the sequence requests, i.e., when we move farther from the initial recommendations for the top popular (and cached) videos. We observe that our findings validate the corresponding measurement results in Fig. 9, i.e., as expected the CHR decreases (from around 70% in the second step to 50% after five steps), however, this decrease is not large.

The observed decrease in the CHR, when moving from the top popular list is due to the fact that there can be found less (directly or indirectly) related contents that are cached. Table 2 shows the fraction of sessions, in which no cached content was found by `CABaRet` at x^{th} request in sequence by a user. After five requests, in 11% of the cases `CABaRet` did not find any cached related video to recommend (i.e., resulting in at least 11% cache misses), while among the first requests this percentage is only 2%.

Finally, we present in Table 3 the achieved CHR in our experiments by considering different number (and sets) of cached contents (i.e., fractions of the total 500 contents

8. Note that we do not cache any video, since this is not allowed by the terms of use of the YouTube service.

9. Note that, similarly to the calculation of Eq. (2), this percentage does not include the first video views of the experiments (i.e., *Action 2*), since all initial recommendations are for cached videos.

TABLE 2: Percentage of experiment samples in which none of the videos in the recommendation list was cached.

Request step	1	2	3	4	5
% experiment samples	2%	5%	8%	10%	11%

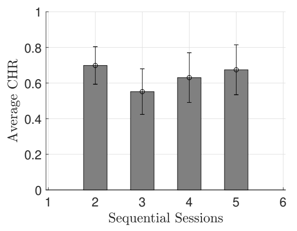


Fig. 14: CHR when at least one recommended content is cached vs. #requests in sequence K (C=500, W_{BFS}=20, D_{BFS}=2).

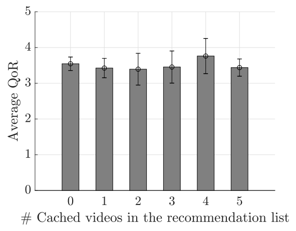


Fig. 15: QoR vs. #cached videos in recommendation list

assumed cached in the baseline scenario)[10]. The increase in CHR is almost linear with C in both scenarios, which is in line with the measurement results (Fig. 8). Most popular caching is more efficient than random caching (as expected), and this effect of the caching policy becomes more important for smaller cache sizes.

TABLE 3: CHR vs. cache size C, with the most popular (top row) or random (bottom row) contents being cached.

C	50	100	200	300	400	500
CHR (most popular)	0.11	0.16	0.24	0.32	0.40	0.47
CHR (random)	0.05	0.12	0.19	0.27	0.38	0.47

Key finding: Users tend to select the top recommendations, even when those are "nudged" towards cached contents.

However, what happens when at least one cached content can be recommended (i.e., is directly or indirectly related to the currently watched content)? Figure 14 shows the CHR per step (x-axis) when at least one cached content is recommended by `CABaRet` (we remind that cached contents are placed in the top of the list). We can observe that the CHR is always more than 55% (and up to 70%), which strongly indicates that users select the `CABaRet` recommendations, when they are provided.

Figure 13 shows in more detail the preference of videos with respect to the `CABaRet` recommendations. Specifically, the y-axis is the $CHR(x)$, i.e., the CHR in sessions where the `CABaRet` recommendations included x cached videos (x-axis). The continuous line shows the $CHR(x)$ observed in our experiments, while the dashed lines correspond to an hypothesized *uniform* selection of contents (i.e., the user selects randonmly one of the 5 presented recommendations), and *Zipf* selection of contents. The main observation is that users tend to prefer the `CABaRet` recommendations presented in the top of the list; this behavior that has been previously reported for the YouTube service, does not seem to be affected by the fact that the recommendations are nudged towards cached videos. This indicates that *using a carefully designed QoS-aware RSs does not have a negative impact on user preferences*.

Finally, these results also provide and insights on the tuning of `CABaRet`: The decrease in the CHR is mainly due to sessions where `CABaRet` did not find any cached video in the related list $\mathcal{L}$, and not due to the number of cached videos in the recommendation list. Hence, for these sessions (with 0 cached recommendations) we could tune `CABaRet` to search in larger depth/width for cached contents; even finding one such content and placing it in the top of the recommendation list, would lead to increased CHR.

Key finding: The recommendations generated by `CABaRet` are perceived as high-quality by users.

The results presented above, demonstrate that applying an algorithm like `CABaRet` in practice, could indeed lead to performance gains, since users are willing to select the nudged recommendations towards cached videos. Apart from the network benefits, in this last part of our analysis, we investigate whether the `CABaRet` recommendations satisfy the user: Do the users select the nudged recommendations because the find them appealing or because they do not have a (much) better alternative? Are they satisfied by the recommended videos?

In the experiments, we asked the users to provide ratings for the relevance of the recommendation list (QoR) and their interest in the watched video. Figure 15 shows the average rating for QoR (y-axis) in sessions where x out of the 5 recommendations are for cached videos. We observe that the users do not significantly differentiate, in terms of QoR, between the initial YouTube recommendations ($x = 0$) from the `CABaRet` recommendations ($x > 0$). This clearly shows that the nudged recommendations are not perceived as intrusive by the user.

In addition to this, we investigate whether the users ultimately liked the video they selected to watch (and, e.g., were not misled by the recommendation). Table 4 shows the distribution of the *Interest* ratings for the cached and non-cached videos. The interest for contents from initial YouTube recommendations (i.e., all non-cached videos) is not significantly different than the interest in the cached videos that the users watched. This further supports our arguments and provide experimental evidence that (a) `CABaRet` can find high-quality recommendations, and (b) nudging recommendations towards cached video, does not have a significant negative impact in user interest.

TABLE 4: Percentage of responses per *Interest* rating for the cached and non-cached videos.

	Rating of *Interest*		
	1-2★	3★	4-5★
Non-cached videos	24%	19%	57%
Cached videos	25%	18%	57%

5 Related Work

The joint network and recommendations paradigm has been recently introduced, in the context of soft cache hits [7], [9] or network-friendly recommendations [8], [12], aiming to

10. While for this analysis we assume a fraction of the 500 cached contents, in the experiments `CABaRet` considered in its recommendations the initial set of the 500 contents. This means that the presented results may underestimate the best CHR that could achieved by `CABaRet`

jointly design the content caching policy and the recommendation policy in order to achieve higher cache hit rates. The promising gains in the caching efficiency (which comes "for free" from a technology point of view, e.g., without extra investment in equipment or new communication technologies) demonstrated by these early works, motivated more work on the topic [10], [11], [13]–[17], [20]–[26], [48], [50].

The majority of related works considers *cache-aware* recommendations in mobile edge caching [7]–[17] to improve the cache hit ratio by optimizing the recommendation and/or caching policies. However, the same principles can easily generalize to *network-aware* recommendations, where each content can be delivered by the network with a given cost or quality [23]. Other aspects considered in literature include coded caching [11], broadcast communications with coded transmissions [20], [21], [26], the extra dimensions of user association to small base stations [24], or swarming systems [51]. A similar concept is similarity caching [14], [17], which can have more generic applications than multimedia services (e.g., machine learning tasks).

Our work is complementary to previous works that study techniques for optimizing the network performance. To our best knowledge, all existing studies have evaluated the performance in simulation setups. On the contrary, we focus on realistic evaluations of the joint network and recommendations. Our goal was to (i) enable researchers perform realistic evaluations of their solutions, (ii) verify that the claimed performance gains can hold also in practice (i.e., in real setups), (iii) provide evidence for the assumptions made by previous works that users will be willing to follow "nudged" network-aware recommendations.

Finally, while in this paper we focused on the YouTube case, the proposed methodology is generalizable to other services and settings. The simplicity of the `CABaRet` algorithm makes it easily implementable, without this having a negative effect on performance, as shown by our results or, e.g., the evaluation in [16] for short-video services scenarios where `CABaRet` achieves comparable performance to state-of-the-art schemes [16], [43].

6 Conclusion

In this paper, we proposed a methodology that enables to evaluate joint network and recommendation techniques in realistic setups, by leveraging available information from real recommendation systems. Enabling realistic evaluations of previous or future works can be important for fine-tuning the parameters and assumptions of the proposed solutions, as well as provide insights for potential practical challenges.

Our results on the YouTube video service showed that the significant gains that have been indicated in related literature, can be achieved in practice as well. This is a positive message for the feasibility and benefits of the joint network and recommendations paradigm. To further strengthen this message, we conducted experiments with real users to investigate the feasibility from the user perspective; our findings are the first to provide experimental evidence that network-aware recommendations can be perceived as non-intrusive by users (a major assumption in related work).

We believe that our findings can motivate further research on the topic. For instance, large-scale experiments with users or measurements in real network conditions could provide useful results and insights for the design of operational network-aware recommendation systems.

Acknowledgements

This research is co-financed by Greece and the European Union (European Social Fund- ESF) through the Operational Programme "Human Resources Development, Education and Lifelong Learning" in the context of the project "Reinforcement of Postdoctoral Researchers - 2nd Cycle" (MIS-5033021), implemented by the State Scholarships Foundation (IKY).

References

[1] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing a key technology towards 5g," *ETSI White Paper No. 11*, 2016.

[2] 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.

[3] B. Bharath, K. G. Nagananda, and H. V. Poor, "A learning-based approach to caching in heterogenous small cell networks," *IEEE Transactions on Communications*, vol. 64, no. 4, pp. 1674–1686, 2016.

[4] J. Du, C. Jiang, E. Gelenbe, H. Zhang, Y. Ren, and T. Q. Quek, "Double auction mechanism design for video caching in heterogeneous ultra-dense networks," *IEEE Transactions on Wireless Communications*, vol. 18, no. 3, pp. 1669–1683, 2019.

[5] 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.

[6] M. Dehghan, B. Jiang, A. Seetharam, T. He, T. Salonidis, J. Kurose, D. Towsley, and R. Sitaraman, "On the complexity of optimal request routing and content caching in heterogeneous cache networks," *IEEE/ACM Transactions on Networking*, vol. 25, no. 3, pp. 1635–1648, 2016.

[7] 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)*, 2016.

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

[9] P. Sermpezis, T. Giannakas, T. Spyropoulos, and L. Vigneri, "Soft cache hits: Improving performance through recommendation and delivery of related content," *IEEE Journal on Selected Areas in Communications*, 2018.

[10] T. Giannakas, P. Sermpezis, and T. Spyropoulos, "Show me the cache: Optimizing cache-friendly recommendations for sequential content access," in *Proc. IEEE WoWMoM*, 2018.

[11] B. Zhu and W. Chen, "Coded caching with joint content recommendation and user grouping," in *Proc. IEEE GLOBECOM*, 2018.

[12] L. E. Chatzieleftheriou, M. Karaliopoulos, and I. Koutsopoulos, "Jointly optimizing content caching and recommendations in small cell networks," *IEEE Transactions on Mobile Computing*, vol. 18, no. 1, pp. 125–138, 2019.

[13] M. Costantini, T. Spyropoulos, T. Giannakas, and P. Sermpezis, "Approximation guarantees for the joint optimization of caching and recommendation," in *Proc. IEEE ICC*, 2020.

[14] M. Garetto, E. Leonardi, and G. Neglia, "Similarity caching: Theory and algorithms," in *Proc. IEEE INFOCOM*, 2020.

[15] D. Tsigkari and T. Spyropoulos, "User-centric optimization of caching and recommendations in edge cache networks," in *Proc. IEEE WoWMoM*, 2020.

[16] S. Li, "Leveraging recommendation systems for improving caching emerging short video in content delivery network," *Transactions on Emerging Telecommunications Technologies*, p. e4117, 2020.

[17] J. Zhou, O. Simeone, X. Zhang, and W. Wang, "Adaptive offline and online similarity-based caching," *IEEE Networking Letters*, 2020.

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

[19] C. 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.

[20] Z. Lin and W. Chen, "Joint pushing and recommendation for susceptible users with time-varying connectivity," in *Proc. IEEE GLOBECOM*, 2018.

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

[22] K. Qi, B. Chen, C. Yang, and S. Han, "Optimizing caching and recommendation towards user satisfaction," in *Int. Conf. on Wireless Communications and Signal Processing*. IEEE, 2018.

[23] T. Giannakas, T. Spyropoulos, and P. Sermpezis, "The order of things: Position-aware network-friendly recommendations in long viewing sessions," in *Proc. WiOpt*, 2019.

[24] L. Chatzieleftheriou, G. Darzanos, M. Karaliopoulos, and I. Koutsopoulos, "Joint user association, content caching and recommendations in wireless edge networks," *PER*, vol. 46, no. 3, pp. 12–17, 2019.

[25] S. Gupta and S. Moharir, "Effect of recommendations on serving content with unknown demand," *ACM TOMPECS*, vol. 4, no. 1, p. 4, 2019.

[26] Z. Lin and W. Chen, "Content pushing over multiuser miso downlinks with multicast beamforming and recommendation: A cross-layer approach," *IEEE Transactions on Communications*, vol. 67, no. 10, pp. 7263–7276, 2019.

[27] 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.

[28] S. Kastanakis, P. Sermpezis, V. Kotronis, and X. Dimitropoulos, "Cabaret: Leveraging recommendation systems for mobile edge caching," in *Proc. of the ACM MECOMM workshop (ACM SIGCOMM workshops)*, 2018, pp. 19–24.

[29] P. Covington, J. Adams, and E. Sargin, "Deep neural networks for YouTube recommendations," in *Proc. ACM RecSys*, 2016.

[30] C. Liang and F. R. Yu, "Wireless network virtualization: A survey, some research issues and challenges," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 1, pp. 358–380, 2015.

[31] Akamai, "Mobile Optimization," 2020, www.akamai.com/us/en/resources/mobile-optimization.jsp.

[32] "Vimeo Developer API," 2020, https://developer.vimeo.com/api/reference/videos#get_related_videos.

[33] "Twitch Developer API," 2020, https://dev.twitch.tv/docs/v5/reference/clips#get-top-clips.

[34] "Dailymotion Developer API," 2020, https://developer.dailymotion.com/api/#video-related_videos_algorithm-filter.

[35] "Spotify for Developers," 2020, https://developer.spotify.com/documentation/web-api/reference/browse/get-recommendations/.

[36] "Netflix (Unofficial Netflix Online Global Search)," 2020, https://unogs.com/.

[37] "Tracking Exposed project," 2020, https://tracking.exposed/.

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

[39] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: Item-to-item collaborative filtering," *IEEE Internet computing*, vol. 7, no. 1, pp. 76–80, 2003.

[40] H. Steck, "Item popularity and recommendation accuracy," in *Proc. ACM RecSys*, 2011.

[41] H. Abdollahpouri, R. Burke, and B. Mobasher, "Controlling popularity bias in learning-to-rank recommendation," in *Proc. ACM RecSys*, 2017.

[42] A. Vall, M. Quadrana, M. Schedl, and G. Widmer, "Order, context and popularity bias in next-song recommendations," *Intl. Journal of Multimedia Information Retrieval*, vol. 8, no. 2, pp. 101–113, 2019.

[43] Y. Zhang, P. Li, Z. Zhang, B. Bai, G. Zhang, W. Wang, and B. Lian, "Challenges and chances for the emerging short video network," in *Proc. IEEE INFOCOM workshops*, 2019.

[44] D. 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*, vol. 11, no. 4, p. 48, 2015.

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

[46] Google, "Google trends api," 2019, https://www.npmjs.com/package/google-trends-api.

[47] S. Kastanakis, "CABaRet experimental testbed and datasets," 2020, https://github.com/kastanakis/CABaRet/tree/master/CABaRetExperimentalTestbed.

[48] P. Sermpezis, S. Kastanakis, J. I. Pinheiro, F. Assis, M. Nogueira, D. Menasché, and T. Spyropoulos, "Towards qos-aware recommendations," *Proc. ACM RecSys workshops (CARS workshop)*, 2020.

[49] YouTube, "Youtube api," 2019, https://developers.google.com/youtube/.

[50] F. Assis and et al., "Recomendação de conteúdo e qoe: Um experimento quantificando o papel da qos nas preferências por vídeos," in *SBC WPERFORMANCE workshop*. SBC, 2019.

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

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

[53] A. A. Bian, J. M. Buhmann, A. Krause, and S. Tschiatschek, "Guarantees for greedy maximization of non-submodular functions with applications," in *Proc. ICML*, 2017, pp. 498–507.

Savvas Kastanakis received the diploma in Computer Science and Engineering from the Computer Science Department, University of Crete. He is currently a PhD student at the School of Computing and Communications, Lancaster University, UK. His research interests include: Internet Measurements, Internet of Things, Network Security.

Pavlos Sermpezis received the Diploma in Electrical and Computer Engineering from the Aristotle University of Thessaloniki (AUTH), Greece, and a PhD in Computer Science and Networks from EURECOM, Sophia Antipolis, France. He was a post-doctoral researcher at FORTH, Greece, and currently is a researcher at the Informatics Dept. at AUTH, Greece. His main research interests are in modeling and performance analysis of communication networks, network measurements, data science.

Vasileios Kotronis received a Diploma in Electrical and Computer Engineering from the National Technical University of Athens, Greece and a PhD in Information Technology and Electrical Engineering from ETH Zurich, Switzerland. He is currently a post-doctoral researcher at FORTH, Greece. His main research interests include: Internet routing and measurements, software defined networking, and network security.

Daniel Sadoc Menasché received the Ph.D.degree in computer science from the University of Massachusetts, Amherst, in 2011. He is currently an Assistant Professor with the Computer Science Department, Federal University of Rio de Janeiro, Brazil. His research interests are in modeling, analysis, security, and performance evaluation of computer systems. He was a recipient of the best paper awards at GLOBECOM 2007, CoNEXT 2009, INFOCOM 2013, and ICGSE2015. He is currently an Affiliated Member of the Brazilian Academy of Sciences.

Thrasyvoulos Spyropoulos received the Diploma in Electrical and Computer Engineering from the National Technical University of Athens, Greece, and a Ph.D degree in Electrical Engineering from the University of Southern California. He was a post-doctoral researcher at INRIA and then, a senior researcher with the Swiss Federal Institute of Technology (ETH) Zurich. He is currently an Assistant Professor at EURECOM, Sophia-Antipolis. He is the recipient of the best paper award in IEEE SECON 2008, and IEEE WoWMoM 2012.

Appendix A
Caching Optimization under CABaRet

A.1 Problem Formulation and Optimization Algorithm

In the following, we first analytically formulate and study the problem of optimizing the caching policy under CABaRet recommendations, and propose an approximation algorithm with provable performance guarantees.

Let a content catalog $\mathcal{V}$, $V = |\mathcal{V}|$, and a content popularity vector $\mathbf{q} = [q_1, ..., q_V]^T$. Let $\mathcal{L}(v) \subseteq \mathcal{V}$ be the set of contents that are explored by CABaRet (at *line 1*) for a content $v \in \mathcal{V}$.

For some set of cached contents $\mathcal{C} \subseteq \mathcal{V}$, and a content v, CABaRet returns a list of recommendations $\mathcal{R}(v)$ ($|\mathcal{R}(v)| = N$). Therefore, CHR can be expressed as

$$CHR(\mathcal{C}) = \sum_{v \in \mathcal{V}} q_v \sum_{i=1}^{N(\mathcal{C},v)} p_i \tag{4}$$

where $N(\mathcal{C}, v) = \min\{|\mathcal{C} \cap \mathcal{L}(v)|, N\}$, and p_i is the probability for a user to select the i^{th} recommended content.

Then, the problem of optimizing the caching policy (to be jointly used with CABaRet), is formulated as follows:

$$\max_{\mathcal{C}} \; CHR(\mathcal{C}) \quad \text{s.t.}, |\mathcal{C}| \leq C \tag{5}$$

where C is the capacity of the –MEC– cache. We prove the following for the optimization problem of Eq. (5).

Lemma 1. *The optimization problem of Eq.* (5)*: (i) is NP-hard, (ii) cannot be approximated within $1 - \frac{1}{e} + o(1)$ in polynomial time, and (iii) has a monotone (non-decreasing) submodular objective function, and is subject to a cardinality constraint.*

Proof. Items (i) and (ii) of the above lemma, are proven by reduction to the *maximum set coverage* problem, and we prove item (iii) using standard methods (see, e.g., similar proofs in [5], [45]). □

If we design a greedy algorithm that starts from an empty set of cached contents $\mathcal{C}_g = \emptyset$, and at each iteration it augments the set $\mathcal{C}_g$ (until $|\mathcal{C}_g| = C$) as follows:

$$\mathcal{C}_g \leftarrow \mathcal{C}_g \cup \arg\max_{v \in \mathcal{V}} CHR(\mathcal{C}_g \cup \{v\}), \tag{6}$$

then the properties stated in item (iii) satisfy that it holds [52]

$$CHR(\mathcal{C}_g) \geq \left(1 - \frac{1}{e}\right) \cdot CHR(\mathcal{C}^*) \tag{7}$$

where $\mathcal{C}^*$ the optimal solution of the problem of Eq. (5).

Remark: While Eq. (7) gives a lower bound for the performance of the greedy algorithm, in practice greedy algorithms have been shown to perform often very close to the optimal [53].

A.2 Results under Greedy Caching

We investigate the performance when the list of cached contents is selected to optimize the CHR by using the greedy algorithm. We consider both "front-page" and "search bar" video demands.

Efficiency vs. scalability. Calculating the CHR from Eq. (4) requires running a BFS (CABaRet, *line 1*) and generating the lists $\mathcal{L}(v)$, for every content $v \in \mathcal{V}$. In practice, for scalability reasons, the most popular contents (i.e., with high q_i) can be considered by the greedy algorithm in the calculation of the objective function Eq. (4), since those contribute more to the objective function. To demonstrate the involved trade-offs between scalability and performance, we consider two scenarios with synthetic content catalogs of size $|\mathcal{V}|$=1000 and $|\mathcal{V}|$=10000 (where content popularity q_i follows a Zipf(a=1) distribution, and each content is related on average with 10 other contents), and calculate the CHR achieved by CABaRet (N=10, D_{BFS}=2, W_{BFS}=5) when the greedy algorithm considers only a fraction $\mathcal{V}^{'}$ of the entire catalog, $\mathcal{V}^{'} \subseteq \mathcal{V}$, and a cache of size C=10. Table 5 presents the achieved CHR, normalized over the maximum CHR achieved when considering the entire catalog $\mathcal{V}$. We can see that even considering very small fractions of the content catalog in the caching decisions, can still achieve significant performance, while considering a 10% of the content catalog can already achieve 90% and 86% of the maximum performance in the case of $|\mathcal{V}|$=1000 and $|\mathcal{V}|$=10000, respectively. This indicates that CABaRet-like approaches can be an efficient and scalable in real systems with very large content catalogs.

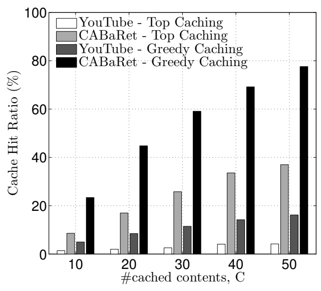
TABLE 5: CHR under caching with the greedy algorithm considering only a fraction of the most popular contents of the catalog, $i \in \mathcal{V}^{'} \subset \mathcal{V}$; values are normalized over the maximum achievable performance.

fraction of the catalog $\frac{\|\mathcal{V}^{'}\|}{\|\mathcal{V}\|}$		0.1%	1%	5%	10%
$\frac{CHR(\mathcal{V}^{'})}{CHR(\mathcal{V})}$	$\|\mathcal{V}\| = 1000$	0.72	0.74	0.89	0.90
	$\|\mathcal{V}\| = 10000$	0.40	0.54	0.82	0.86

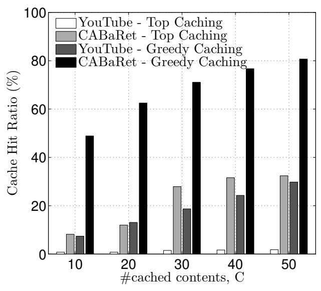
The reason that the greedy algorithm remains efficient even with this simplification, is that any video in the catalog is still candidate to be cached, e.g., a video with low q_i can bring a large increase in the CHR through its association with many popular contents. In fact, in our experiments, for the calculation of Eq. (4), we consider only the 50 most popular videos, for which we set $q_i = \frac{1}{50}$. Nevertheless, in the different scenarios we tested, only 10% to 30% of the cached videos (selected by the greedy algorithm) were also in the top 50 most popular.

Greedy vs. Top caching. In Fig. 16(a), we compare the achieved CHR for "Front Page" video demand, when the cache is populated according to the greedy algorithm of Eq. (6) (*Greedy Caching*) and with the top most popular videos (*Top Caching*). *Greedy caching* always outperforms *top caching*, with an increase in the CHR of around a factor of 2 for uniform video selection (for the Zipf(a=1) scenarios we tested, the CHR values are even higher, and the relative performance is 1.5 times higher). This clearly demonstrates that the gains from joint recommendation and caching [8], [45], are applicable even in simple practical scenarios (e.g., CABaRet & greedy caching). Finally, while *greedy caching* increases the CHR even with regular YouTube recommendations, the CHR is still less than 50% of the CABaRet case with *top caching*. This further stresses the benefits from CABaRet's cache-aware recommendations.

Similar findings can be seen in Fig. 16(b) for scenarios with "Search Bar" video demand. A difference is that in



(a) "Front Page" video demand



(b) "Search Bar" video demand

Fig. 16: CHR vs. # cached contents with `CABaRet` parameters W_{BFS}=50 and D_{BFS}=2, and video demand (a) "Front Page" video demand with p_i ∼*uniform*, and (b) "Search Bar" video demand with p_i ∼*Zipf(a=1)* .

these scenarios the CHR under YouTube recommendations with *greedy caching* is comparable to `CABaRet` recommendations with *top caching*, which indicates that similar performance can be achieved by carefully selecting either only the recommendations (`CABaRet` + *top caching*) or only the caching (YouTube + *greedy caching*). However, when combining both (`CABaRet` + *greedy caching*), increases more than two times the CHR.