

Accounting for Information Freshness in Scheduling of Content Caching

Ghafour Ahani and Di Yuan

Department of Information Technology, Uppsala University, Sweden
Emails: {ghafour.ahani, di.yuan}@it.uu.se

Abstract—In this paper, we study the problem of optimal scheduling of content placement along time in a base station with limited cache capacity, taking into account jointly the offloading effect and freshness of information. We model offloading based on popularity in terms of the number of requests and information freshness based on the notion of age of information (AoI). The objective is to reduce the load of backhaul links as well as the AoI of contents in the cache via a joint cost function. For the resulting optimization problem, we prove its hardness via a reduction from the Partition problem. Next, via a mathematical reformulation, we derive a solution approach based on column generation and a tailored rounding mechanism. Finally, we provide performance evaluation results showing that our algorithm provides near-optimal solutions.

Index Terms—Age of information, base station, caching, time-varying popularity.

I. INTRODUCTION

Content caching at the network edge is considered to be an enabler for future wireless networks. This technique strives to mitigate the heavy burden on backhaul links via providing the users with their contents of interest from the network edge without the need of going to the core networks.

In designing effective caching strategies, previous works have focused on content popularity, whereas another important aspect is information freshness. Popularity of a content is defined as the number of users requesting the content. Popularity may vary over time [1]. Thus, some contents may be added to or removed from the cache as they become popular or unpopular. Freshness of contents in the cache refers to how recent the content has been obtained from the core network. The longer a content is stored in the cache without an update, the higher risk is that the cached content becomes obsolete. Hence, we would like to refresh the cached contents often, which however leads to higher load on the backhaul. Freshness of contents naturally arises in applications such as news, traffic information, etc., and it may have a great impact on user satisfaction. We model freshness of contents using the notion of age of information (AoI). For content caching, AoI is defined as the amount of time elapsed since the time that the content is refreshed. In this paper, we use a joint cost function to address the trade-off between the benefit of offloading via caching and AoI.

The works such as [2]–[5] took into account only the popularities of contents in designing cache placement strategies. The works in [2], [3] considered content caching with known popularities of contents. The studies in [4], [5] showed that the popularities of contents can be estimated via learning-based

algorithms. However, in the mentioned works popularity of a content are time-invariant. In [6], [7], caching with time-varying popularity profiles are investigated. In [7] an algorithm is proposed to estimate the time-varying popularities of contents. The studies in [8], [9] considered information freshness but not popularity of contents in their caching problems. Recently, a few works [10]–[12] have considered both popularity and freshness of contents. However, these works have the following limitations. In [10], the downloading cost of contents from the server is neglected. In [11], only one content of the cache could be updated in each time slot. In [12], it is assumed that the cache capacity is unlimited.

In this paper, we study optimal scheduling of content caching along time in a base station (BS) with limited storage capacity taking into account jointly offloading via caching and freshness of contents. The objective is to mitigate the load of backhaul links via minimizing a penalty cost function related to content downloading, content updating, and AoI costs subject to the cache capacity. The main contributions of this work are summarized as follows:

- The caching scheduling problem is formulated as an optimization problem. Specifically, it is formulated as an integer linear program (ILP) and the hardness of the problem is proved based on a reduction from the Partition problem.
- Via a problem reformulation, a column generation algorithm (CGA) is developed. We prove that the subproblem of CGA can be converted to a shortest path problem that can be solved in polynomial time. In addition, the CGA provides an effective lower bound (LB) of global optimum.
- The solution obtained from CGA could be fractional, thus an advanced and problem-tailored rounding algorithm (RA) is derived to construct integer solutions.
- Simulations show the effectiveness of our solution approach by comparing the obtained solutions to the LB as well as the conventional algorithms. Our algorithm provides solutions within 1% of global optimum.

II. SYSTEM SCENARIO AND PROBLEM FORMULATION

A. System Scenario

The system scenario consists of a content server, a BS and a set of users $\mathcal{U} = \{1, 2, \dots, U\}$ within the coverage of the BS. The server has all the contents, and the BS is equipped with a cache device of capacity S . The contents are dynamic, i.e., the information they contain may change over time. Denote

by $\mathcal{F} = \{1, 2, \dots, F\}$ the set of the contents. We assume the server has always the up-to-date version of the contents. Denote by l_f the size of content f . Each content is either fully stored or not stored at all at the BS. The system scenario is shown in Figure 1.

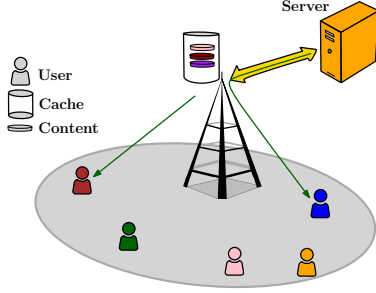


Figure 1. System scenario.

We consider a slotted time system of $\mathcal{T} = \{1, 2, \dots, T\}$ time slots. At the beginning of each time slot, the contents to be stored in the cache need to be determined by an updating/placement action. Namely, some stored contents may be removed from the cache, some contents may be added to the cache, and some contents may be re-downloaded from the server. The freshness of a content may decrease along time. We use AoI to model the freshness of contents. A content that is newly downloaded from the cache has AoI 0, and for each time slot it remains in the cache without re-downloading, its AoI increases with one time slot. Denote by $p_f(i)$ the cost associated with an AoI of i time slots for content f . A content has AoI i time slots when the content has been stored in the cache for i continuously time slots without any update.

In our model, user $u \in \mathcal{U}$ requests at most R_u contents within the T time slots based on its interest. The set of requests for user u is denoted by $\mathcal{R}_u = \{1, \dots, R_u\}$. The downloading process of a content starts as soon as the request is made. The content can be downloaded either from the cache if the content is in the cache, or otherwise from the server. We assume the time of each request is known or can be predicted via using a prediction model, e.g., the one in [7]. For user u and its r -th request, the requested content and the time slot of request are denoted by $h(u, r)$ and $o(u, r)$, respectively.

B. Cost Model

Denote by x_{tf} a binary optimization variable which equals one if and only if the f -th content is stored in time slot t . Denote by c_s and c_b the costs for downloading one unit of data from the server and the cache to a user, respectively. We have $c_s > c_b$ to encourage downloading from the cache. The downloading cost for user u to obtain its r -th request, denoted by C_{ur} , is expressed as:

$$C_{ur} = l_{h(u,r)}[c_b x_{o(u,r)h(u,r)} + c_s(1 - x_{o(u,r)h(u,r)}). \quad (1)$$

The downloading cost for completing all requests of all users, denoted by $C_{download}$, is $C_{download} = \sum_{u=1}^U \sum_{r=1}^{R_u} C_{ur}$.

Denote by binary variable a_{tfi} , $i \in \{0, 1, \dots, t-1\}$, whether or not content f is in the cache and has AoI i time slots. The overall AoI cost is expressed as:

$$C_{AoI} = \sum_{f=1}^F \sum_{t=1}^T \sum_{i=1}^{t-1} p_f(i) m_{tf} a_{tfi}, \quad (2)$$

where m_{tf} is the number of users requesting content f in time slot t . Updating contents in the cache incurs an updating cost. The updating cost, denoted by C_{update} , is expressed as:

$$C_{update} = \sum_{t=1}^T \sum_{f=1}^F l_f (c_s - c_b) a_{tf0}, \quad (3)$$

where a_{tf0} means that the content is just downloaded from the server and has cost $l_f(c_s - c_b)$. Here $(c_s - c_b)$ is the downloading cost unit from the server to the cache. Finally, the total cost is denoted by C_{total} and expressed as:

$$C_{total} = C_{download} + C_{update} + \lambda C_{AoI}. \quad (4)$$

Here, λ is a weighting factor between C_{AoI} and C_{update} . Larger λ means frequently updating the contents of the cache and consequently smaller AoI for cached contents.

C. Problem Formulation

The update-enabled caching problem (UECP) is formulated as an ILP, and shown in (5). Constraints (5b) indicate that

$$(UECP) \quad \min_{\mathbf{x}, \mathbf{a}} \quad C_{download} + C_{update} + \lambda C_{AoI} \quad (5a)$$

$$\text{s.t.} \quad \sum_{f=1}^F x_{tf} l_f \leq S, t \in \mathcal{T}, \quad (5b)$$

$$\sum_{i=0}^{t-1} a_{tfi} = x_{tf}, t \in \mathcal{T}, f \in \mathcal{F}, \quad (5c)$$

$$a_{tfi} \geq x_{tf} + a_{(t-1)f(i-1)} - a_{tf0} - 1, \\ t \in \mathcal{T} \setminus \{1\}, f \in \mathcal{F}, i \in \{1, \dots, t-1\}, \quad (5d)$$

$$x_{tf}, a_{tfi} \in \{0, 1\}, t \in \mathcal{T}, f \in \mathcal{F}, i \in \{0, \dots, t-1\}. \quad (5e)$$

used storage space is less than or equal to the cache capacity in each time slot. Constraints (5c) state that if the content is in the cache, it has to have one of the AoIs $0, \dots, t-1$. Constraints (5d) indicate content f in time slot t has AoI i if and only if the content is in the cache in time slot t , has not AoI 0 in time slot t , and has AoI $i-1$ in time slot $t-1$.

Even though this ILP can be solved by a standard solver, it needs significant computational time. Exploiting the structure of the problem, we develop an solution method based on column generation.

D. Complexity Analysis

Theorem 1. *UECP is \mathcal{NP} -hard.*

Proof. The proof is established by a polynomial reduction from the Partition problem that is \mathcal{NP} -complete [13]. Consider a Partition problem with a set of N integers, i.e., $\mathcal{N} = \{n_1, \dots, n_N\}$. The task is to decide whether it is possible to partition \mathcal{N} into two subsets \mathcal{N}_1 and \mathcal{N}_2 with equal sum.

The reduction is constructed as follows. We set the cache capacity as $S = \frac{1}{2} \sum_{i=1}^N n_i$, the set of contents to $\mathcal{F} = \{1, \dots, N\}$, size of content $f \in \mathcal{F}$ to $l_f = n_f$, and the number of time slots to one, i.e., $T = 1$. As $T = 1$, there is no updating or AoI costs. The time slots of all requests are set to 1, i.e., $o(u, r) = 1, u \in \mathcal{U}, r \in \mathcal{R}_u$. We set $m_{1f} = 2$ for $f \in \mathcal{F}$, $c_s = 2$, and $c_b = 1$. By this setting, if the cache stores content f , $4l_f - l_f - 2l_f = l_f$ gain is achieved. As the cache capacity is $S = \frac{1}{2} \sum_{i=1}^N n_i$, a maximum possible of $\frac{1}{2} \sum_{i=1}^N n_i$ gain can be achieved. Now, the question is if this maximum gain can be achieved. This question can be answered by solving UECP which also will answer the Partition problem. Hence the conclusion. \square

III. REFORMULATION OF UECP

We provide a reformulation of the problem that enables a CGA. We define the caching and updating decisions for content f across the T time slots as tuple $(\mathbf{x}_f, \mathbf{a}_f)$ in which $\mathbf{x}_f = [x_{1f}, x_{2f}, \dots, x_{Tf}]^T$ and $\mathbf{a}_f = [a_{1f0}, a_{2f0}, \dots, a_{Tf(t-1)}]^T$. In total, 3^T of such tuples exist and one of them is used in a solution. Denote by $\mathcal{K} = \{1, 2, \dots, 3^T\}$ the index set for all possible solutions. We refer to a possible solution as a column. The cost of column $k \in \mathcal{K}$ for content $f \in \mathcal{F}$ is denoted by C_{fk} and can be calculated by the formula in (6).

$$C_{fk} = \sum_{t=1}^T l_f m_{tf} [c_b x_{tf}^{(k)} + c_s (1 - x_{tf}^{(k)})] + \sum_{t=1}^T l_f (c_s - c_b) a_{tf0}^{(k)} + \lambda \sum_{t=1}^T \sum_{i=1}^{t-1} p_f(i) m_{tf} a_{tfi}^{(k)}. \quad (6)$$

In (6), $x_{tf}^{(k)}$ and $a_{tfi}^{(k)}$ are constants and represent the values of x_{tf} and a_{tfi} with respect to k -th column, respectively. Now, ILP (5) can be reformulated as (7).

$$\min_{\mathbf{w}} \sum_{f \in \mathcal{F}} \sum_{k \in \mathcal{K}} C_{fk} w_{fk} \quad (7a)$$

$$\text{s.t.} \quad \sum_{f \in \mathcal{F}} \sum_{k \in \mathcal{K}} l_f x_{tf}^{(k)} w_{fk} \leq S, t \in \mathcal{T} \quad (7b)$$

$$\sum_{k \in \mathcal{K}} w_{fk} = 1, f \in \mathcal{F} \quad (7c)$$

$$w_{fk} \in \{0, 1\}, f \in \mathcal{F}, k \in \mathcal{K}. \quad (7d)$$

Here, w_{fk} is a binary variable where $w_{fk} = 1$ if and only if the k -th column of content f is selected, otherwise it is zero. Constraints (7b) are the cache capacity constraints, and constraints (7d) indicate that only one of the columns is used.

IV. ALGORITHM DESIGN

In this section, we present our solution method which consists of two algorithms. Algorithm 1 is a column generation algorithm (CGA) applied to the continuous version of (7). Algorithm 2 is a rounding algorithm (RA) applied to the solution obtained from CGA if the solution is fractional. These algorithms are applied alternately until an integer solution is constructed. The solution method is shown in Algorithm 1. The term RMP in the algorithm will be discussed later.

Algorithm 1: CGA and RA

```

1: STOP  $\leftarrow 0$ 
2: while (STOP = 0) do
3:   Apply CGA to RMP and obtain  $\mathbf{w}^*$ 
4:   if ( $\mathbf{w}^*$  is an integer solution) then
5:     STOP  $\leftarrow 1$ 
6:   else
7:     Apply RA to  $\mathbf{w}^*$ 

```

A. Column Generation Algorithm

In column generation, the problem is decomposed into a so called master problem (MP) and a subproblem (SP). The algorithm starts with a subset of columns and solves alternately MP and SP. Each time SP is solved a new column that possibly improves the objective function is generated. The benefit of CGA is to exploit the fact that at optimum only a few columns are used.

1) *MP and RMP*: MP is the continuous version of formulation (7). Restricted MP (RMP) is the MP but with a small subset $\mathcal{K}'_f \subset \mathcal{K}$ for any content $f \in \mathcal{F}$. RMP is expressed in (8). Denote by K'_f the cardinality of \mathcal{K}'_f .

$$(\text{RMP}) \quad \min_{\mathbf{w}} \sum_{f \in \mathcal{F}} \sum_{k \in \mathcal{K}'_f} C_{fk} w_{fk} \quad (8a)$$

$$\text{s.t.} \quad \sum_{f \in \mathcal{F}} \sum_{k \in \mathcal{K}'_f} l_f x_{tf}^{(k)} w_{fk} \leq S, t \in \mathcal{T}, \quad (8b)$$

$$\sum_{k \in \mathcal{K}'_f} w_{fk} = 1, f \in \mathcal{F}, \quad (8c)$$

$$0 \leq w_{fk} \leq 1, f \in \mathcal{F}, k \in \mathcal{K}'_f. \quad (8d)$$

2) *Subproblem*: The SP uses the dual information to generate new columns. Denote by $\mathbf{w}^* = \{w_{fk}^*, f \in \mathcal{F} \text{ and } k \in \mathcal{K}'_f\}$ the optimal solution of RMP. Denote by $\boldsymbol{\pi}^*$ and $\boldsymbol{\beta}^*$ the corresponding optimal dual variables of (8b) and (8c), respectively, i.e., $\boldsymbol{\pi}^* = [\pi_1^*, \pi_2^*, \dots, \pi_T^*]^T$ and $\boldsymbol{\beta}^* = [\beta_1^*, \beta_2^*, \dots, \beta_F^*]^T$. After obtaining \mathbf{w}^* , we need to check whether \mathbf{w}^* is the optimal solution of RMP. This can be determined by finding a column with the minimum reduced cost for each content $f \in \mathcal{F}$. If all these values are nonnegative, the current solution is optimal. Otherwise, we add the columns with negative reduced cost to corresponding sets.

Given $\boldsymbol{\pi}^*$ and $\boldsymbol{\beta}^*$ for content $f \in \mathcal{F}$, the reduced cost of column $(\mathbf{x}_f, \mathbf{a}_f)$ is $C_f - \sum_{t=1}^T l_f \pi_t^* x_{tf} - \beta_f^*$ where C_f can be computed using expression (6) in which constants $x_{tf}^{(k)}$ and $a_{tfi}^{(k)}$ are replaced with optimization variables x_{tf} and a_{tfi} , respectively. To find the column with minimum reduced cost for content $f \in \mathcal{F}$, we need to solve subproblem SP_f , shown in (9). Denote by $(\mathbf{x}_f^*, \mathbf{a}_f^*)$ the optimal solution of SP_f . If the reduced cost of $(\mathbf{x}_f^*, \mathbf{a}_f^*)$ is negative, we add it to \mathcal{K}'_f .

Even though (9) is an ILP, in the following, we show that it can be solved as a shortest path problem using for example Dijkstra's algorithm [14] in polynomial time.

Theorem 2. For content $f \in \mathcal{F}$, SP_f can be solved in polynomial time as a shortest path problem.

$$\begin{aligned}
(\text{SP}_f) \quad & \min_{(\mathbf{x}_f, \mathbf{a}_f)} C_f - \sum_{t=1}^T l_f \pi_t^* x_{tf} - \beta_f^* \quad (9a) \\
\text{s.t.} \quad & \sum_{i=0}^{t-1} a_{tfi} = x_{tf}, t \in \mathcal{T}, f \in \mathcal{F}, \quad (9b) \\
& a_{tfi} \geq x_{tf} + a_{(t-1)f(i-1)} - a_{tf0} - 1, \\
& t \in \mathcal{T} \setminus \{1\}, f \in \mathcal{F}, i \in \{1, \dots, t-1\}, \quad (9c) \\
& a_{tfi} \leq a_{(t-1)f(i-1)}, t \in \mathcal{T} \setminus \{1\}, f \in \mathcal{F}, \\
& i \in \{1, \dots, t-1\}, \quad (9d) \\
& x_{tf} \in \{0, 1\}, t \in \mathcal{T}, f \in \mathcal{F}, \quad (9e) \\
& a_{tfi} \in \{0, 1\}, t \in \mathcal{T}, f \in \mathcal{F}, i \in \{0, \dots, t-1\}. \quad (9f)
\end{aligned}$$

Proof. Consider content $f \in \mathcal{F}$. We construct an acyclic directed graph where finding the shortest path from the source to destination is equivalent to solving SP_f . The objective function (9a) can be rewritten as (10). Denote by C the total cost for downloading content f via the server for all users requesting the content over all time slots, i.e., $C = \sum_{t=1}^T l_f m_{tf} c_s$. Denote by $v_{it} = p_f(i) m_{tf}$ the scenario where m_{tf} users request content f in time slot t and the content has AoI i . Denote by $c_1 = l_f(c_s - c_b)$ the downloading cost from the server to the cache. Denote by $g_t = l_f m_{tf}(c_s - c_b) - l_f \pi_t^*$ the reduction in C due to storing content f .

The graph is constricted as follows. Nodes S and D are used to represent the source and destination. Node V_{00} is used to represent $x_0 = 0$. For time slot t , there are $t+1$ vertically aligned nodes. Using node V_{t0} means that the content is not in the cache, and using node V_{t1}^i , $i \in \{0, \dots, t-1\}$, means that the content is in the cache and has AoI i . From node S to V_{00} there is an arc with weight C . For each node V_{t0} there are two outgoing arcs one to $V_{(t+1)0}$ which means that the content is not stored in the next time slot and has weight 0, and the other to $V_{(t+1)1}^0$ which has weight $c_1 - g_t$ and means that the content is downloaded to the cache in the next time slot and has AoI 0. For each node V_{t1}^i there three outgoing arcs to $V_{(t+1)0}$, $V_{(t+1)1}^0$, and $V_{(t+1)1}^{i+1}$, respectively. Using the first arc means that the content is deleted for the next time slot and has weight 0. Using the second arc means the content is re-downloaded from the cache and has AoI 0 with weight $c_1 - g_t$. Using the third arc means that the content is kept and its AoI increases with one unit and has weight $v_{(i+1)(t+1)} - g_{(t+1)}$. Finally, there are T arcs from V_{T0} and V_{T1}^i for $i \in \{0, \dots, T-1\}$ to D each with weight $-\beta_f$.

Given any solution of (9), by construction of the graph, the solution directly maps to a path from the source to the destination with the same objective function. Conversely, given a path we construct an ILP solution. For time slot t , if flow is in node V_{t0} then we set $x_{tf} = 0$. If the flow is in V_{t1}^i , we set $x_{t1} = 1$ and $a_{tfi} = 1$. The resulting ILP solution has the same objective function value as length of the given path in terms of the arcs's weights. Hence the conclusion. \square

Algorithm 2: Column Generation Algorithm (CGA)

Input: $S, c_b, c_s, \lambda, l_f, f \in \mathcal{F}, o(u, r)$ and $h(u, r), u \in \mathcal{U}, r \in \mathcal{R}_u$
Output: \mathbf{w}^*
1: $\mathcal{K}'_f \leftarrow \{(\mathbf{0}^T, \mathbf{0}^T)\}$ for $f \in \mathcal{F}$
2: $\text{STOP} \leftarrow 0$
3: **while** ($\text{STOP} = 0$) **do**
4: Solve RMP and obtain $\mathbf{w}^*, \pi^*,$ and β^*
5: $\text{STOP} \leftarrow 1$
6: **for** $f = 1$ to F **do**
7: Solve SP_f and obtain $(\mathbf{x}_f^*, \mathbf{a}_f^*)$
8: **if** $C_{fk^*} - \sum_{t=1}^T l_f \pi_t^* x_{tf}^* - \beta_f^* < 0$ **then**
9: $\mathcal{K}'_f \leftarrow \mathcal{K}'_f \cup \{(\mathbf{x}_f^*, \mathbf{a}_f^*)\}$
10: $\text{STOP} \leftarrow 0$

B. Rounding Algorithm

The solution of CGA could be fractional. Thus, we need a mechanism to construct integer solutions. We design a rounding algorithm (RA) to achieve this. RA repeatedly fixes the caching decisions of contents over time slots until an integer solution is constructed. The caching decision for content f and time slot t is determined based on value z_{tf} , defined as $z_{tf} = \sum_{k \in \mathcal{K}'_f} x_{tf}^{(k)} w_{fk}^*$. This value indicates how likely it is optimal to store content f in time slot t . In the following we prove a relationship between \mathbf{z} and \mathbf{w} and then give the RA.

Theorem 3. For any content $f \in \mathcal{F}$, w_{fk}^* is binary for any k if and only if every element of $\mathbf{z}_f = [z_{1f}, z_{2f}, \dots, z_{Tf}]$ is binary.

Proof. For necessity, for any content $f \in \mathcal{F}$, if w_{fk}^* is binary for any k , $k \in \mathcal{K}'_f$, it is obvious from the definition that all elements of \mathbf{z}_f are binary. Now, we prove the sufficiency. For any content $f \in \mathcal{F}$, assume that every element in \mathbf{z}_f is binary. Assume that $w_{fk}^* > 0$ for $k \in \mathcal{K}''_f \subseteq \mathcal{K}'_f$, then $z_{tf} = \sum_{k \in \mathcal{K}''_f} x_{tf}^{(k)} w_{fk}^*$. To satisfy that element z_{tf} for $t \in \mathcal{T}$ is binary, elements $x_{tf}^{(k)}$ for $k \in \mathcal{K}''_f$ either are all zero or all one. Otherwise, as $\sum_{k \in \mathcal{K}''_f} w_{fk}^* = 1$, one of the z_{tf} becomes fractional. This means that all columns corresponding to w_{fk}^* for $k \in \mathcal{K}''_f$ must be the same. Having two vectors with the same values violates the condition that the columns of any two w_{fk}^* are different. Therefore, for any content f , $f \in \mathcal{F}$, if z_{tf} is binary for any t , $t \in \mathcal{T}$, then w_{fk}^* is an binary for any k , $k \in \mathcal{K}'_f$. Hence the proof. \square

RA consists of three main steps which are shown in Algorithm 3. First, for content $f \in \mathcal{F}$ in time slot $t \in \mathcal{T}$, the decision is to store the content if $z_{tf} = 1$. All columns that do not comply with this caching decision will be discarded. These are done by Lines 2-3. Second, the element of \mathbf{z} being closest to zero or one is found and rounded. Based on the rounding outcome, the caching decision is determined and non-complying columns are discarded. These are done via Lines 4-16. Finally, the algorithm fixes the decisions of the contents across the time slots to zero if there is no remained spare space in the cache to store them in these time slots. This is done by Lines 20-23. The caching decisions made until now will be remained fixed in all subsequent iterations. Note that

$$\begin{aligned}
C_f - \sum_{t=1}^T l \pi_t^* x_{tf} &= \sum_{t=1}^T l_f m_{tf} [c_b x_{tf} + c_s (1 - x_{tf})] + \sum_{t=1}^T l_f (c_s - c_b) a_{tf0} + \sum_{t=1}^T \sum_{i=1}^{t-1} p_f(i) m_{tf} a_{tfi} - \sum_{t=1}^T l_f \pi_{tf}^* x_{tf} \\
&= \underbrace{\sum_{t=1}^T l_f m_{tf} c_s}_C + \sum_{t=1}^T \left[\underbrace{l_f (c_s - c_b)}_{c_1} a_{tf0} + \sum_{i=1}^{t-1} \underbrace{p_f(i) m_{tf}}_{v_{it}} a_{tfi} - \underbrace{[l_f m_{tf} (c_s - c_b) - l_f \pi_{tf}^*]}_{g_t} x_{tf} \right]. \tag{10}
\end{aligned}$$

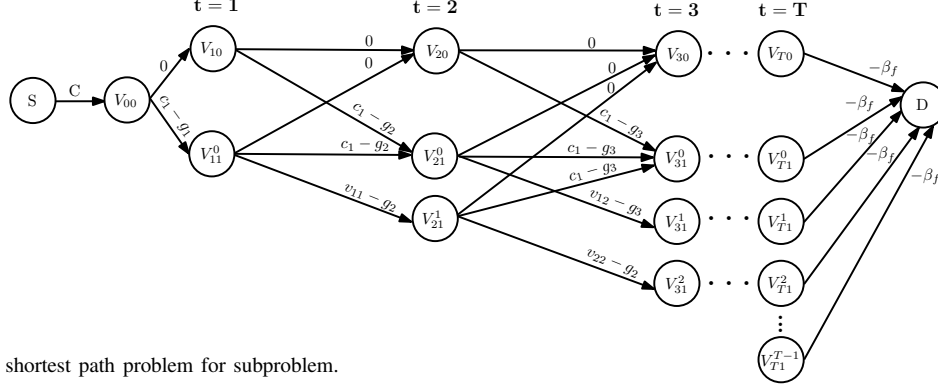


Figure 2. Graph of the shortest path problem for subproblem.

with these fixings SP_f can still be solved as a shortest problem. If x_{tf} is set to 0, nodes V_{t1}^i for $i \in \{0, \dots, t-1\}$ and their connected arcs will be deleted from the graph. If x_{tf} is set to 1, node V_{t0} and its connected arcs will be deleted.

Algorithm 3: Rounding Algorithm (RA)

Input: w^* and (x, a)

- 1: Compute $z = \{z_{tf}, t \in \mathcal{T}, f \in \mathcal{F}\}$ where
 $z_{tf} = \sum_{k \in \mathcal{K}_f'} x_{tf}^{(k)} w_{fk}^*$
 - 2: Fix $x_{tf} = 1$ in SP_f if $z_{tf} = 1, t \in \mathcal{T}, f \in \mathcal{F}$
 - 3: Fix $w_{fk} = 0$ in RMP if $x_{tf}^{(k)} = 0, k \in \mathcal{K}_f', t \in \mathcal{T}, f \in \mathcal{F}$
 - 4: $\bar{z} \leftarrow \min_{t \in \mathcal{T}, f \in \mathcal{F}} \{z_{tf} | z_{tf} > 0 \text{ and } z_{tf} < 1\}$
 - 5: $(\bar{t}, \bar{f}) \leftarrow \arg \min_{t \in \mathcal{T}, f \in \mathcal{F}} \{z_{tf} | z_{tf} > 0 \text{ and } z_{tf} < 1\}$
 - 6: $\bar{z} \leftarrow \min_{t \in \mathcal{T}, f \in \mathcal{F}} \{1 - z_{tf} | z_{tf} > 0 \text{ and } z_{tf} < 1\}$
 - 7: $(\bar{t}, \bar{f}) \leftarrow \arg \min_{t \in \mathcal{T}, f \in \mathcal{F}} \{1 - z_{tf} | z_{tf} > 0 \text{ and } z_{tf} < 1\}$
 - 8: **if** $(\bar{z} < \bar{z})$ **then**
 - 9: Fix $x_{\bar{t}\bar{f}} = 0$ in $\text{SP}_{\bar{f}}$
 - 10: Fix $w_{\bar{f}k} = 0$ if $x_{\bar{t}\bar{f}}^{(k)} = 1, k \in \mathcal{K}_{\bar{f}}'$
 - 11: **else if** $(l_{\bar{f}} \leq S')$ **then**
 - 12: Fix $x_{\bar{t}\bar{f}} = 1$ in $\text{SP}_{\bar{f}}$
 - 13: Fix $w_{\bar{f}k} = 0$ if $x_{\bar{t}\bar{f}}^{(k)} = 0, k \in \mathcal{K}_{\bar{f}}'$
 - 14: **else**
 - 15: Fix $x_{\bar{t}\bar{f}} = 0$ in $\text{SP}_{\bar{f}}$
 - 16: Fix $w_{\bar{f}k} = 0$ if $x_{\bar{t}\bar{f}}^{(k)} = 1, k \in \mathcal{K}_{\bar{f}}'$
 - 17: **for** $t = 1$ **to** T **do**
 - 18: $\mathcal{F}' \leftarrow \{f \in \mathcal{F} | x_{tf} \text{ is set to one}\}$
 - 19: $S' \leftarrow S - \sum_{f \in \mathcal{F}'} l_f$
 - 20: **for** $f \in \mathcal{F} \setminus \mathcal{F}'$ **do**
 - 21: **if** $l_f > S'$ **then**
 - 22: Fix $x_{tf} = 0$ in SP_f
 - 23: Fix $w_{fk} = 0$ in RMP if $x_{tf}^{(k)} = 1, k \in \mathcal{K}_f'$
-

V. PERFORMANCE EVALUATION

We compare CGA to the LB and two conventional caching algorithms: random-based algorithm (RBA) [15] and popularity-based algorithm (PBA) [16]. Both algorithms treat contents one by one. In RBA, the contents are considered randomly, but with respect to their total numbers of requests; a content with higher number of requests will be more likely selected for caching. In PBA, popular contents, i.e., contents with higher number of requests, will be considered first. For the content under consideration, if the content was not in the cache in the previous time slot, it is downloaded with AoI zero. Otherwise, if AoI cost has reached fifty percent of downloading cost, the content is re-downloaded. Otherwise, the content is kept and the AoI increases by one.

The content popularity distribution is modeled by a ZipF distribution [17], i.e., the probability that a user requests the f -th content is $\frac{f^{-\gamma}}{\sum_{i \in \mathcal{F}} i^{-\gamma}}$. The popularities of contents are changed randomly across the time slots. We set $U = 600$, $F = 200$, and $T = 24$ with length of one hour for each time slot [18]. The sizes of files are uniformly generated within interval $[1, 10]$. The cache capacity is set as $S = \rho \sum_{f \in \mathcal{F}} l_f$. Here, $\rho \in [0, 1]$ shows the size of cache in relation to the total size of all contents. The number of requests for each user is randomly generated in $[1, 15]$.

The performance results are reported in Figures 3-5. The deviation from global optimum is bounded by the deviation from the LB, as LB is always less than or equal to the global optimum. We refer to the deviation from LB as optimality gap. The CGA provides solutions within 1% gap from LB and outperforms the conventional algorithms. Figure 3 shows the impact of U . When U increases from 400 to 800 the cost nearly linearly increases, however, the optimality gap of algorithms decreases. The reason is that with larger number of users, more contents from the content set are requested by users. As the cache capacity is limited, the only way to get

many of requested contents is from the server by all algorithms which leads to a lower optimality gap.

Figure 4 shows the impact of F . Recall that the cache capacity is set to 50% of the total size of the files. For CGA, when $F = 50$ the capacity of cache is extremely limited and as F is small, almost all contents will be requested by users. These together imply that many requests need to be satisfied from the server which leads to a high cost. When F increases to 150, the cost decreases. Because as F increases the cache capacity increases, and CGA is able to efficiently utilize the cache capacity. However when F further increases to 300, the cost increases. The reason is that even though the capacity increases with F but the diversity of requested contents becomes too large, and consequently some of them need to be satisfied from the server which leads to a higher cost.

Figure 5 shows the impact of λ . Recall that larger λ means higher backhaul load but smaller AoI. From the figure, it can be seen that when λ grows, PBA and RBA push down the average AoI of contents to almost zero but incur substantial amount of load on the backhaul. In contrast, the solutions of CGA achieve a much better balance between the backhaul load and AoI of contents with respect to λ . Note that the backhaul load and average AoI are normalized to interval $[0, 100]$.

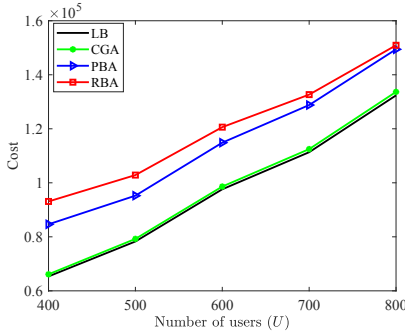


Figure 3. Impact of U on cost when $T = 24$, $F = 200$, $\lambda = 0.5$, $\rho = 0.5$, $\gamma = 0.54$, $c_s = 10$, and $c_b = 1$.

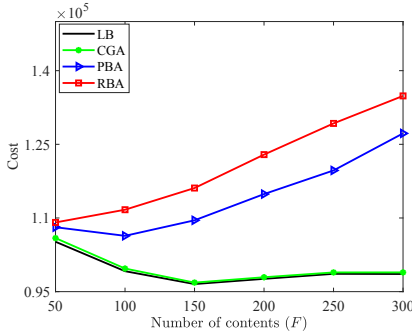


Figure 4. Impact of F on cost when $T = 24$, $U = 600$, $\lambda = 0.5$, $\rho = 0.5$, $\gamma = 0.54$, $c_s = 10$, and $c_b = 1$.

VI. CONCLUSIONS

This paper has investigated scheduling of content caching along time where jointly offloading effect and freshness of the contents are accounted for. The problem is formulated as an ILP and \mathcal{NP} -hardness of the problem is proved. Next, via a mathematical reformulation, a solution approach based on column generation and a rounding mechanism is developed. Via the joint cost function, it is possible to address the trade-off between the updating and AoI costs. The numerical

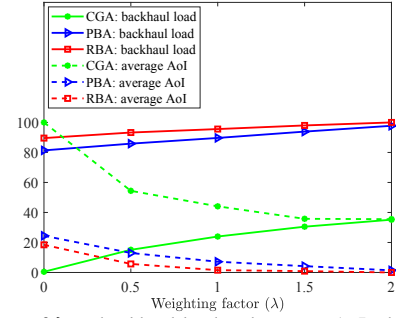


Figure 5. Impact of λ on backhaul load and average AoI when $T = 24$, $U = 600$, $F = 200$, $\rho = 0.5$, $c_s = 10$, and $c_b = 1$.

results show that our algorithm is able to balance between the two costs. Simulation results demonstrated that our solution approach provides near-optimal solutions.

REFERENCES

- [1] L. Li, G. Zhao, and R. S. Blum, "A survey of caching techniques in cellular networks: Research issues and challenges in content placement and delivery strategies," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 3, pp. 1710–1732, 2018.
- [2] M. K. Kiskani and H. R. Sadjadpour, "Capacity of cellular networks with femtocache," in *IEEE INFOCOM Wkshps*, 2016, pp. 9–14.
- [3] T. Deng, G. Ahani, P. Fan, and D. Yuan, "Cost-optimal caching for D2D networks with user mobility: Modeling, analysis, and computational approaches," *IEEE Trans. Wireless Commun.*, vol. 17, no. 5, pp. 3082–3094.
- [4] P. Blasco and D. Gndz, "Learning-based optimization of cache content in a small cell base station," in *IEEE ICC*, 2014, pp. 1897–1903.
- [5] M. A. Kader, E. Bastug, M. Bennis, E. Zeydan, A. Karatepe, A. S. Er, and M. Debbah, "Leveraging big data analytics for cache-enabled wireless networks," in *IEEE GLOBECOM Wkshps*, 2015, pp. 1–6.
- [6] B. N. Bharath, K. G. Nagananda, D. Gndz, and H. V. Poor, "Caching with time-varying popularity profiles: A learning-theoretic perspective," *IEEE Trans. Commun.*, vol. 66, no. 9, pp. 3837–3847, 2018.
- [7] N. Zhang, K. Zheng, and M. Tao, "Using grouped linear prediction and accelerated reinforcement learning for online content caching," in *IEEE ICC*, 2018, pp. 1–6.
- [8] Y. Sun, E. Uysal-Biyikoglu, R. D. Yates, C. E. Koksal, and N. B. Shroff, "Update or wait: How to keep your data fresh," *IEEE Trans. Inf. Theory*, vol. 63, no. 11, pp. 7492–7508, 2017.
- [9] H. Tang, P. Ciblat, J. Wang, M. Wigger, and R. Yates, "Age of information aware cache updating with file- and age-dependent update durations," <https://arxiv.org/pdf/1909.05930.pdf>, 2019.
- [10] C. Kam, S. Kompella, G. D. Nguyen, J. E. Wieselthier, and A. Ephremides, "Information freshness and popularity in mobile caching," in *IEEE ISIT*, 2017, pp. 136–140.
- [11] R. D. Yates, P. Ciblat, A. Yener, and M. Wigger, "Age-optimal constrained cache updating," in *IEEE ISIT*, 2017, pp. 141–145.
- [12] M. S. Heydar Abad, E. Ozfatura, O. Ercetin, and D. Gndz, "Dynamic content updates in heterogeneous wireless networks," in *IEEE/IFIP 15th WONS*, 2019, pp. 107–110.
- [13] M. R. Garey and D. S. Johnson, *Computers and intractability: A guide to the theory of NP-completeness*. W. H. Freeman, 1979.
- [14] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to algorithms*. Third edition, The MIT Press, 2009.
- [15] J. Elias and B. Blaszczyszyn, "Optimal geographic caching in cellular networks with linear content coding," in *15th WiOpt*, 2017, pp. 1–6.
- [16] H. Ahlehagh and S. Dey, "Video-aware scheduling and caching in the radio access network," *IEEE/ACM Trans. Netw.*, vol. 22, no. 5, pp. 1444–1462, Oct. 2014.
- [17] K. Shanmugam, N. Golrezaei, A. G. Dimakis, A. F. Molisch, and G. Caire, "Femtocaching: Wireless content delivery through distributed caching helpers," *IEEE Trans. Inf. Theory*, vol. 59, no. 12, pp. 8402–8413, 2013.
- [18] G. Ahani and D. Yuan, "On optimal proactive and retention-aware caching with user mobility," in *IEEE 88th VTC-Fall*, 2018, pp. 1–5.