Content Caching with Personalized and Incumbent-aware Recommendation: An Optimization Approach

Yi Zhao[1], Zhanwei Yu[1], Qing He[2], and Di Yuan[1]

[1]Department of Information Technology, Uppsala University, Sweden
[2]Department of Information Systems and Technology, Mid Sweden University, Sweden
Emails: {yi.zhao, zhanwei.yu, di.yuan@uu.se}, qing.he@miun.se

***Abstract*—Content recommendation can be tailored by not only personal interests, but also the incumbent content, namely the content that a user is currently viewing. Incumbent-aware recommendation adds a new dimension to optimizing content caching. We study this optimization problem subject to user satisfaction constraints. We prove the problem's NP-hardness, and present an integer linear programming formulation that enables global optimality for small-scale instances. On the algorithmic side, we first present a polynomial-time algorithm that delivers the global optimum of the recommendation sub-problem, by leveraging the problem's inherent graph structure. Next, we propose a fast, alternating algorithm for the overall problem. Numerical results using synthesized and real-world data show the close-to-optimal performance of the proposed algorithm.**

***Index Terms*—Caching, content delivery networks, recommendation.**

I. Introduction

Edge caching is expected to play an important role in future communication systems [1]. To high efficiency, network providers tend to cache the most popular contents frequently requested by users. The popularity of contents is determined by many factors, among which recommendation systems [2] form a major one. For example, it has been reported that up to 80% of requests on Netflix originate from recommendation [3].

It is natural to seek the potential of recommendation, as a tool of reshaping content popularity, to improve the cache efficiency. In practice, the appearance of content delivery networks (CDNs) [4], such as Netflix Open Connect [5] and Google Global Cache [6], enable a single entity to jointly optimize caching and recommendation, instead of the conventional scheme of addressing the two aspects separately by network providers and content providers respectively.

Most content providers (such as Youtube, Netflix, Bilibili, etc.) employ two recommendation schemes. One is to recommend contents based on the long-term interest of users. Another is to recommend relevant contents with respect to the content that is currently being viewed by a user, i.e., the incumbent content, based on both long-term and short-term interests. We call the latter incumbent-aware recommendation. Apparently, this type of recommendation adds more dynamics to the system.

In recent years, joint optimization of edge caching and recommendation is gaining growing interest. The most common performance objective is the overall probability that the cached items are requested; this is also referred to as the cache-hit-ratio in some papers. For personalized content recommendation, its impact on probability distribution of context requests has been modeled in [7], [8], based on user preference and content features. In the resulting cache-hit-ratio maximization problem, cache capacity, recommendation list length, and limit on preference distortion are accounted for, and an alternating optimization algorithm has been proposed. This framework has been adopted in many follow-up works, such as [9]–[11]. In [9], a variation of the above problem with multiple Internet content providers is considered. The study in [10] explores the optimization problem with multiple caches, while [11] addresses the joint optimization of caching and recommendation in device-to-device (D2D) scenarios. In addition, the works in [12]–[14] emphasize on machine learning methods which can assist on solving the joint cache and recommendation optimization problem.

In this paper, we consider the joint optimization of caching and incumbent-aware recommendation for improving the caching efficiency, while accounting for user satisfaction in respect of recommendation. The caching efficiency is represented by the overall probability that the users request cached contents. The specific contributions consist in the following.

- We develop a system model characterizing caching and incumbent-aware recommendation; the latter is a novel feature of the system model under consideration, in comparison to previous works.
- We approach the problem via a graph perspective, that enables to analyze problem complexity. A rigorous proof is derived for the problem's NP-hardness.
- We formulate the problem using integer linear programming (ILP), to approach optimality for small-scale instances.
- For the recommendation sub-problem, we present a polynomial-time algorithm guaranteeing global optimality. We then propose a fast, alternating algorithm for the overall problem.
- We assess the proposed algorithms using synthetic datasets as well as real-world datasets. Simulation results show the strength of the algorithm in terms of high cache efficiency.

ISBN 978-3-903176-49-2

II. System Model

Consider a set of users denoted by $\mathcal{K} = \{1, \ldots, K\}$ and a base station (BS) equipped with a cache of capacity C. The contents of interest form set $\mathcal{I} = \{1, \ldots, I\}$[1]. Content $i \in \mathcal{I}$ is of size s_i. We use binary variable x_i to represent whether or not content i is cached.

Content recommendation is both personalized and incumbent-aware. We use binary variable y_{ji}^k to indicate if content i is in the recommendation list for user k with incumbent content j. The recommended contents form a list, of which the length limit is denoted by B.

For content i, the probability of being requested by user k originates from the user's own preference as well as incumbent-aware recommendation. For the former, we denote by p_i^k ($p_i^k \in [0,1]$, $\sum_{i\in\mathcal{I}} p_i^k = 1, \forall k \in \mathcal{K}$) the probability that user k directly requests content i without recommendation. The corresponding probability due to recommendation is $\sum_{j\in\mathcal{I}\setminus i} y_{ji}^k p_j^k p_{ji}^k$, where p_{ji}^k ($p_{ji}^k \in [0,1]$) is the probability that user k requests recommended content i, provided that the incumbent content is j. This probability is related to the preference of user k as well as the correlation between contents i and j.

Requesting different contents in a recommendation list by a user is regarded as independent events. Moreover, the likelihood accessing recommended contents varies by user. Hence we do not normalize p_{ji}^k on i, for given j and k. Therefore, the popularity of content i, denoted by q_i, i.e., the overall probability that i is requested, is the following function of recommendation decision $\boldsymbol{y}$.

$$q_i(\boldsymbol{y}) = \sum_{k\in\mathcal{K}} \left(p_i^k + \sum_{j\in\mathcal{I}\setminus i} y_{ji}^k p_j^k p_{ji}^k \right). \tag{1}$$

We maximize the overall popularity of the cached contents, also called cache efficiency, as formulated below.

$$\max_{\boldsymbol{x},\boldsymbol{y}} \sum_{i\in\mathcal{I}} x_i q_i(\boldsymbol{y}) \tag{2}$$

One can observe from (2) that recommending uncached contents gives no contribution to the objective. Thus, without additional restrictions in the system model, the recommendation system will recommend only cached contents. However, cached contents are not necessarily those that an individual user is most interested in, and hence such a recommendation strategy may lead to poor user experience. To address this, we consider two types of user satisfaction constraints.

1) **Content-level satisfaction constraint**: We impose a threshold on the correlation of the recommended contents and the incumbent one. Denote the threshold of user k by β^k. The system will only consider recommending contents with correlation meeting the threshold, namely, for user k and incumbent content j, the subset of contents $\{i \in \mathcal{I}, i \neq j : p_j^k p_{ji}^k \geq \beta^k\}$. In the sequel, we use $\mathcal{D}_j^k$ to denote this set.
2) **List-level constraint constraint**: At the list level, we introduce a constraint to ensure the overall correlation between the incumbent and the recommended contents (no matter cached or not) is not too low, as stated below.

$$\frac{\sum_{i\in\mathcal{D}_j^k} y_{ji}^k p_j^k p_{ji}^k}{u_j^k} \geq \alpha^k, \quad j \in \mathcal{I}, k \in \mathcal{K}, \tag{3}$$

where u_j^k is the sum of the $\min\{B, |\mathcal{D}_j^k|\}$ highest values of $\mathcal{D}_j^k$, and $\alpha^k \in [0,1]$ is a threshold. The left-hand side of (3) relates the correlation of recommended contents to these being most relevant to the incumbent, to represent the overall satisfaction level of the list for user k.

The optimization problem of caching and incumbent-aware recommendation (CIAR) has the objective function (2), subject to the above constraints for user satisfaction, as well as cache capacity limit C and list length limit B.

III. Structural Analysis

A. A Graph Perspective for CIAR

Let us first approach CIAR with a graph perspective. As will be clear later on, this approach will be instructive for structural insights, analysis of problem complexity, as well as algorithm development. We construct a content graph $G = (V, E)$ that is a multi-graph allowing multiple arcs between nodes, with V and E being the sets of nodes and arcs respectively. Each node i represents a content in $\mathcal{I}$ with two labels: content size s_i and weight $\sum_{k\in\mathcal{K}} p_i^k$. For each user k, there is an arc from node j to node i, denoted by $(j,i)^k$, with weight $p_j^k p_{ji}^k$, if $i \in \mathcal{D}_j^k$. Denote by $G^k = (V, E^k)$ the sub-graph of k, consisting in node set V and the arcs defined for user k.

CIAR is equivalent to solving a node and arc selection problem in G. From (1) and (2), we obtain (4) below, by which we can see that the values of $\boldsymbol{x}$ and $\boldsymbol{y}$ respectively correspond to the selection of nodes and arcs. Namely, having $x_i = 1$ implies to select content node i, and having $y_{ji}^k = 1$ implies to select arc $(j,i)^k$. Subsequently, we also refer to the selected and unselected nodes as "cached nodes" and "uncached nodes", respectively.

$$\max_{\boldsymbol{x},\boldsymbol{y}} \sum_{i\in\mathcal{I}} \sum_{k\in\mathcal{K}} x_i p_i^k + \sum_{j\in\mathcal{I}} \sum_{k\in\mathcal{K}} \sum_{j\in\mathcal{D}_j^k} x_i y_{ji}^k p_j^k p_{ji}^k \tag{4}$$

In (4), the first term is the total weight of selected nodes, and the second term is the total weight of selected arcs that end at selected nodes. Note that a selected arc with an unselected tail node will not contribute to the function value, since $x_i y_{ji}^k p_j^k p_{ji}^k$ will be non-zero only if both $x_i = 1$ and $y_{ji}^k = 1$.

Among the constraints of CIAR, that for content-level user satisfaction is accounted for in the construction of graph G. The other constraints imply the following restrictions to node and arc selection:

- **Cache capacity**: The total size of selected nodes may not exceed C.

[1]The set of contents can be obtained via for example collaborative filtering [15] and content-based models [16].

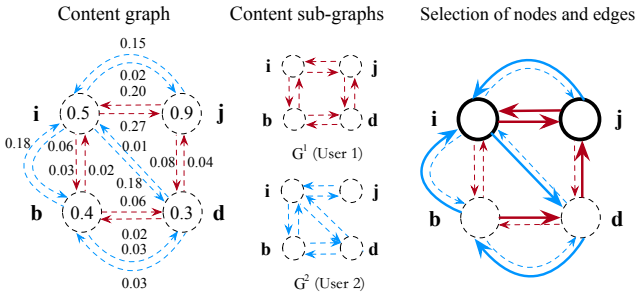


Figure 1: An illustration of the graph and an optimal solution for an instance of CIAR with two users (k in $\{1, 2\}$) and four contents (i, j, b, and d), with $C = 2$, $B = 1$, $\alpha^k = 0.8$, $\beta^k = 0.02$, $s_i = s_j = 1$, and $s_b = s_d = 2$.

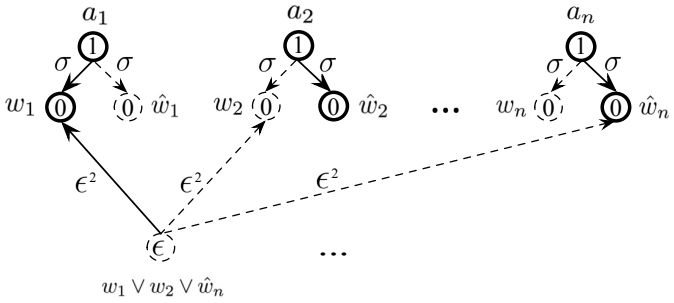


Figure 2: An illustration of reduction for the literals and one example clause, and the solution structure of the CIAR instance.

- **Recommendation list length**: The number of selected arcs originating from any node in any content sub-graph may not exceed B.
- **List-level user satisfaction**: The sum weight of selected arcs originating from node j in content sub-graph G^k is no less than $\alpha^k u_j^k$.

Fig. 1 gives an illustration of G. The arcs corresponding to the two users are indicated by red and blue colors respectively. The optimal selection of nodes and arcs is indicated by solid circles and lines. One can observe that, to account for list-level user satisfaction, an arc may be selected because it has higher correlation with the incumbent content, even if it has less weight than another arc from the same node. This is the reason for selecting arc (b, d) instead of (b, i) for user one.

B. Problem Complexity

Using the notion of graph representation of CIAR, we state and formally prove its NP-hardness.

Theorem 1. *CIAR is NP-hard, even for the special case where $|\mathcal{K}| = 1$ (i.e., one user), $B = 1$ (i.e., at most one recommended content), the content size s_i are uniform $\forall i \in \mathcal{I}$, and no list-level user satisfaction constraints are present.*

Proof. We use a polynomial-time reduction from the 3-satisfiability (3-SAT) problem. The 3-SAT problem [17] is NP-complete. Consider a 3-SAT instance with n boolean variables $w_1, w_2, ..., w_n$, and m clauses. Denote by $\hat{w}_i$ the negation of w_i. A literal refers to a variable or its negation. Each clause is composed by a disjunction of exactly three distinct literals, e.g., $w_1 \vee w_2 \vee \hat{w}_n$. The problem is to determine whether there exists an assignment of true/false values to the variables, such that all clauses hold true.

For any 3-SAT instance, we construct an instance of CIAR with $B = 1$, $|\mathcal{K}| = 1$ (superscript k can thus be dropped), $\alpha = 0$, $\beta = 0$, and $s_i = 1$, $\forall i \in \mathcal{I}$. For clarity, we define the CIAR instance using the graph representation. There is a pair of nodes (w_i and $\hat{w}_i$) and an additional node a_i in respect of each boolean variable, and there is a node corresponding to each clause. Hence there are $3n+m$ contents in total. The arcs that are present, as well as the node and arc weights are given in Fig. 2, where $\sigma = 0.5$, $\epsilon^2 = \sigma/(m * N^2)$ (ϵ^2 is the square of ϵ), and N is a positive number satisfying $N^2 > 1/\sigma$ (hence $N > 1$). Let cache capacity $C = 2n$. With the given weights, one can verify that the following conditions hold: $\sigma + m\epsilon^2 < 1$, $\epsilon < 1$, $m\epsilon^2 < \sigma$, and $\epsilon < \sigma$.

We claim and prove the following characteristics of an optimal solution of the CIAR instance.

1) Node a_i is selected, $\forall i \in \mathcal{I}$. Suppose the opposite and assume some node a_i is not selected. Selecting a_i instead and possibly deleting a literal node or a clause node (depending on whether or not cache capacity has been reached) will cause at least $1-(\sigma+m\epsilon^2)$ improvement of the objective function value, because $\epsilon < \sigma + m\epsilon^2 < 1$, contradicting the solution's optimality.
2) Exactly one of w_i and $\hat{w}_i$ is selected, $\forall i \in \mathcal{I}$. We prove this again using contradiction. Suppose for any i, w_i and $\hat{w}_i$ are both selected. Then there must exist another pair w_j and $\hat{w}_j$ that are not selected. Note that a_i and a_j are both present by 1) above. Without loss of generality, suppose w_i and w_j are the nodes with lower sum weight of selected arcs ending at them, of the two pairs respectively. In this case, arcs (a_i, w_i) and (a_j, w_j) will not be selected at optimum because $m\epsilon^2 < \sigma$. By deselecting node w_i and selecting node $\hat{w}_j$ and arc $(a_j, \hat{w}_j)$, the objective value will improve by at least $\sigma - m\epsilon^2$. Hence the contradiction.
3) The arc between a_i and the selected node among w_i and $\hat{w}_i$ is selected, $\forall i \in \mathcal{I}$. This is because, if arc (a_i, w_i) is selected but node w_i itself is not selected, then deselecting arc (a_i, w_i) and adding arc $(a_i, \hat{w}_i)$ will increase the objective value by σ.

Fig. 2 illustrates the structure of the optimum with respect to the above conditions. Next, observe that, without accounting for selected arcs emanating from the clause nodes, the objective function value, by 1)–3), equals exactly $n(\sigma + 1)$. For the m clause nodes, at most one of the outgoing arcs may be selected because $B = 1$. Moreover, selecting such an arc contributes to the objective function by σ^2, only if the arc ends at a w-node that is selected (i.e., cached). Therefore, if the optimal solution reaches a total objective function value of $n(\sigma + 1) + m\epsilon^2$, the answer to the 3-SAT instance is yes.

Conversely, it is easy to see that if the 3-SAT instance is satisfiable, this value will be reached for the CIAR instance at optimum, and the theorem follows. □

IV. Integer Linear Programming

Problem CIAR can be formulated using integer linear programming (ILP). This allows for global optimality for small-scale instances, using cutting-edge optimization solvers such as CPLEX [18] and GUROBI [19]. Thus for such instances the results of ILP enable accurate performance evaluation of any sub-optimal algorithm.

We introduce a new binary variable z^k_{ji} to represent the product $x_i y^k_{ji}$ at optimum. The ILP reformulation is given below.

$$\max_{\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}} \sum_{i\in\mathcal{I}}\sum_{k\in\mathcal{K}} x_i p^k_i + \sum_{j\in\mathcal{I}}\sum_{k\in\mathcal{K}}\sum_{i\in\mathcal{D}^k_j} z^k_{ji} p^k_j p^k_{ji} \tag{5a}$$

$$\text{s. t.} \quad \sum_{i\in\mathcal{I}} s_i x_i \le C \tag{5b}$$

$$\sum_{i\in\mathcal{D}^k_j} y^k_{ji} \le B, \quad j\in\mathcal{I}, k\in\mathcal{K} \tag{5c}$$

$$\sum_{i\in\mathcal{D}^k_j} y^k_{ji} p^k_j p^k_{ji} \ge \alpha^k u^k_j, \quad j\in\mathcal{I}, k\in\mathcal{K} \tag{5d}$$

$$z^k_{ji} \le y^k_{ji}, \quad k\in\mathcal{K}, j\in\mathcal{I}, i\in\mathcal{D}^k_j \tag{5e}$$

$$z^k_{ji} \le x_i, \quad k\in\mathcal{K}, j\in\mathcal{I}, i\in\mathcal{D}^k_j \tag{5f}$$

$$x_i, y^k_{ji}, z^k_{ji} \in \{0,1\}, \quad k\in\mathcal{K}, j\in\mathcal{I}, i\in\mathcal{D}^k_j \tag{5g}$$

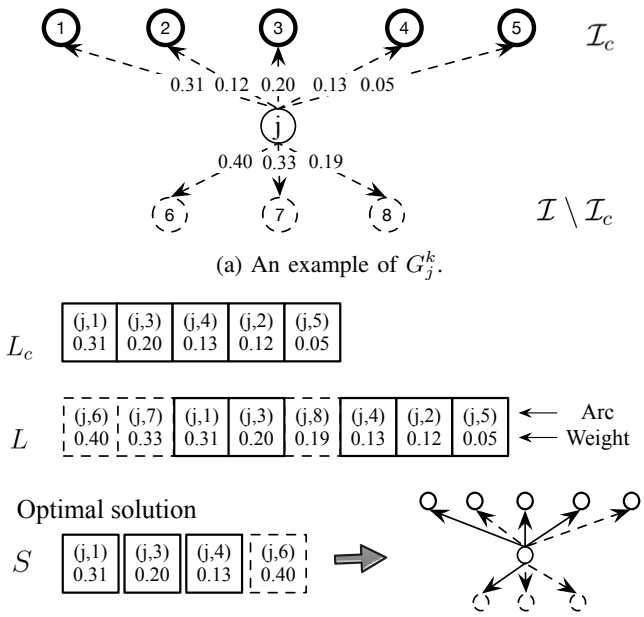
The two components of (5a) are the probability of requesting cached contents directly by the users and via recommendation, respectively. This is a linearized formulation of (2). Inequalities (5b) state cache capacity, whereas (5c) poses the length limit on recommendation lists. Constraints (5d) formulate the list-level user satisfaction requirement. By (5e) and (5f), z^k_{ji} can be one only if $x_i = 1$ (i.e., content i is cached) and $y^k_{ji} = 1$ (i.e., content i is recommended for user k and incumbent content j).

V. Optimization Algorithm

Observing ILP formulation (5), CIAR can be decomposed into two sub-problems. The first is the *recommendation problem*, which amounts to optimizing the recommendation variables $\boldsymbol{y}$ under given caching $\boldsymbol{x}$. The second is to optimize caching with fixed recommendation, which we refer to as the *caching problem*.

A. The Recommendation Problem

We denote by $\mathcal{I}_c = \{i \in \mathcal{I} : x_i = 1\}$ the set of cached contents. Since $\mathcal{I}_c$ is known, the sum of $x_i p^k_i$ in (5a) is fixed. Hence the recommendation problem reads as follows.



(a) An example of G^k_j.

(b) The initial state of L and L_c, and the optimal solution S (and the corresponding $\boldsymbol{y}$) for G^k_j obtained by Algorithm 1, with $|B| = 4$, $\alpha^k = 0.8$, and $\beta^k = 0.02$. In L, L_c and S, the elements corresponding to the arcs pointing to the cached nodes are indicated by solid rectangles, while others are indicated by dashed rectangles.

Figure 3: An example sub-graph and an optimal solution.

$$\max_{\boldsymbol{y}} \sum_{i\in\mathcal{I}_c\cap\mathcal{D}^k_j} y^k_{ji} p^k_j p^k_{ji} \tag{6a}$$

$$\textbf{s.t.} \quad \sum_{i\in\mathcal{D}^k_j} y^k_{ji} \le B, \quad j\in\mathcal{I}, k\in\mathcal{K} \tag{6b}$$

$$\sum_{i\in\mathcal{D}^k_j} y^k_{ji} p^k_j p^k_{ji} \ge \alpha^k u^k_j, \quad j\in\mathcal{I}, k\in\mathcal{K} \tag{6c}$$

$$y^k_{ji} \in \{0,1\}, \quad i\in\mathcal{D}^k_j, j\in\mathcal{I}, k\in\mathcal{K} \tag{6d}$$

We analyze (6) from the graph perspective introduced in Section III. Let us denote by $G^k_j = (V, E^k_j)$ a sub-graph of G^k, where E^k_j is the set of arcs emanating from node j. In Fig. 3a we illustrate an example of G^k_j, where cached nodes are indicated by solid circles, and the arcs in G^k_j are classified into those ending at cached nodes and those ending at uncached nodes. The optimization task is to select at most B arcs starting from j in G^k_j so as to maximize the sum weight of selected arcs ending at cached nodes, subject to the list-level user satisfaction constraints (6c) on all selected arcs.

We denote by L the list of all arcs in G^k_j, sorted by their weights in descending order. Denoting by L_c a sublist of L containing arcs ending at cached nodes, in the same order as in L, we prove the following lemma.

Lemma 2. ***(Necessary optimality condition for*** (6)*): If an arc in L_c is present at optimum, then all the arcs in L_c with strictly higher weights than this arc are present at optimum as well.***

Proof. Suppose at optimum, arc (j,i) in L_c is present, while arc (j,k) with a strictly higher weight in L_c is not. We construct a new solution by including (j,k), removing (j,i), and keeping the remaining arcs. With the new solution, both the objective value and the list-level satisfaction are improved, and (6b)-(6c) are satisfied. This contradicts the assumption of optimality and hence the conclusion. □

We remark that the weights of elements in L_c may not be strictly decreasing as ties may occur. In this case multiple optimal solutions may exist. To ease the presentation, we define the sum function $g(A,t) = \sum_{i=1}^{t} A(i)$, where $A(i)$ is the i^{th} element in list A and $1 \leq t \leq |A|$. Denoting by S_c the set of the selected arcs in L_c in a solution of (6), we show Lemma 3.

Lemma 3. ***(Feasibility condition for*** (6)*):*** *For any feasible solution of* (6)*, $g(L_c,|S_c|)+g(L\setminus S_c,\min\{B,|L|\}-|S_c|) \geq \alpha^k u_j^k$ holds.*

Proof. For any solution of (6), the list-level satisfaction as defined in (6c) consists of the sum weight of the elements in S_c and that of selected elements in $L\setminus S_c$. By constraint (6b), at most $\min\{B,|L|\}-|S_c|$ elements are in $L\setminus S_c$. The use of min-operator is because the total number of relevant contents with respect to incumbent content j is possibly less than B. Due to the descending order of both L_c and $L\setminus S_c$, selecting the first $|S_c|$ elements in L_c and the first $\min\{B,|L|\}-|S_c|$ elements in $L\setminus S_c$ will result in the maximum achievable list-level satisfaction $g(L_c,|S_c|)+g(L\setminus S_c,\min\{B,|L|\}-|S_c|)$. Hence for any feasible solution satisfying (6c), the inequality $g(L_c,|S_c|)+g(L\setminus S_c,\min\{B,|L|\}-|S_c|) \geq \alpha^k u_j^k$ holds. □

Next we derive the following sufficient optimality condition for optimum of (6).

Lemma 4. ***(Sufficient optimality condition for*** (6)*):*** *For any feasible solution of* (6) *that satisfies the necessary condition defined in Lemma 2, if $g(L_c,|S_c|+1)+g(L\setminus S_c^+,\min\{B,|L|\}-|S_c|-1) < \alpha^k u_j^k$, where S_c^+ denotes the set of the first $|S_c|+1$ elements in L_c, then the solution is optimal.*

Proof. We consider a solution (called Solution 1) satisfying the conditions in the statement of Lemma 4. By definition, Solution 1 satisfies the necessary condition stated in Lemma 2. By Lemma 3, $g(L_c,|S_c|)+g(L\setminus S_c,\min\{B,|L|\}-|S_c|) \geq \alpha^k u_j^k$ holds. Putting together the condition $g(L_c,|S_c|+1)+g(L\setminus S_c^+,\min\{B,|L|\}-|S_c|-1) < \alpha^k u_j^k$, we have $g(L_c,|S_c|)+g(L\setminus S_c,\min\{B,|L|\}-|S_c|) \geq \alpha^k u_j^k > g(L_c,|S_c|+1)+g(L\setminus S_c^+,\min\{B,|L|\}-|S_c|-1)$, implying that the weight of the $(|S_c|+1)^{th}$ element in L_c is strictly lower than that of the $(\min\{B,|L|\}-|S_c|)^{th}$ element in $L\setminus S_c^+$. Furthermore, we have $\alpha^k u_j^k > g(L_c,|S_c|+1)+g(L\setminus S_c^+,\min\{B,|L|\}-|S_c|-1) \geq g(L_c,|S_c|+a)+g(L\setminus S_c^{+a},\min\{B,|L|\}-|S_c|-a)$, where S_c^{+a} denotes the set of first $|S_c|+a$ elements in L_c for any $1 \leq a \leq |L_c|-|S_c|$, because L_c and $L\setminus S_c$ are in descending order.

Algorithm 1: Optimal recommendation for given $\mathcal{I}_c$

```
for k ∈ K do
    for j ∈ I do
        L ← Sort the arcs in G_j^k in descending order of weights
        L_c ← The list of arcs in L ending at the cached nodes
        S ← ∅,  u_j^k ← g(L, min{B, |L|})
        Bi-section search to maximize |S_c| for set
          S_c ← L_c[1, 2, ..., |S_c|] satisfying
          g(L_c, |S_c|)+g(L\S_c, min{B, |L|}−|S_c|) ≥ α^k u_j^k
        Add the first |S_c| elements in L_c to S
        Add the first min{B, |L|} − |S_c| elements in
          L \ S_c to S for i ∈ I do
            if (j, i) ∈ S then
                y_ji^k ← 1
            else
                y_ji^k ← 0
return y
```

Assuming Solution 1 is not optimal, then there exists another solution (called Solution 2) with a higher objective value than Solution 1. We denote by S_c and S_c' to distinguish between the selected arcs in Solution 1 and Solution 2. Since both Solutions 1 and 2 satisfy the necessary condition in Lemma 2, to achieve a higher objective value by Solution 2, it must be the case that $|S_c'| = |S_c|+a$, where $a \geq 1$. By the arguments in the proof of Lemma 2, the list-level satisfaction in Solution 2 could not be higher than $g(L_c,|S_c'|)+g(L\setminus S_c',\min\{B,|L|\}-|S_c'|) = g(L_c,|S_c|+a)+g(L\setminus S_c^{+a},\min\{B,|L|\}-|S_c|-a) < \alpha^k u_j^k$. Hence Solution 2 is not feasible, and we conclude the lemma. □

Motivated by Lemma 4, we propose a bi-section search algorithm for the recommendation problem. The principle of the algorithm is to construct a solution containing as many as possible elements from the top of list L_c and meanwhile guaranteeing the list-level constraints being satisfied. Specifically, we deploy bi-section search to maximize $|S_c|$. For each $|S_c|$, the algorithm selects the first $|S_c|$ elements in L_c and checks if the feasibility condition in Lemma 3 is met. Then the first $\min\{B,|L|\}-|S_c|$ elements in $L\setminus S_c$ will be selected to fulfill the list-level constraint. The proposed algorithm is presented in Algorithm 1, where S denotes the set of recommended contents. An example output is shown in Fig. 3b.

Theorem 5. *Algorithm 1 gives the optimum of* (6)*.*

Proof. By construction, the output of Algorithm 1 satisfies the sufficient condition in Lemma 4, and the theorem follows. □

B. The Caching Problem

We consider the second sub-problem to optimize caching with given recommendation $\boldsymbol{y}$. As $\boldsymbol{y}$ is fixed, the optimization

problem becomes

$$\max_{\boldsymbol{x}} \sum_{i\in\mathcal{I}} x_i q_i(\boldsymbol{y}) \tag{7a}$$

$$\textbf{s.t.} \quad \sum_{i\in\mathcal{I}} s_i x_i \le C \tag{7b}$$

$$x_i \in \{0,1\}, \quad i \in \mathcal{I} \tag{7c}$$

Since $q_i(\boldsymbol{y})$ is a given value for content i, the caching problem as defined in (7) is a knapsack problem, where each content i is an item with weight $q_i(\boldsymbol{y})$ and the knapsack capacity is C. To solve the caching problem, we leverage the *modified greedy* algorithm for solving a generic knapsack problem. The algorithm consists of the following three steps. First, the algorithm computes the utility for each content i, defined as $\frac{q_i(\boldsymbol{y}')}{s_i}$, and sorts them in descending order. Second, the algorithm greedily caches contents in the given order until caching the j^{th} content would exceed the cache capacity. Third, the algorithm takes the best of caching the first $j-1$ contents or the j^{th} content. The modified greedy algorithm has an $\frac{1}{2}$-approximation guarantee that can be easily established using an argument similar to that in [20].

C. The Alternating (ALT) Optimization Algorithm

Utilizing the solutions of the recommendation problem and the caching problem, we propose an alternating (ALT) optimization algorithm for CIAR. The basic idea of ALT is to alternatively optimize $\boldsymbol{x}$ and $\boldsymbol{y}$. The first phase of ALT is the initialization of recommendation. For any user k with incumbent content j, we select the most relevant $\min\{B, |\mathcal{D}_j^k|\}$ contents for j under the preference of user k to form the recommendation solution $\boldsymbol{y}'$. By the definition of u_j^k, one can verify that $\boldsymbol{y}'$ satisfies the list-level constraint (5d) with any value of $\alpha \in [0,1]$. Secondly, the algorithm solves the caching problem in (7) with given $\boldsymbol{y}'$, see Sec. V-B. Now the obtained $\boldsymbol{x}$ and $\boldsymbol{y}'$ have formed a feasible solution of CIAR. Third, we solve the recommendation problem in (6) with given $\boldsymbol{x}$ by invoking Algorithm 1. The ALT algorithm then alternates between solving the two sub-problems, and terminates if the solution remains unchanged or after a maximum of R iterations. Clearly, by construction ALT runs in polynomial time in the size of the content set $\mathcal{I}$ for any constant R.

VI. Numerical Results

A. Preliminaries

In our simulation, we use a well-known movie reviews dataset MovieLens [21] to evaluate the proposed algorithm, with different values of cache capacity and recommendation list length. Besides, as a supplement for testing the impact of graph structures on algorithm performance, we also use a synthetic dataset. Table I shows the basic parameters of the two datasets.

1) Real-world Dataset: The value of p_i^k $(\forall i \in \mathcal{I}, k \in \mathcal{K})$ is derived by evaluating how content i matches the preference of user k. Specifically, we consider 18 themes for describing both contents and users. A binary vector $\boldsymbol{v}_i = [v_{i1}, v_{i2}, \ldots, v_{i18}]^{\mathrm{T}}$ is used to represent the feature of content i, where an element is one if content i is related to corresponding theme. Similarly, vector $\boldsymbol{v}^k = [v_1^k, v_2^k, \ldots, v_{18}^k,]^{\mathrm{T}}$ shows the preference of user k for these themes. Each element in $\boldsymbol{v}^k$ is a value within $[0,1]$, and a higher value means a stronger preference. Then level of match between content i and user k is quantified by computing the similarity between the two corresponding vectors [22], i.e.,

$$d_i^k = \frac{1}{1+||\boldsymbol{v}^k - \boldsymbol{v}_i||_2}, \forall i \in \mathcal{I}, k \in \mathcal{K} \tag{8}$$

where $||\cdot||_2$ computes Euclidean distance. By normalization, the value of p_i^k can be computed by

$$p_i^k = \frac{d_i^k}{\sum_{i\in\mathcal{I}} d_i^k}, \forall i \in \mathcal{I}, k \in \mathcal{K} \tag{9}$$

For parameter p_{ji}^k, its values in fact reflect the similarity between contents i and j with respect to attracting user k, referred to as the content correlation with respect to user preference. We compute the value of p_{ji}^k by

$$p_{ji}^k = p_{ij}^k = \frac{1}{1+||\boldsymbol{v}^k \circ (\boldsymbol{v}_j - \boldsymbol{v}_i)||_2}, \forall i,j \in \mathcal{I}, k \in \mathcal{K} \tag{10}$$

where $\circ$ represents element-wise product of two vectors.

2) Synthetic Dataset: The density of a content graph is represented by the number of arcs in relation to the number of nodes. For a content, this corresponds to the proportion of contents that are relevant. The arcs are randomly generated subject to the density parameter, and the values of p_i^k and p_{ji}^k $(\forall i,j \in \mathcal{I}, k \in \mathcal{K})$ are randomly generated obeying the uniform distribution in $[0,1]$. Moreover, the values of p_i^k $(\forall i \in \mathcal{I})$ are normalized for each user k. We consider non-uniform content sizes for the dataset.

Table I: The parameters of the two datasets

	Parameters	Values
Real-world	The number of users $\lvert\mathcal{K}\rvert$	20
	The number of contents $\lvert\mathcal{I}\rvert$	50
	Threshold α^k $(\forall k \in \mathcal{K})$	0.4
	Threshold β^k $(\forall k \in \mathcal{K})$	0.01
	Cache capacity C	$\{1, 1.5, 2, 2.5\}$
	Recommendation list length B	$\{6, 12, 18, 24\}$
	Content size	Uniform (0.1)
Synthetic	The number of users $\lvert\mathcal{K}\rvert$	20
	The number of contents $\lvert\mathcal{I}\rvert$	$\{30, 40, 50, 60, 70\}$
	Threshold α^k $(\forall k \in \mathcal{K})$	0.4
	Threshold β^k $(\forall k \in \mathcal{K})$	0.01
	Cache capacity C	4
	Recommendation list length B	6
	Content size	$[0.1, 0.9]$

B. Benchmarks

Two benchmarks are used for comparison. One benchmark is ILP, for which the optimal solution of (5) is obtained by Gurobi [19]. Another benchmark is a popularity-based algorithm, referred to as POP, where caching and recommendation are optimized separately based on content popularity.

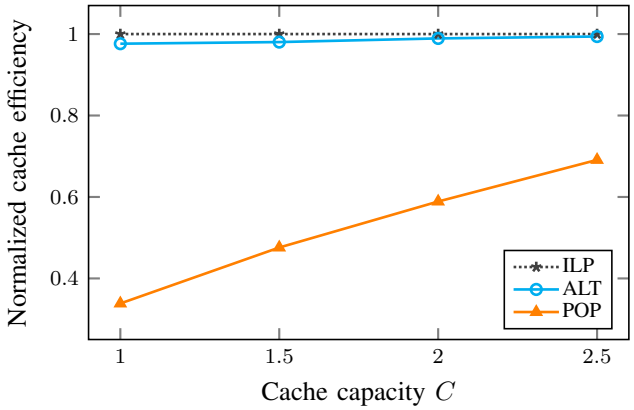


Figure 4: The normalized cache efficiency as function of the cache capacity C on dataset MovieLens when $B = 6$.

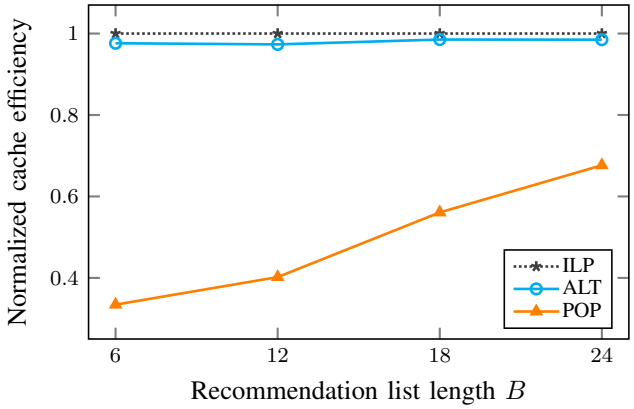


Figure 5: The normalized cache efficiency as function of the recommendation list length B on dataset MovieLens when $C = 4$.

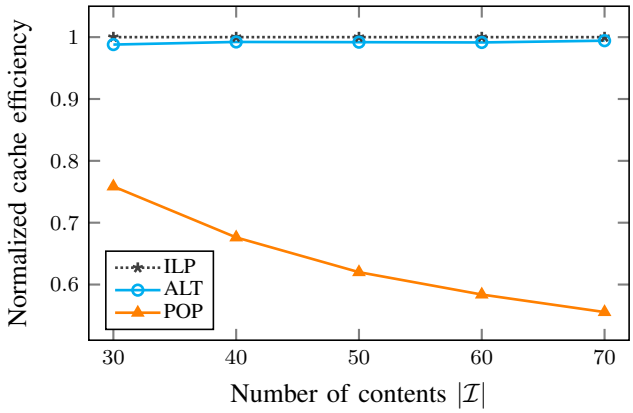


Figure 6: The normalized cache efficiency as function of the number of contents $|\mathcal{I}|$ on the synthetic dataset when density=0.8.

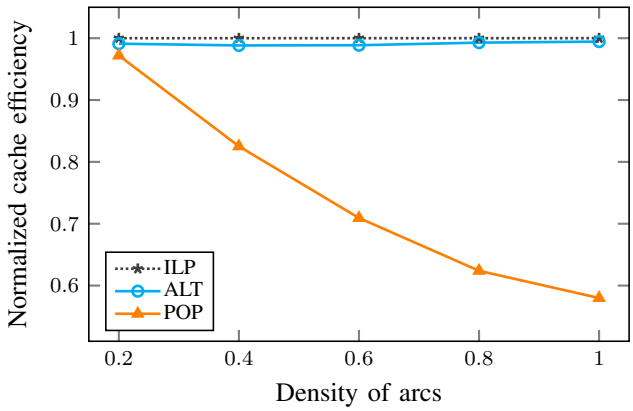


Figure 7: The normalized cache efficiency as function of the density of arcs on the synthetic dataset when $|\mathcal{I}| = 50$.

For recommendation optimization, the recommendation list is created by sorting contents in descending order of p^k_{ji} and including the first $\min\{B, |\mathcal{D}^k_j|\}$ elements into the list, for each user k and content j. For caching optimization, the contents are sorted in descending order of the popularity-to-size ratio (i.e., $\sum_{k\in\mathcal{K}} p^k_i/s_i$), and contents are greedily added into the cache until the capacity becomes exceeded.

C. Results and Discussion

The numerical results are shown in Figs. 5-7. For each combination of the values of parameters, we use 20 instances and present the average result. In addition, the results are normalized by the global optimum. Hence the numerical performance of ILP is always one. Empirically, algorithm ALT converges after the initialization and one iteration of solving the two sub-problems, for all tested instances.

Overall, ALT can provide very good solutions with at most 2.4% gap to the optimum for both the real-world and synthetic datasets. Fig. 4 and Fig. 5 show the stable performance of ALT with varying recommendation list length and cache capacity. Besides, ALT apparently outperforms POP, even though the gap between them gets smaller with larger cache capacity and longer recommendation lists. Note that two critical differences between ALT and POP are whether or not to consider: (1) content popularity affected by recommendation, when optimizing caching, and (2) reshaping the popularity distribution to utilize the cache efficiently, when optimizing recommendation. The optimal recommendation will tend to largely improve the popularity of cached contents. For ALT, compared with POP, the advantage brought by (1) will be more obvious for smaller B, while the advantage brought by (2) is more prominent for smaller C. These effects can be observed in the figures. Note that, in theory, all the three algorithms will achieve the same value, if the cache capacity is infinity.

Fig. 6 and Fig. 7 show that the performance of ALT remains close-to-optimal, when varying the number of contents and density. Algorithm ALT clearly outperforms POP, in particular with more contents and higher density. For ALT, compared with POP, the advantage brought by (2) above is more prominent with more contents. We notice that for a very low density of 0.2, the performance of ALT and POP are close. That

is because in this case, for a content, there are few related contents, and the number of feasible recommendation solutions is very limited. In other words, which algorithm to use does not matter much, since any feasible solution is close to optimum.

VII. Conclusions

In this paper, we jointly optimize caching and incumbent-aware recommendation for improving the caching efficiency, considering the content-level and list-level user satisfaction constraints. This problem is proved to be NP-hard. We approach it via a graph perspective and derive an alternating algorithm where the recommendation sub-problem is solved to optimum. Simulation results show the close-to-optimal performance of the proposed algorithm, hence making content recommendation both personalized and incumbent aware is a promising scheme.

Acknowledgment

The second and third authors are supported by the Swedish Research Council, under Project 2018-05247.

References

[1] G. S. Paschos, G. Iosifidis, M. Tao, D. Towsley, and G. Caire, "The role of caching in future communication systems and networks," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 6, pp. 1111–1125, 2018.

[2] L. Sun, X. Wang, Z. Wang, H. Zhao, and W. Zhu, "Social-aware video recommendation for online social groups," *IEEE Transactions on Multimedia*, vol. 19, no. 3, pp. 609–618, 2017.

[3] C. A. Gomez-Uribe and N. Hunt, "The netflix recommender system: Algorithms, business value, and innovation," *ACM Transactions on Management Information Systems*, vol. 6, no. 4, pp. 1–19, 2016.

[4] J. Sung, M. Kim, K. Lim, and J.-K. K. Rhee, "Efficient cache placement strategy in two-tier wireless content delivery network," *IEEE Transactions on Multimedia*, vol. 18, no. 6, pp. 1163–1174, 2016.

[5] T. V. Doan, V. Bajpai, and S. Crawford, "A longitudinal view of Netflix: Content delivery over IPv6 and content cache deployments," in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, 2020, pp. 1073–1082.

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

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

[8] ——, "Jointly optimizing content caching and recommendations in small cell networks," *IEEE Transactions on Mobile Computing*, vol. 18, no. 1, pp. 125–138, 2019.

[9] D. Zheng, Y. Chen, M. Yin, and B. Jiao, "Cooperative cache-aware recommendation system for multiple internet content providers," *IEEE Wireless Communications Letters*, vol. 9, no. 12, pp. 2112–2115, 2020.

[10] L. E. Chatzieleftheriou, G. Darzanos, M. Karaliopoulos, and I. Koutsopoulos, "Joint user association, content caching and recommendations in wireless edge networks," *ACM SIGMETRICS Performance Evaluation Review*, vol. 46, no. 3, pp. 12–17, 2019.

[11] Y. Fu, L. Salaün, X. Yang, W. Wen, and T. Q. S. Quek, "Caching efficiency maximization for device-to-device communication networks: A recommend to cache approach," *IEEE Transactions on Wireless Communications*, pp. 1–1, 2021.

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

[13] Y. Fu, Z. Yang, T. Q. S. Quek, and H. H. Yang, "Towards cost minimization for wireless caching networks with recommendation and uncharted users' feature information," *IEEE Transactions on Wireless Communications*, vol. 20, no. 10, pp. 6758–6771, 2021.

[14] S. Krishnendu, B. N. Bharath, and V. Bhatia, "Joint edge content cache placement and recommendation: Bayesian approach," in *2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring)*, 2021, pp. 1–5.

[15] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 6, pp. 734–749, 2005.

[16] 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.

[17] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[18] "IBM ILOG CPLEX Optimizer, version 20.1." [Online]. Available: https://www.ibm.com/se-en/analytics/cplex-optimizer.

[19] "Gurobi Optimizer, version 9.5." [Online]. Available: https://www.gurobi.com/products/gurobi-optimizer/

[20] D. P. H. Kellerer, U. Pferschy, *Knapsack Problems*. Springer-Verlag Berlin Heidelberg, 2004.

[21] "MovieLens dataset." [Online]. Available: https://grouplens.org/datasets/movielens/100k/.

[22] H.-S. Park and C.-H. Jun, "A simple and fast algorithm for k-medoids clustering," *Expert Systems with Applications*, vol. 36, no. 2, Part 2, pp. 3336–3341, 2009.