4-16-2018

Algorithms for Asynchronous Coded Caching

Hooshang Ghasemi
Iowa State University, ghasemi@iastate.edu

Aditya Ramamoorthy
Iowa State University, adityar@iastate.edu

Algorithms for Asynchronous Coded Caching

Abstract
The original formulation of the coded caching problem assumes that the file requests from the users are synchronized, i.e., they arrive at the server at the same time. Several subsequent contributions work under the same assumption. Furthermore, the majority of prior work does not consider a scenario where users have deadlines. In our previous work we formulated the asynchronous coded caching problem where user requests arrive at different times. Furthermore, the users have specified deadlines. We proposed a linear program for obtaining its optimal solution. However, the size of the LP (number of constraints and variables) grows rather quickly with the number of users and cache sizes. In this work, we explore a dual decomposition based approach for solving the LP under consideration. We demonstrate that the dual function can be evaluated by equivalently solving a number of minimum cost network flow algorithms. Minimum cost network flow algorithms have been the subject of much investigation and current solvers routinely solve instances with millions of nodes in minutes. Our proposed approach leverages these fast solvers and allows us to solve several large scale instances of the asynchronous coded caching problem with manageable time and memory complexity.

Keywords
Servers, Delays, Encoding, Schedules, Synchronization, Linear programming, Indexes

Disciplines
Electrical and Computer Engineering | Signal Processing | Systems and Communications

Comments
This is a manuscript of a proceeding published as Ghasemi, Hooshang, and Aditya Ramamoorthy. "Algorithms for asynchronous coded caching." In *2017 51st Asilomar Conference on Signals, Systems, and Computers,* (2017): 636-640.

Algorithms for Asynchronous Coded Caching

Hooshang Ghasemi and Aditya Ramamoorthy
Department of Electrical and Computer Engineering, Iowa State University, Ames IA 50011 U.S.A.
Email: {ghasemi,adityar} @iastate.edu

***Abstract*—The original formulation of the coded caching problem assumes that the file requests from the users are synchronized, i.e., they arrive at the server at the same time. Several subsequent contributions work under the same assumption. Furthermore, the majority of prior work does not consider a scenario where users have deadlines. In our previous work we formulated the asynchronous coded caching problem where user requests arrive at different times. Furthermore, the users have specified deadlines. We proposed a linear program for obtaining its optimal solution. However, the size of the LP (number of constraints and variables) grows rather quickly with the number of users and cache sizes. In this work, we explore a dual decomposition based approach for solving the LP under consideration. We demonstrate that the dual function can be evaluated by equivalently solving a number of minimum cost network flow algorithms. Minimum cost network flow algorithms have been the subject of much investigation and current solvers routinely solve instances with millions of nodes in minutes. Our proposed approach leverages these fast solvers and allows us to solve several large scale instances of the asynchronous coded caching problem with manageable time and memory complexity.**

I. Introduction

Caching is a core component of solving the problem of large scale content delivery over the Internet. Traditional caching works by placing popular content closer to the end users. The work of [1] considered the usage of coding in the caching problem and demonstrated that significant reductions in the induced network traffic were possible.

However, the original formulation of the coded caching problem assumes that all file requests from the users arrive at the server at the same time, i.e., it works with an (idealized) synchronized model. From a practical perspective, it is important to consider the case when the requests of the users are not synchronized. We studied this "asynchronous coded caching" problem in [2]. In the asynchronous scenario, a simple strategy would be to wait for the last request to arrive and then apply the scheme of [1]. While such a strategy will result in low overall rate of transmission from the server, the delay experienced by the users will essentially be dominated by the arrival time of the last request. Thus, certain end users may experience unacceptable delays.

In our prior work [2] we considered both the offline and the online variants of this problem. Each user has a specific deadline by which his/her demand needs to be satisfied. In the offline scenario, where the server knows the arrival times and deadlines of each user in advance, we posed a linear programming (LP) problem which if feasible, allows the server to determine a schedule of transmissions, such that each user can be satisfied within its deadline. In this work we make further progress on this problem.

1) Main contributions: Fast algorithms based dual decomposition. The size of the LP in [2] grows very quickly with the problem parameters and solving it is impractical for large scale instances. In this work we demonstrate that we can instead work with the dual of an equivalent LP. The dual function can be evaluated by solving a set of minimum cost network flow problems. Minimum cost network flow problems have been the subject of much investigation in the optimization literature and large scale instances can be solved very quickly [3]. We present results that indicate that significant time savings are obtained by applying our approach. Moreover, our results indicate that the coded caching rate degrades quite gracefully in the presence of asynchronism.

2) Related Work: The delay sensitive coded caching problem was first studied in [4]. They considered the decentralized coded caching model, and considered a situation where each subfile has a specific deadline. Only the online case was considered and heuristics for transmission from the server were proposed. The heuristics are found to have good performance. However, the transmission time for each packet was not considered in their formulation. Our LP formulation can be viewed as a bound on the possible performance of any online scheme. The work of [5] investigated the problem of updating the cache content in the coded caching context; however synchronous file requests were considered. We note that another important practical issue within the coded caching domain includes subpacketization [6], i.e., the requirement that each file needs to be subdivided into a large number of parts in the original scheme. Some recent work addressing these issues can be found in [7]–[11].

II. Problem Formulation

A coded caching system contains a server with N files, denoted W_i, $i = 1, \dots, N$, each of size F subfiles, where a subfile is a basic unit of storage. The system also contains K users each connected to the server through an error free, broadcast shared link. Each of the users is equipped with a local cache of size MF subfiles; we denote the cache content of user i by Z_i which is a function of $W_1, \dots, W_N$. The system operates in two distinct phases. In the *placement phase* the content of the caches is populated by the server. This phase does not depend on the future requests of the users which are assumed to be arbitrary. In the *delivery phase* each user makes a request and the server transmits potentially coded signals to satisfy the requests of the users.

This work was supported in part by the National Science Foundation by grants CCF- 1320416, CCF- 1149860 and CCF-1718470.

Our assumption is that a specific uncoded placement scheme is being used by the coded caching system. It is known that the delivery phase in this case corresponds to an index coding problem [12]. In general, the optimal solution for an arbitrary index coding problem is known to be hard [12]. However, techniques such as clique cover on the side information graph are well-recognized to have good performance [12]. In particular, the delivery phase in [1] is precisely a clique cover on their side information graph, assuming worst case demands. Each transmitted equation is such that a certain number of users benefit from it simultaneously. We assume that our delivery phase in the asynchronous setting also transmits equations of this type.

We describe our approach by assuming that the system works with the placement and delivery scheme of [1]. However, our proposed approach is general and can work with any uncoded placement scheme. In particular, we assume a system with K users where each user has a cache of size MF subfiles. The server contains $N \geq K$ files[1] denoted $W_n, n = 1, \dots, N$. Let $t = KM/N$ be an integer. For this system, $F = \binom{K}{t}$, i.e., each file is divided in $\binom{K}{t}$ subfiles corresponding to t-sized subsets of $[K]$. Thus file $W_n = \{W_{n,\mathcal{A}} : \mathcal{A} \subset [K], |\mathcal{A}| = t\}$. User i caches all subfiles $W_{n,\mathcal{A}}$ where $i \in \mathcal{A}$.

We assume that time $\tau \geq 0$ is slotted. Let $[n]$ denote the set $\{1, \dots, n\}$ and the symbol $\oplus$ represent the XOR operation. We say that an equation E is of type *all-but-one* if $E = \oplus_{l=1}^{\ell} W_{d_{i_l}, \mathcal{A}_{i_l}}$ where for each $l \in [\ell]$, we have $i_l \notin \mathcal{A}_{i_l}$ and $i_l \in \mathcal{A}_{i_j}$ for all $j \in [\ell] \setminus \{l\}$. It is easy to see that the equations transmitted in the delivery phase in [1] are of the all-but-one type. With this specified placement, the asynchronous coded caching problem with deadlines can be formulated as follows.
Inputs.

- *User requests.* User i requests file W_{d_i}, with $d_i \in [N]$. User i's request arrives at the server at time T_i.
- *Deadlines.* The i-th user needs to be satisfied by time $T_i + \Delta_i$, where Δ_i is a positive integer.
- *Transmission delay.* We assume that the size of each subfile is such that it needs r time-slots to be transmitted over the shared link, i.e., each subfile can be treated as equivalent to r packets, where each packet can be transmitted in one time slot.

As the problem is symmetric with respect to users, w.l.o.g. we assume that $T_1 \leq T_2 \leq \ldots \leq T_K$. Let $T_{\max} = \max_i (T_i + \Delta_i)$. Note that upon sorting the set of arrival times and deadlines, i.e., $\cup_{i=1}^{K} \{T_i, T_i + \Delta_i\}$, we can divide the interval $[T_1, T_{\max})$ into *at most* $2K - 1$ intervals. Let the integer β, where $1 \leq \beta \leq 2K - 1$ denote the number of intervals. Let $\Pi_1, \dots, \Pi_\beta$ represent the intervals where Π_i appears before Π_j if $i < j$. The intervals are left-closed and right-open. An easy to see but very useful property of the intervals that we have defined is that for a given i, either $[T_i, T_i + \Delta_i) \cap \Pi_\ell = \Pi_\ell$ or $[T_i, T_i + \Delta_i) \cap \Pi_\ell = \emptyset$. We define

$$U_\ell = \{i \in [K] : \ [T_i, T_i + \Delta_i) \cap \Pi_\ell = \Pi_\ell\}, \text{and}$$
$$D_\ell = \{d_i \in [N] : \ i \in U_\ell\}.$$

[1]We assume that $N \geq K$ as it corresponds to the worst case rate and is also the more practical scenario.

Thus, U_ℓ is the set of active users in time interval Π_ℓ and D_ℓ is the corresponding set of active file requests.
Outputs.

- *Transmissions at each time slot.* If the problem is feasible, the schedule specifies which equations need to be transmitted at each time. The equations need to be of the all-but-one type. The schedule is such that each user can recover all its missing subfiles within its deadline. The equations transmitted at time $\tau \in \Pi_\ell$ only depend on D_ℓ.

We work with fractional solutions, i.e., we assume that each packet that is transmitted over the shared edge can be subdivided as finely as needed. Thus, in each time slot we could transmit multiple equations that may serve potentially different subsets of users. This assumption is reasonable if the underlying subfiles and hence the packets are quite large. In this work we only consider the offline scenario, i.e., we assume that the server is aware of $\{T_i, \Delta_i, d_i\}_{i=1}^{K}$ at $\tau = 0$. However, at time $\tau \in \Pi_\ell$ the transmitted equation(s) will only depend on D_ℓ, i.e., the server cannot start sending missing subfiles for a given user until its request arrives.

III. Offline asynchronous coded caching

We present the following example to clarify the problem inputs and outputs.

Example 1: Consider a scenario with $N = K = 3$, $M = 1$, and $r = 1$. We assume that $T_i = i$ and $\Delta_i = 2$ for $i \in \{1, 2, 3\}$. The server contains three files A, B, and C. According to placement scheme in [1], each file is divided to $\binom{K}{t} = 3$ subfiles and user i caches $Z_i = \{A_i, B_i, C_i\}$. We assume that the first user requests file A, the second user requests file B, and the third user requests file C.

There are four time intervals with $\Pi_i = [i, i+1)$ for $i = 1, \dots, 4$. An offline solution for this problem corresponds to the server transmitting equations A_3 at Π_1, $A_2 \oplus B_1$ at Π_2, $B_3 \oplus C_2$ at Π_3, and finally C_1 at Π_4. It can be observed that each user can recover the missing subfiles that they need. The solution is shown in Fig. 1.
In our prior work [2], we proposed a LP for the asynchronous coded caching problem. This LP is discussed below.

A. Linear programming formulation

Let $\Omega = \{\mathcal{A} : \mathcal{A} \subset [K], |\mathcal{A}| = t\}$, so that it represents the indices of all the subfiles. Let $\Omega^{(i)} = \{\mathcal{A} : \mathcal{A} \in \Omega, i \notin \mathcal{A}\}$ represent the indices that are not cached by user i. Recall that U_ℓ is the set of active users in time interval Π_ℓ and D_ℓ represents their file requests. Let $\mathcal{U}_\ell$ be the set of all nonempty subsets $U \subseteq U_\ell$ with $|U| \leq t+1$. In the subsequent discussion we call such a U, a user group. In time interval Π_ℓ, a user group $U \in \mathcal{U}_\ell$ represents a collection of users that can be serviced simultaneously by the server. For any $U \subseteq [K]$, let $\mathcal{I}_U$ be the set of indices of all time intervals where the users in U are simultaneously active, i.e., $\mathcal{I}_U = \{\ell : \ [T_i, T_i + \Delta_i) \cap \Pi_\ell \neq \emptyset, \ \forall \ i \in U\}$.

For each missing subfile $W_{d_i, \mathcal{A}}$ (where $\mathcal{A} \in \Omega^{(i)}$) we let $\mathcal{U}_{\{i, \mathcal{A}\}}$ be the set of user groups where it can be transmitted, i.e., $\mathcal{U}_{\{i, \mathcal{A}\}} = \{U \in \cup_{\ell=1}^{\beta} \mathcal{U}_\ell : \ i \in U, \ \ U \setminus \{i\} \subseteq \mathcal{A}\}$. We note here that for a fixed i, there are potentially multiple subfiles

$\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_l \in \Omega^{(i)}$ such that $U \in \mathcal{U}_{\{i,\mathcal{A}_j\}}$ for $j = 1, \ldots, l$. For example, suppose that $K = 4, t = 2$ and let $U = \{1, 2\}$. In this case $U \in \mathcal{U}_{1,\{2,3\}}$ and $U \in \mathcal{U}_{1,\{2,4\}}$. Thus, user group U can be used to potentially transmit different missing subfiles needed by user i.

We let $|\Pi_\ell|$ denote the length of the time interval Π_ℓ. For each time interval Π_ℓ with $\ell = 1, \ldots, \beta$ and for each $U \in \mathcal{U}_\ell$ we define variable $x_U(\ell) \in [0, |\Pi_\ell|]$ that represents the portion of time interval Π_ℓ that is allocated to an equation that benefits user group U (more details on the reasoning underlying the variables in the LP can be found in [2]).

For each missing subfile $W_{d_i,\mathcal{A}}$ and each $U \in \mathcal{U}_{\{i,\mathcal{A}\}}$ we define variable $y_{\{i,\mathcal{A}\}}(U) \in [0, r]$ that represents the portion of the missing subfile $W_{d_i,\mathcal{A}}$ transmitted within some or all of the equations associated with $x_U(\ell)$ for $\ell \in \mathcal{I}_U$. As pointed out before, for a fixed i, U can be used to transmit different missing subfiles needed by user i. However, a single equation can only help recover one missing subfile needed by i. Thus, $\sum_{\ell \in \mathcal{I}_U} x_U(\ell)$ must be shared between the appropriate $y_{\{i,\mathcal{A}\}}(U)$'s.

Example 2: For the system in Example 1, we have $\Omega^{(1)} = \{A_2, A_3\}$, $\Omega^{(2)} = \{B_1, B_3\}$, and $\Omega^{(3)} = \{C_1, C_2\}$. Also, the active users at Π_2 are $U_2 = \{1, 2\}$ thus $\mathcal{U}_2 = \{\{1\}, \{1, 2\}, \{2\}\}$.

In [2], we proposed the following LP that minimizes the overall rate of transmission from the server in the offline scenario.

$$\min \sum_{\ell=1}^{\beta} \sum_{U \in \mathcal{U}_\ell} x_U(\ell) \tag{1}$$

$$\begin{aligned}
\text{s.t.} \quad & \sum_{U \in \mathcal{U}_\ell} x_U(\ell) \le |\Pi_\ell|, && \forall\, \ell = 1, \ldots, \beta, \\
& \sum_{\mathcal{A} \in \mathcal{B}_{\{i,U\}}} y_{\{i,\mathcal{A}\}}(U) \le \sum_{\ell \in \mathcal{I}_U} x_U(\ell), && \forall\, U \in \mathcal{V}_i,\ \forall i \in [K], \\
& \sum_{U \in \mathcal{U}_{\{i,\mathcal{A}\}}} y_{\{i,\mathcal{A}\}}(U) = r, && \forall\, \mathcal{A} \in \Omega^{(i)},\ \forall i \in [K], \\
& 0 \le x_U(\ell) \le |\Pi_\ell|, \text{ and} \\
& 0 \le y_{\{i,\mathcal{A}\}}(U) \le r.
\end{aligned}$$

where $\mathcal{V}_i = \{U \in \cup_{\ell=1}^{\beta} \mathcal{U}_\ell : \ i \in U \text{ and } |U| \le t+1\}$ and $\mathcal{B}_{\{i,U\}} = \{\mathcal{A} : \mathcal{A} \in \Omega^{(i)}, U \in \mathcal{V}_i, U \setminus \{i\} \subseteq \mathcal{A}\}$.

In Section III.A.1. of [2], we show that a feasible solution of the above LP can be interpreted as a coding solution for the offline asynchronous problem. In particular, assuming that subfiles can be subdivided finely enough, we show that the LP can be used to arrive at a set of all-but-one equations that can be transmitted by the server such that each user is satisfied.

The main issue with the LP formulation presented above is complexity. The number of variables and constraints grows very quickly with the problem parameters. Both the number of variables and the number of constraints grows proportional to $K\binom{K-1}{t}\sum_{j=0}^{t+1}\binom{K}{j}$; a more precise analysis can be found in [2]. The worst case complexity of solving a LP is cubic in the problem size. Therefore, the solving the LP in (1) is not practical for a large values of K and t.

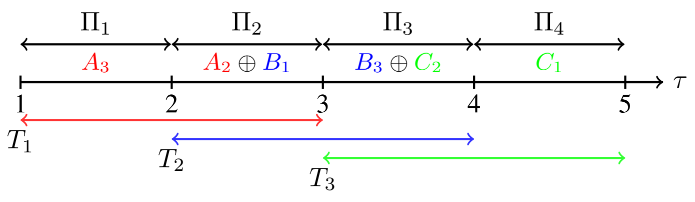

Fig. 1: Offline solution corresponding to the Example 1. The double-headed arrows show the active time slots for each user. The transmitted equations are shown above the timeline. The intervals Π_1, Π_2, Π_3, and Π_4 are allocated in entirety to user groups $\{1\}$, $\{1, 2\}$, $\{2, 3\}$, and $\{3\}$ respectively; therefore, $x_{\{1\}}(1)$, $x_{\{1,2\}}(2)$, $x_{\{2,3\}}(3)$, and $x_{\{3\}}(4)$ are set to one. Furthermore, the solution of Example 1 (where $r = 1$) is such that missing subfiles of the first user (A_3 and A_2) are transmitted within user groups $\{1\}$ and $\{1, 2\}$ respectively so that $y_{\{1,A_3\}}(\{1\}) = y_{\{1,A_2\}}(\{1, 2\}) = 1$. Similarly, we have $y_{\{2,B_1\}}(\{1, 2\}) = y_{\{2,B_3\}}(\{2, 3\}) = 1$ and $y_{\{3,C_2\}}(\{2, 3\}) = y_{\{3,C_1\}}(\{3\}) = 1$.

The main contribution of our work is to demonstrate that the complexity can be made quite manageable by instead solving the dual of an equivalent LP.

IV. Dual Decomposition based approach

The overall idea of reducing the complexity of the LP in (1) is to formulate an equivalent LP and work with its dual. The structure of the corresponding dual function is such that it can be evaluated by solving a system of *decoupled minimum cost network flow optimizations*. Minimum cost network flow is a well investigated problem and optimized solutions for it routinely allow problems with over a million nodes to be solved in a few minutes [3].

As it stands, the LP in (1) cannot be interpreted as network flow. Yet, intuitively one can view the missing subfiles from each user as flowing through the user groups and getting absorbed in sinks that correspond to their valid time slots. However, the flows corresponding to different users can be shared as the all-but-one equations allow different users to benefit from the same equation. We note here that a similar sharing of flows also occurs in the problem of minimum cost multicast with network coding [13]. The LP in (1) can however be modified slightly so that an appropriate decoupling can be exploited in the dual program. Towards this end, we introduce new variables $x_U^{(i)}(\ell)$ in the original LP. For $i = 1, \ldots, K$, let $\mathcal{C}_i$ denote the set of constraints:

$$\begin{aligned}
& \sum_{(i,\mathcal{A}) \in \mathcal{B}_{\{i,U\}}} y_{\{i,\mathcal{A}\}}(U) = \sum_{\ell \in \mathcal{I}_U} x_U^{(i)}(\ell), && \forall\, U \in \mathcal{V}_i, i \in [K], \\
& \sum_{U \in \mathcal{U}_{\{i,\mathcal{A}\}}} y_{\{i,\mathcal{A}\}}(U) = r, && \forall\, \mathcal{A} \in \Omega^{(i)},\ i \in [K], \\
& 0 \le x_U^{(i)}(\ell) \le |\Pi_\ell|, && \forall\, i \in U,\ U \in \mathcal{U}_\ell,\ \ell \in [\beta], \\
& 0 \le y_{\{i,\mathcal{A}\}}(U) \le r, && \forall\, U \in \mathcal{U}_{\{i,\mathcal{A}\}},\ \mathcal{A} \in \Omega^{(i)},\ i \in [K].
\end{aligned}$$

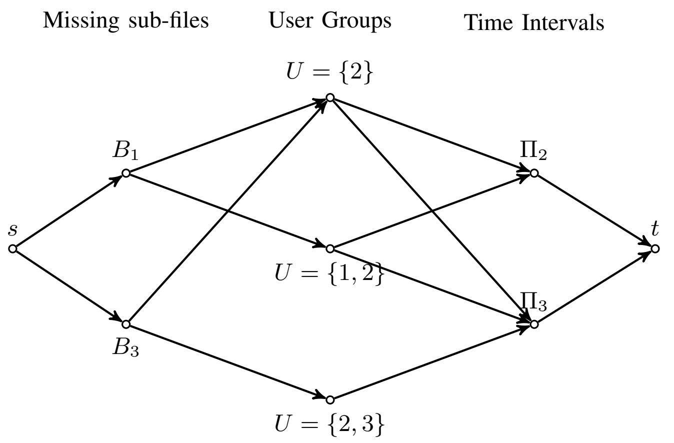

Fig. 2: Minimum cost flow network associated with subproblem (5) corresponding to the second user. The edges from s to $B_i, i = 1, 3$ have a capacity of $r = 1$ and the edges from $\Pi_i, i = 2, 3$ to t have a capacity of $|\Pi_i|$, which in this case equals 1.

Then, the original LP can be compactly rewritten as

$$\begin{aligned} \min \quad & \sum_{\ell=1}^{\beta} \sum_{U \in \mathcal{U}_\ell} x_U(\ell) && (2) \\ \text{s.t.} \quad & x_U^{(i)}(\ell) \le x_U(\ell) \qquad \forall\, i \in U,\ \forall\, U \in \mathcal{U}_\ell,\ \forall\, \ell \in [\beta], \\ & \sum_{U \in \mathcal{U}_\ell} x_U(\ell) \le |\Pi_\ell|, \qquad \forall\, \ell \in [\beta], \\ & \mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_K. \end{aligned}$$

The only difference in the above LP is the introduction of variables $x_U^{(i)}(\ell)$ (for appropriate ranges of i, U and ℓ) such that the second set of inequality constraints in (1) are replaced by equality constraints. Moreover, the original constraints are maintained by setting $x_U^{(i)}(\ell) \le x_U(\ell)$. It is not too hard to see that the two LPs are equivalent.

A. Dual problem

We proceed by considering the dual of the LP in (2) with respect to the constraints that involve the variables $x_U(\ell)$. The Lagrangian $\mathcal{L}(\mathbf{x}, \{\lambda_U^{(i)}(\ell)\}_{i \in U, U \in \mathcal{U}_\ell, \ell \in [\beta]}, \{\zeta_\ell\}_{\ell \in [\beta]})$ can be expressed as

$$\begin{aligned} \mathcal{L} = & \sum_{\ell=1}^{\beta} \sum_{U \in \mathcal{U}_\ell} x_U(\ell) && (3) \\ & + \sum_{\ell=1}^{\beta} \sum_{U \in \mathcal{U}_\ell} \sum_{i \in U} \lambda_U^{(i)}(\ell) \left(x_U^{(i)}(\ell) - x_U(\ell) \right) \\ & + \sum_{\ell=1}^{\beta} \zeta_\ell \left(\sum_{U \in \mathcal{U}_\ell} x_U(\ell) - |\Pi_\ell| \right) \end{aligned}$$

where $\mathbf{x}$ represents all the primal variables in the LP in (1) and $\lambda_U^{(i)}(\ell)$'s and ζ_ℓ's are nonnegative. The dual function $g(\{\lambda_U^{(i)}(\ell)\}_{i \in U, U \in \mathcal{U}_\ell, \ell \in [\beta]}, \{\zeta_\ell\}_{\ell \in [\beta]})$ is therefore obtained by solving for

$$\begin{aligned} & \min \mathcal{L} && (4) \\ & \text{s.t.} \quad C_1, C_2, \ldots, C_K, \end{aligned}$$

where the minimization is over the primal variables. It is evident that the dual function g takes a nontrivial value only if

$$\sum_{i \in U} \lambda_U^{(i)}(\ell) = 1 + \zeta_\ell, \qquad \forall\, U \in \mathcal{U}_\ell,\ \ell \in [\beta].$$

The evaluation of g at a fixed set of dual variables $\{\lambda_U^{(i)}(\ell)\}_{i \in U, U \in \mathcal{U}_\ell, \ell \in [\beta]}, \{\zeta_\ell\}_{\ell \in [\beta]})$ can therefore be written as

$$\begin{aligned} \min \quad & \sum_{\ell=1}^{\beta} \sum_{U \in \mathcal{U}_\ell} \sum_{i \in U} \lambda_U^{(i)}(\ell) x_U^{(i)}(\ell) - \sum_{\ell=1}^{\beta} \zeta_\ell |\Pi_\ell| \\ \text{s.t.} \quad & C_1, C_2, \ldots, C_K. \end{aligned}$$

This minimization can be addressed by solving K independent minimum cost network flow problems corresponding to each of the constraints $\mathcal{C}_i, i = 1, \ldots, K$. This can be seen as follows. Each subproblem for $i = 1, \ldots, K$ has the following structure.

$$\begin{aligned} \min \quad & \sum_{U \in \mathcal{V}_i} \sum_{\ell \in \mathcal{I}_U} \lambda_U^{(i)}(\ell) x_U^{(i)}(\ell) && (5) \\ \text{s.t.} \quad & \mathcal{C}_i. \end{aligned}$$

The subproblem in (5) is a standard minimum-cost flow problem. The associated flow network contains a source node and three intermediate layers followed by a terminal node (see Fig. 2). The layers are such that the nodes in the first layer correspond to missing subfiles in $\Omega^{(i)}$, the nodes in the second layer correspond to user groups in $\mathcal{V}_i$ and the nodes in the third layer correspond to the time interval $\Pi_\ell, \ell = 1, \ldots, \beta$.

In this flow network a zero cost is assigned to all edges except those from the user group nodes to the time intervals. For the i-th flow network (corresponding to the constraints in $\mathcal{C}_i$) the cost of the edge between user group U and time interval Π_ℓ is $\lambda_U^{(i)}(\ell)$. The edges between time interval Π_ℓ and the terminal node have a capacity constraint of $|\Pi_\ell|$ and the edges between the source node and the missing subfiles have a capacity constraint of r; the other edges have no capacity constraints. The variable $x_U^{(i)}(\ell)$ is the amount of the flow carried by the edge from user group U to time interval Π_ℓ. Furthermore, the supply of the source node and terminal node are $|\Omega_i|r$ and $-|\Omega_i|r$ respectively. The supply of the other nodes are zero. An instance of this flow network for a user in asynchronous coded caching of system in Example 1 is shown in Fig. 2.

Finally, the dual function g is maximized by a subgradient search and the original primal variables are recovered by the methods described in [14]; these were also used in [13]. It turns out that even this subgradient search can be simplified by using the underlying structure of the problem. The details are omitted owing to lack of space.

(K,t)	# nodes	# edges	exe. time (min)	exe. time original (min)
$(100,2)$	$986,161$	$17,643,986$	$8,026$	—
$(20,4)$	$178,542$	$1,778,703$	1065	—
$(40,2)$	$61,959$	$567,780$	63	—
$(20,2)$	$7,542$	$43,507$	1.9	21.9
$(10,4)$	$3,917$	$29,369$	0.8	5.33
$(10,2)$	915	$3,866$	0.08	0.03

TABLE I: Execution time for solving the LP using our approach; we run 1000 iterations of subgradient ascent. Columns 2 & 3 indicate the size of the associated flow network. The table is ordered by the number of nodes in the flow network.

V. Simulations results

In our experiments we generate the arrival times, $T_1, \ldots, T_K$, according to a Poisson process with parameter λ and then quantize them to the nearest integer. The deadlines, $\Delta_1, \ldots, \Delta_K$ are generated from a uniform distribution so that Δ_i's, $i = 1, \ldots, K$ are uniformly chosen from integers in the interval $[\Delta_{\min},\ \Delta_{\max}]$.

In the first set of simulations we examine the execution time of our approach for various values of (K,t). In these simulations we set $r = 1$, $\lambda = 0.4$, $\Delta_{\min} = \binom{K-1}{t}$, and $\Delta_{\max} = \binom{K}{t+1}$. Note that each user needs $\binom{K-1}{t}$ subfiles. Therefore, it needs at least $\Delta_{\min}$ time slots to receive the its missing subfiles and otherwise the problem will certainly be infeasible. On the other hand, the delivery scheme of the synchronized scenario allows all users to be supported within $\Delta_{\max}$ time slots. This explains the choice of $\Delta_{\min}$ and $\Delta_{\max}$. Table I shows the details of the overall execution time and the size of the corresponding flow networks for the various instances. The last column of the table corresponds to the execution time (in MATLAB) of the LP in (1), while the second-last column corresponds to the execution time of the proposed approach above. It is evident that the proposed approach is significantly faster. Furthermore, memory requirements make infeasible to even formulate the problems corresponding to the first three rows in MATLAB.

In the next set of simulations, we explore the average rate for different values of λ. The value of λ is inversely proportional to the average spacing between the request arrival times. As can be seen in Fig. 3, the rate gradually increases when the spread in arrival times increases. Thus, under mild asynchronism much of the gains of coded caching can be leveraged. Finally, Fig. 4 shows the convergence of the primal recovery to the actual rate for a system with $N = K = 10$, $t = 4$, and $r = 1$. It can be observed that there is a clear convergence of the solution to the optimal value.

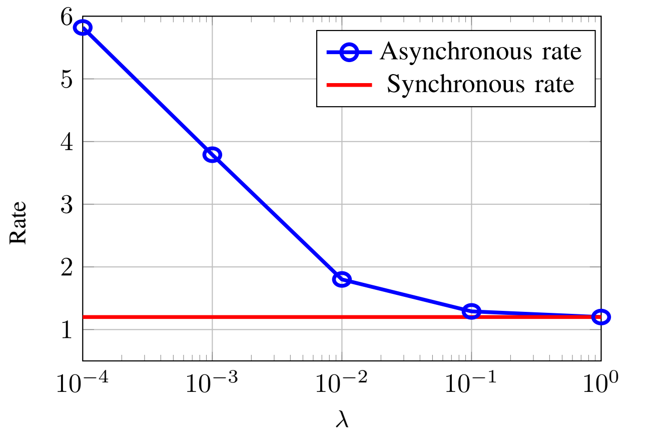

Fig. 3: Rate of an asynchronous system (blue line) and a synchronous system (red line) vs. λ. System parameters are $K = 10$, $t = 4$.

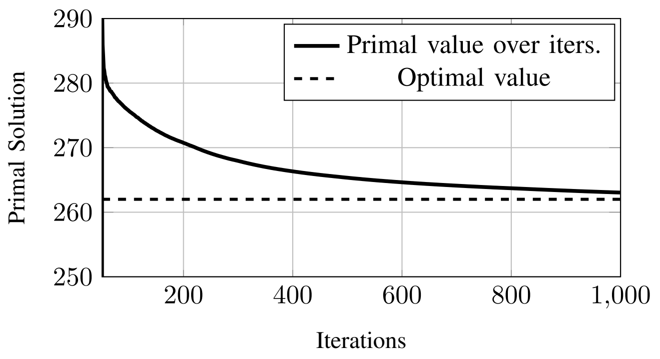

Fig. 4: Convergence of primal recovery to the optimal solution for a system with $N = K = 10$, $r = 1$, and $t = 4$. Dashed line is the optimal value obtained by solving (1).

References

[1] M. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *IEEE Trans. on Info. Th.*, vol. 60, no. 5, pp. 2856–2867, May 2014.

[2] H. Ghasemi and A. Ramamoorthy, "Asynchronous coded caching," *IEEE Intl. Symposium on Info. Th.*, 2017.

[3] P. Kovacs, "Minimum-cost flow algorithms: an experimental evaluation," *Optimization Methods and Software*, vol. 30, no. 1, pp. 94–127, 2015.

[4] U. Niesen and M. A. Maddah-Ali, "Coded caching for delay-sensitive content," in *IEEE Intl. Conf. Comm.*, 2015, pp. 5559–5564.

[5] R. Pedarsani, M. A. Maddah-Ali, and U. Niesen, "Online coded caching," *IEEE/ACM Trans. Netw.*, vol. 24, no. 2, pp. 836–845, 2016.

[6] K. Shanmugam, M. Ji, A. M. Tulino, J. Llorca, and A. G. Dimakis, "Finite length analysis of caching-aided coded multicasting," in *52nd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2014, pp. 914–920.

[7] Q. Yan, M. Cheng, X. Tang, and Q. Chen, "On the placement delivery array design in centralized coded caching scheme," 2015 [Online] Available: http://arxiv.org/abs/1510.05064.

[8] L. Tang and A. Ramamoorthy, "Coded Caching with Low Subpacketization Levels," in *Workshop on Network Coding (NetCod)*, 2016.

[9] ——, "Low Subpacketization Level Schemes for Coded Caching," in *IEEE Intl. Symposium on Info. Th.*, 2017.

[10] ——, "Low Subpacketization Schemes for Coded Caching," 2017 [Online] Available: https://arxiv.org/abs/1706.00101.

[11] ——, "Coded caching for networks with the resolvability property," in *IEEE Intl. Symposium on Info. Th.*, 2016.

[12] Z. Bar-Yossef, Y. Birk, T. Jayram, and T. Kol, "Index coding with side information," *IEEE Trans. on Info. Th.*, vol. 57, no. 3, pp. 1479–1494, March 2011.

[13] D. S. Lun, N. Ratnakar, M. Médard, R. Koetter, D. R. Karger, T. Ho, E. Ahmed, and F. Zhao, "Minimum-cost multicast over coded packet networks," *IEEE Trans. on Info. Th.*, vol. 52, no. 6, pp. 2608–2623, 2006.

[14] H. D. Sherali and G. Choi, "Recovery of primal solutions when using subgradient optimization methods to solve Lagrangian duals of linear programs," *Operations Research Letters*, vol. 19, no. 3, pp. 105–113, 1996.