On the Fundamental Limits of Coded Data Shuffling for Distributed Machine Learning

Adel Elmahdy and Soheil Mohajer

***Abstract*— We consider the data shuffling problem in a distributed learning system, in which a master node is connected to a set of worker nodes, via a shared link, in order to communicate a set of files to the worker nodes. The master node has access to a database of files. In every shuffling iteration, each worker node processes a new subset of files, and has excess storage to partially cache the remaining files, assuming the cached files are uncoded. The caches of the worker nodes are updated every iteration, and they should be designed to satisfy any possible unknown permutation of the files in subsequent iterations. For this problem, we characterize the exact load-memory trade-off for worst-case shuffling by deriving the minimum communication load for a given storage capacity per worker node. As a byproduct, the exact load-memory trade-off for any shuffling is characterized when the number of files is equal to the number of worker nodes. We propose a novel deterministic coded shuffling scheme, which improves the state of the art, by exploiting the cache memories to create coded functions that can be decoded by several worker nodes. Then, we prove the optimality of our proposed scheme by deriving a matching lower bound and showing that the placement phase of the proposed coded shuffling scheme is optimal over all shuffles.**

***Index Terms*— Data shuffling, coded caching, distributed computing, distributed machine learning.**

I. Introduction

With the emergence of big data analytics, distributed computing systems have attracted enormous attention in recent years. The computational paradigm in the era of big data has shifted towards distributed systems, as an alternative to expensive supercomputers. Distributed computing systems are networks that consist of a massive number of commodity computational nodes connected through fast communication links. Examples of distributed computing applications span distributed machine learning, massively multilayer online games (MMOGs), wireless sensor networks, real-time process control, etc. Prevalent distributed computing frameworks, such as Apache Spark [2], and computational primitives, such as MapReduce [3], Dryad [4], and CIEL [5], are key enablers to process substantially large data-sets (in the order of terabytes), and execute production-scale data-intensive tasks.

Manuscript received July 9, 2018; revised July 29, 2019; accepted November 12, 2019. Date of publication January 10, 2020; date of current version April 21, 2020. This work was supported in part by the National Science Foundation under Grant CCF-1749981. This article was presented in part at the 2018 IEEE International Symposium on Information Theory.

The authors are with the Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN 55455 USA (e-mail: adel@umn.edu; soheil@umn.edu).

Communicated by E. Abbe, Associate Editor for Machine Learning.

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIT.2020.2964547

Data Shuffling is one of the core components in distributed learning algorithms. Broadly speaking, the data shuffling stage is introduced to prepare data partitions with desirable properties for parallel processing in future stages. A prototypical iterative data processing procedure is outlined as follows: (i) randomly shuffle the training data-set, (ii) equally partition the data-set into non-overlapping batches, and assign each batch to a local worker[1], (iii) each local worker performs a local computational task to train a learning model, (iv) reshuffle the training data-set to provide each worker with a new batch of data points at each learning model and continue the model training. Data shuffling is known to enhance the learning model quality and lead to significant statistical gains in ubiquitous applications for machine learning and optimization. One prominent example is stochastic gradient descend (SGD) [7]–[14]. Recht and Ré [7] conjectured a non-commutative arithmetic-geometric mean inequality, and showed that the expected convergence rate of the random shuffling version of SGD is faster than that of the usual with-replacement version provided the inequality holds [2]. In recent years, it has been demonstrated that shuffling the data before running SGD results in superior convergence performance [8]–[14]. For instance, Meng *et al*. [13] have proposed an extensive analysis on the desirable convergence properties of distributed SGD with random shuffling, in both convex and non-convex cases. In practice, however, the benefits of data shuffling come at a price. In every shuffling iteration, the entire data-set is communicated over the network of workers. Consequently, this leads to performance bottlenecks due to the communication overhead.

The idea of incorporating coding theory into the context of distributed machine learning has been introduced in a recent work by [15]. The authors posed an intriguing question as to how to use coding techniques to ensure robust speedups in distributed computing. To address this question, the work flow of distributed computation is abstracted into three main phases; a storage phase, a communication phase, and a computation phase. Coding theory is utilized to alleviate the bottlenecks in the computation and communication phases of distributed learning algorithms. More specifically, the authors proposed novel algorithms for coded computation to speed up the

[1]One may consider storing the entire training data-set in a massive shared storage system and let the workers directly access the new batches every learning epoch. Although this setting eliminates the communication overhead of the shuffling mechanism, it suffers from network and disk I/O bottlenecks, and hence, this approach is notoriously sluggish and cost-inefficient as well [6].

[2]It is a long-standing problem in the theory of SGD to prove this statement, and the correctness of the full conjecture is still an open problem.

performance of linear operations, and coded data shuffling to overcome the significant communication bottlenecks between the master node and worker nodes during data shuffling.

A. Related Works

The data shuffling problem has been extensively studied from various perspectives under different frameworks. In what follows, we survey the literature and present the progress and the current status of the problem.

1) Data Shuffling in Master-Worker Distributed Computing Framework: In the master-worker distributed setup, the master node has access to the entire data-set that is randomly permuted and partitioned into batches at every iteration of the distributed algorithm. The data shuffling phase aims at communicating these batches to the worker nodes in order to locally perform their distributed tasks in parallel. Then, the master node aggregates the local results of the worker nodes to complete the computation and give the final result. Lee *et al*. [15] proposed the first coded shuffling algorithm, based on random data placement, that leverages the excess storage of the local caches of the worker nodes to slash the communication bottlenecks. The coded shuffling algorithm consists of three main strategies: a coded transmission strategy designed by the master node, and decoding and cache updating strategies executed by the worker nodes. It is demonstrated, through extensive numerical experiments, the significant improvement in the achievable communication load[3] and the average transmission time of coded shuffling framework, compared to uncoded shuffling. The theoretical guarantees of [15] hold only when the number of data points approaches infinity. Moreover, the broadcast channel between the master node and the worker nodes in [15] is assumed to be perfect. In pursuance of a practical shuffling algorithm, Chung *et al*. [6] have recently proposed a novel coded shuffling algorithm, coined "UberShuffle", to individually address the practical considerations of the shuffling algorithm of [15]. However, it is not evident how far these coded shuffling algorithms are from the fundamental limits of communication load.

Attia and Tandon [16]–[18] investigated the coded data shuffling problem in a distributed computing system, consisting of a master node that communicates data points (or coded functions of them) to worker nodes with limited storage capacity. An information-theoretic formulation of the data shuffling problem was proposed for data delivery and storage update phases. Furthermore, the worst-cast communication load is defined to be the maximum communication load from the master node to the worker nodes over all possible consecutive data shuffles for any achievable scheme characterized by the encoding, decoding, and cache updating functions. Accordingly, the authors characterized the optimal trade-off between the storage capacity per worker node and the worst-case communication load for certain cases of the number of files N, the number of worker nodes K, and the available storage per worker node S. More specifically, the communication load was characterized when the number of worker nodes is limited to $K \in \{2, 3\}$ in [17]. Furthermore, the special case of *no-excess storage* (arbitrary N and K, but $S = N/K$) was addressed in [18]. However, the proposed schemes in these works do not generalize for arbitrary parameters. Recently, the authors have proposed "aligned coded shuffling scheme" [16] that is optimal for $K < 5$, and suboptimal for $K \geq 5$ with maximum multiplicative gap of $(K - \frac{1}{3})/(K - 1)$ from the lower bound on the load for the worst-case communication scenario. The proposed placement strategy is similar to the one in coded caching literature [19], and the achievable scheme hinges on the concept of interference alignment [20]. On the other hand, the proposed information-theoretic lower bound is based on a similar bounding technique introduced in [21].

Under the same master-worker framework, Song *et al*. [22] considered the data shuffling problem from the perspective of index coding [23], where the new data assigned by the master node at every iteration constitute the messages requested by the worker nodes, and the data cached at the worker nodes form the side information. Motivated by the NP-hardness of the index coding problem [23], the authors proposed a pliable version of the index coding problem to enhance the communication efficiency for distributed data shuffling. It is assumed that the worker nodes are pliable in such a way that they are only required to obtain new messages, that are randomly selected from original set of messages, at every iteration. This degree of freedom enables the realization of semi-random data shuffling that yields more efficient coding and transmission schemes, as opposed to fully random data shuffling.

Recently, Wan *et al*. [24] have considered a decentralized communication paradigm for the data shuffling problem, where worker nodes only communicate data points among each other, and the master node does not participate in the data communication, except for the initial placement. The authors have proposed coded distributed data shuffling schemes that are within a factor of 2 from the optimal trade-off, under the constraint of uncoded cache placement. Moreover, the exact trade-off is characterized for $K \leq 4$, and $S \in \{1, K - 2, K - 1\}$ where $S = S/(N/K)$.

2) Data Shuffling in MapReduce Distributed Computing Framework: MapReduce [3] is a programming paradigm that allows for parallel processing of massive data-sets across large clusters of computational nodes. More concretely, the overall computation is decomposed into computing a set of "Map" and "Reduce" functions in a distributed and parallel fashion. Typically, a MapReduce job splits the input data-set into blocks, each of which is locally processed by a computing node that maps the input block into a set of intermediate key/value pairs. Next, the intermediate pairs are transferred to a set of processors that reduce the set of intermediate values by merging those with the same intermediate key. The process of inter-server communication between the mappers and reducers is referred to as *data shuffling*. Li *et al*. [25] introduced a variant implementation of MapReduce, named "Coded MapReduce" that exploits coding to considerably reduce the communication load of the data shuffling phase.

[3] In the literature, the communication load is referred to as "communication rate", e.g. [15], [16]. However, the more accurate term should be communication (or delivery) load which we use throughout the manuscript.

The key idea is to create coded multicast opportunities in the shuffling phase through an assignment strategy of repetitive mappings of the same input data block across different servers. The fundamental trade-off between computation load and communication cost in Coded MapReduce is characterized in [26]. A unified coding framework for distributed computing in the presence of straggling servers was proposed in [27], where the trade-off between the computation latency and communication load is formalized for linear computation tasks.

We would like to highlight the subtle distinction between the coded caching problem and the coded shuffling problem. Both problems share the property that the prefetching scheme is designed to minimize the communication load for any possible unknown demand (or permutation) of the data. However, the coded shuffling algorithm is run over a number of iterations to store the data batches and compute some task across all worker nodes. In addition to that, the permutations of the data in subsequent iterations are not revealed in advance. Therefore, the caches of the worker nodes should be adequately updated after every iteration to maintain the structure of the data placement, guarantee the coded transmission opportunity, and achieve the minimum communication load for any undisclosed permutation of the data. Another subtle distinction that we would like to emphasize is the difference between the concept of data shuffling in the master-worker setup and that in the MapReduce setup. In the master-worker setup, a master node randomly shuffles data points among the computational worker nodes for a number of iterations to enhance the statistical efficiency of distributed computing systems. A coded data shuffling algorithm enables coded transmission of batches of the data-set through exploiting the excess storage at the worker nodes. On the other hand, in the MapReduce setup, the whole data-set is divided among the computational nodes, and a data placement strategy is designed in order to create coding opportunities that can be utilized by the shuffling scheme to transfer locally computed results from the mappers to the reducers. In other words, coded MapReduce enables coded transmission of blocks of the data processed by the mappers in the shuffling phase through introducing redundancy in the computation of the Map stage.

Another distinction that should be highlighted is between index coding problem and coded shuffling problem. As discussed in [16] (Attia and Tandon), the data shuffling problem can also be considered as an index coding problem, in which the data cached at the worker nodes form the side information, and the new data assignments are the messages required by each worker node. It is worth noting that both the side information and requested files in index coding problem usually consist of one (of several) complete files, without sub-packetization, and this concern make the two problems different. Moreover, while the side information is given in index coding problem, here we can partially select the side information (the content of the excess storage) to reduce the communication load. Perhaps the most related work in the context of index coding to data shuffling is [22] (Song *et al.*), where a pliable version of index coding is adopted for data shuffling. In the pliable index coding it is assumed that the worker nodes are only required to obtain new messages (not previously cached in their storage). This degree of freedom enables the realization of semi-random data shuffling that yields more efficient coding and transmission schemes, as opposed to fully random data shuffling. It was demonstrated that a communication load of order $O(\log^2(K))$ can be achieved under this framework. However, it is worth mentioning that the constraints of the pliable index coding problem are very relaxed compared to the problem we study in this paper. Under the data shuffling model we study in this work, the worst case communication load is of order of $O(\frac{K-S}{S})$, where S is the normalized storage size of each worker. Therefore, the communication cost is still $O(K)$ when S is small, but decreases to $O(1)$ if $S = \Theta(K)$.

B. Main Contributions

In this paper, we consider a data shuffling problem in a master-worker distributed computing system, in which we have a master node and K worker nodes. The master node has access to the entire data-set of N files. Each worker node has a limited cache memory that can store up to S files. In each iteration of the distributed algorithm, the master node randomly shuffles the data points among the worker nodes. We summarize the main results of the paper as follows:

- We first study the data shuffling problem when $N = K$. We propose a novel linear coded shuffling algorithm that is based on interference alignment and elimination techniques. It comprises the following phases: (i) file partitioning and labeling, (ii) cache placement, (iii) encoding, (iv) decoding, (v) cache updating and subfile relabeling. We show how cache memories are leveraged in order to create coded functions that can be decoded by several worker nodes that process different files at every iteration of the distributed algorithm. The proposed scheme is generalized for arbitrary K and S.
- Next, we derive a matching information-theoretic lower bound on the communication load for the data shuffling problem when $N = K$, and we prove that among all possible placement and delivery strategies, our proposed coded shuffling scheme is universally optimal over all shuffling scenarios, and achieves the minimum communication load. Therefore, the optimal load-memory trade-off when $N = K$ is characterized for any shuffling.
- Finally, we extend the results obtained for the canonical setting of $N = K$ to investigate the general setting of the data shuffling problem when $N \geq K$. Inspired by the concept of perfect matching in bipartite graphs, we develop a coded shuffling scheme by decomposing the file transition graph into N/K subgraphs, each of which reduces to a canonical data shuffling problem with K files, K worker nodes, and storage capacity per worker node $S/(N/K)$. Hence, we can apply our coded shuffling scheme for $N = K$ to each sub-problem and obtain a delivery scheme for the original shuffling problem with arbitrary parameters N, K and S. This leads to an achievable scheme whose delivery load can provides us with an upper bound for the optimum communication load of the general coded data shuffling problem. Furthermore, we study an instance of the problem

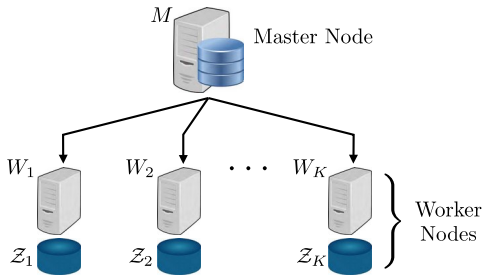


Fig. 1. Data shuffling in a distributed computing system.

(the worst-case scenario), and show that the upper bound obtained by the achievable scheme is indeed tight to the proposed instance of the problem. As a result, the optimal load-memory trade-off is exactly characterized when $N \geq K$ for the worst-case shuffling.

We emphasize on the fact that the proposed coded shuffling algorithms works for any shuffling model. Note that, while the proposed algorithm universally works for any shuffling, its required communication load is a function of the desired permutation. The worst-case shuffling is the one that requires the maximum communication load to permute the data points. In other words, the communication load of random shuffling does not exceed that of the worst-case shuffling. Hence, we use this load as a benchmark to evaluate the performance of the proposed algorithm.

A brief report of the main results of this paper has been given in [1]. This paper presents complete proofs of all results, along with numerous illustrative examples to explain the essential concepts.

C. Paper Outline

The remainder of the paper is organized as follows. We first present the formal definition of data shuffling problem as well as the main results of this work in Section II. The cache placement scheme is proposed in Section III. For the canonical setting of the shuffling problem, i.e. when $N = K$, two achievable coded shuffling schemes, along with illustrative examples, are delineated in Section IV. Then, the optimality proof for our proposed delivery scheme is presented in Section V. Next, for the general and practical setting of the shuffling problem, i.e. when $N \geq K$, is studied in Section VI, where the achievable delivery scheme, an illustrative example, and the optimality of proposed delivery scheme for the worst-case shuffling are presented. Finally, the paper is concluded and directions for future research are discussed in Section VIII.

II. Problem Formulation and Main Results

A. Formulation of Data Shuffling Problem

For an integer K, let $[K]$ denote the set of integers $\{1, 2, \ldots, K\}$. Fig. 1 depicts a distributed computing system with a master node, denoted by M, and a set of K worker nodes, denoted by $W = \{W_i : i \in [K]\}$. The master node is assumed to have access to a data-set, including N files, denoted by $F = \{F^j : j \in [N]\}$, where the size of each file is normalized to 1 unit. In practice, the number of files is remarkably larger than the number of worker nodes, and hence we study the data shuffling problem under the practical assumption of $N \geq K$. At each iteration, each worker node should perform a local computational task on a subset of N/K files[4]. The assignment of files to worker nodes is done by the master node, either randomly or according to some predefined mechanism. Each worker node W_i has a cache Z_i that can store up to S files, including those N/K under-processing files. This imposes the constraint $S \geq N/K$ on the size of the cache at each worker node. Once the computation at the worker nodes is done, the result is sent back to the master node. A new batch of N/K files will be assigned to each worker node for iteration $t + 1$, and the cache contents of the worker nodes should be accordingly modified. The communication of files from the master node to the worker nodes occurs over a shared link, i.e., any information sent by the master node will be received by all of the worker nodes. Similar to [15]–[18], the broadcast communication channel between the master node and worker nodes is assumed to be perfect.

For a given iteration t, we denote by $u(i)$ the set of indices of the files to be processed by W_i, and by P^i the portion of the cache of W_i dedicated to the *under-processing* files: $P^i = \{F^j : j \in u(i)\}$. The subsets $\{u(i) : i \in [K]\}$ provide a partitioning for the set of file indices, i.e., $u(i) \cap u(j) = \emptyset$ for $i = j$, and $\bigcup_{i=1}^{K} u(i) = [N]$. Similarly, $d(i)$ denotes the subset of indices of N/K files to be processed by W_i at iteration $t+1$, where $\{d(i) : i \in [K]\}$ also forms a partitioning for $[N]$. When $S > N/K$, each worker node has an excess storage to cache (parts of) the other files in F, in addition to the N/K files in P^i. We denote by $E_i = Z_i \setminus P^i$ the contents of the remaining space of the cache of W_i, which is called the *excess* storage. Therefore, $Z_i = P^i \cup E_i$. Let $P^i(t)$, $E_i(t)$ and $Z_i(t)$ denote the realizations of P^i, E_i and Z_i at iteration t, respectively. For the sake of brevity, we may drop the iteration index t whenever it is clear from the context.

Filling the excess part of the caches of worker nodes is performed *independently* of the new assigned subsets $\{d(i) : i \in [K]\}$. Between iterations t and $t + 1$, the master node should compute and broadcast a message (a function of all files in F), such that each worker node W_i can retrieve all files in $d(i)$ from its cached data Z_i and the broadcast message X. The communication load $R = R(K, N, S)$ is defined as the size of the broadcast message X for the parameters introduced above. We interchangeably refer to R as *delivery load* and *communication load* of the underlying data-shuffling system. The goal is to develop a cache placement strategy and design a broadcast message X to minimize R for any $\{d(i) : i \in [K]\}$. For $S \geq N$, we have $R = 0$ since each worker node can store all the files in its cache and no communication is needed between the master node and worker nodes for any shuffling. Thus, we can focus on the regime of $N/K \leq S \leq N$. We define $S = S/(N/K)$ to be the cache size normalized by the size of data to be processed by each worker node. Accordingly, we have $1 \leq S \leq K$.

B. File Transition Graph

A file transition graph is defined as a directed graph $G(V, E)$, where V, with $|V| = K$, denotes the set of vertices

[4]Unless otherwise stated, we assume that N/K and $S/(N/K)$ are integers.

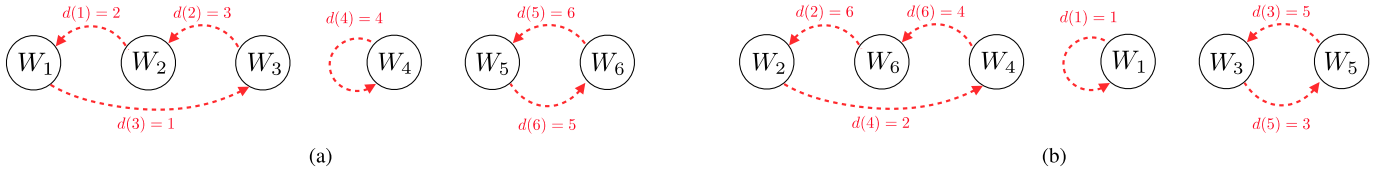


Fig. 2. The file transition graphs for two instances of a data shuffling system with $K = 6$ worker nodes and $N = 6$ files. Assume $u(i) = i$ for $i \in [6]$, and consider two different assignment functions; $d_1(\cdot)$ and $d_2(\cdot)$. The shown graphs are isomorphic. (a) $d_1(1) = 2, d_1(2) = 3, d_1(3) = 1, d_1(4) = 4, d_1(5) = 6$ and $d_1(6) = 5$. (b) $d_2(1) = 1, d_2(2) = 6, d_2(3) = 5, d_2(4) = 2, d_2(5) = 3$ and $d_2(6) = 4$.

each corresponding to a worker node, and E, with $|E| = N$, is the set of directed edges, each associated to one file (see Fig. 2). An edge $e_j = (i,) \in E$ with $j \in [N]$ and $i, \in [K]$ indicates that $j \in u(i) \cap d()$, i.e., file F^j is being processed by worker node W_i at iteration t, and assigned to worker node W to be processed at iteration $t + 1$. Note that in general $G(V, E)$ is a multigraph, since there might be multiple files in $u(i) \cap d()$, and we include one edge from W_i to W for each of such files.

Without loss of generality, let us assume a fixed assignment function $u(\cdot)$ at iteration t, for example, $u(i) = \{(i-1)N/K +$ $: \in [N/K]\}$ for $i \in [K]$, otherwise we can relabel the files. Hence, the problem and its file transition graph are fully determined by the assignment function $d(\cdot)$ at iteration $t + 1$. Let D_G be the set of assignment functions $d(\cdot)$ whose corresponding file transition graphs are *isomorphic* to G. Fig. 2 captures two instances of a shuffling problem with isomorphic file transition graphs. For a given graph $G(V, E)$, we define the average delivery load over all assignment functions in D_G as

$$R(G) = \frac{1}{|D_G|} \sum_{d \in D_G} R(N, K, S; d). \tag{1}$$

Our ultimate goal in this paper is to characterize $R(G)$ for given parameter (N, K, S) and for all feasible file transition graphs G.

C. Main Results

First, we present our main results to characterize the exact load-memory trade-off for the canonical setting of data shuffling problem, when $N = K$, for any shuffling. Since $N = K$, then $S = S$, each worker node processes one file at each iteration. Without loss of generality, we assume that W_i processes file F^i at every iteration, i.e., $u(i) = i$, for $i \in [K]$, otherwise we can relabel the files. The following theorems summarize our main results.

Theorem 1: For a data shuffling problem with a master node, K worker nodes, each with a cache of size S files with $S \in [N]$, the communication load $R = R(N = K, K, S)$ required to shuffle $N = K$ files among the worker nodes for any file transition graph is upper bounded by [5]

$$R \leq \frac{\binom{K-1}{S}}{\binom{K-1}{S-1}}. \tag{2}$$

For non-integer values of S, where $1 \leq S \leq N$, the lower convex envelope of the N corner points, characterized by (2), is achievable by memory-sharing.

[5]Note that $\binom{n}{k} = 0$ when $n < k$.

An achievability argument consists of a cache placement strategy and a delivery scheme. We propose a cache placement in Section III which will be used for all achievable schemes discussed in this paper. The delivery scheme, along with the memory-sharing argument for non-integer values of S, is presented in Section IV-A. Illustrative examples are then given in Section IV-B.

The next theorem provides an achievable delivery load (depending on the file transition graph) by an opportunistic coding scheme. We will show later that the underlying file transition graph of any data shuffling problem, $G(V, E)$, comprises a number of directed cycles. We denote the number of cycles in the file transition graph by γ, and denote the cycle lengths by $(_1, _2, \cdots, _\gamma)$ where $\sum_{i=1}^{\gamma} _i = K$.

Theorem 2: For a data shuffling system with a master node and K worker nodes, each with a cache of size S files, for $S \in [N]$, the shuffling of $N = K$ files among the worker nodes for a given file transition graph that comprises γ cycles can be performed by broadcasting a message of size R, where

$$R \leq \frac{\binom{K-1}{S} - \binom{\gamma-1}{S}}{\binom{K-1}{S-1}}. \tag{3}$$

For non-integer values of S, where $1 \leq S \leq N$, the lower convex envelope of the N corner points, characterized by (3), is achievable by memory-sharing.

The proposed delivery scheme and achievability proof for Theorem 2 are presented in Section IV-C. The memory-sharing argument for non-integer values of S follows a similar reasoning as the one in Theorem 1. We provide an illustrative example in Section IV-D.

Theorem 3: For the data shuffling system introduced in Theorem 2, the communication load R required to shuffle $N = K$ files among the worker nodes for a given assignment with a file transition graph that comprises γ cycles is lower bounded by

$$R \geq \frac{\binom{K-1}{S} - \binom{\gamma-1}{S}}{\binom{K-1}{S-1}}. \tag{4}$$

The proof of optimality (converse) is presented in Section V, where we also provide an illustrative example to describe the proof technique.

Corollary 1: Theorems 2 and 3 prove the optimality of the proposed coded shuffling scheme for an arbitrary number of worker nodes K, storage capacity per worker node S, and file transition graph with γ cycles, when $N = K$. Therefore,

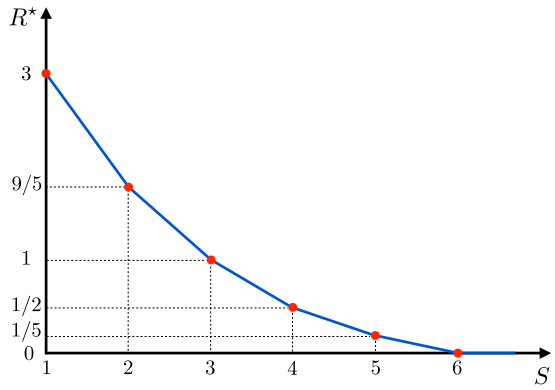


Fig. 3. The optimum trade-off curve between the delivery load R and the storage capacity per worker node S, when $N = K = 6$ and $\gamma = 3$.

the optimal delivery load R is characterized as

$$R\ (N = K, K, S) = \frac{\binom{K-1}{S} - \binom{\gamma-1}{S}}{\binom{K-1}{S-1}}, \quad S \in [N]. \tag{5}$$

For non-integer values of S, where $1 \le S \le N$, the optimal delivery load R is equal to the lower convex envelope of the N corner points given in (5). Furthermore, when $\gamma - 1 < S$, the achievable delivery load of Theorem 2 is equal to that of Theorem 1, and takes its maximum. This characterizes the optimal worst-case delivery load $R_{\text{worst-case}}$ which is given by

$$R_{\text{worst-case}} = \frac{\binom{K-1}{S}}{\binom{K-1}{S-1}}. \tag{6}$$

This indicates that the upper bound of Theorem 1 is the best universal (assignment independent) bound that holds for all instances of the data shuffling problem.

Remark 1: The essence of the information-theoretic lower bound on the communication load for the worst-case shuffling proposed in this paper for $N = K$ is equivalent to that of [16] (Attia and Tandon) for general N and K. However, in more details, the proof of [16] involves solving a linear program, while we use set theoretic arguments to obtain a closed-form expression. Moreover, for general parameters N and K, we provide a simple reduction argument to re-use the bound proved for the canonical setting.

Fig. 3 captures the optimum trade-off curve between $R\ (K, K, S)$ as a function of S for $N = K = 6$ and a file transition graph with $\gamma = 3$ cycles.

Next, based on the results obtained for the canonical setting of data shuffling problem when $N = K$, we present our main results in Theorem 4 to characterize an upper bound on the load-memory trade-off for the general setting of data shuffling problem when $N \ge K$. This upper bound turns out to be optimum for the worst-case shuffling, as stated in Theorem 5.

Theorem 4: For a data shuffling system that processes N files, and consists of a master node and K worker nodes, each with a normalized storage capacity of $S = S/(N/K)$ files, the achievable delivery load $R = R(N, K, S)$ required to shuffle N files among the worker nodes for any file transition graph is upper bounded by

$$R \le \frac{N}{K} \frac{\binom{K-1}{S}}{\binom{K-1}{S-1}}, \quad S \in [K]. \tag{7}$$

For non-integer values of S, where $1 \le S \le K$, the lower convex envelope of the N corner points, characterized by (7) is achievable by memory-sharing.

The delivery scheme and achievability proof are presented in Section VI-A. Note that the proposed achievable scheme is an extension to the one developed for the canonical setting of $N = K$ in Theorem 2. The memory-sharing argument for non-integer values of S follows a similar reasoning as the one in Theorem 1. We also present an illustrative example in Section VI-C.

Theorem 5: For the data shuffling system introduced in Theorem 4, the communication load $R_{\text{worst-case}}$ required to shuffle N files among the worker nodes according to the worst-case shuffling is given by

$$R_{\text{worst-case}} = \frac{N}{K} \frac{\binom{K-1}{S}}{\binom{K-1}{S-1}}. \tag{8}$$

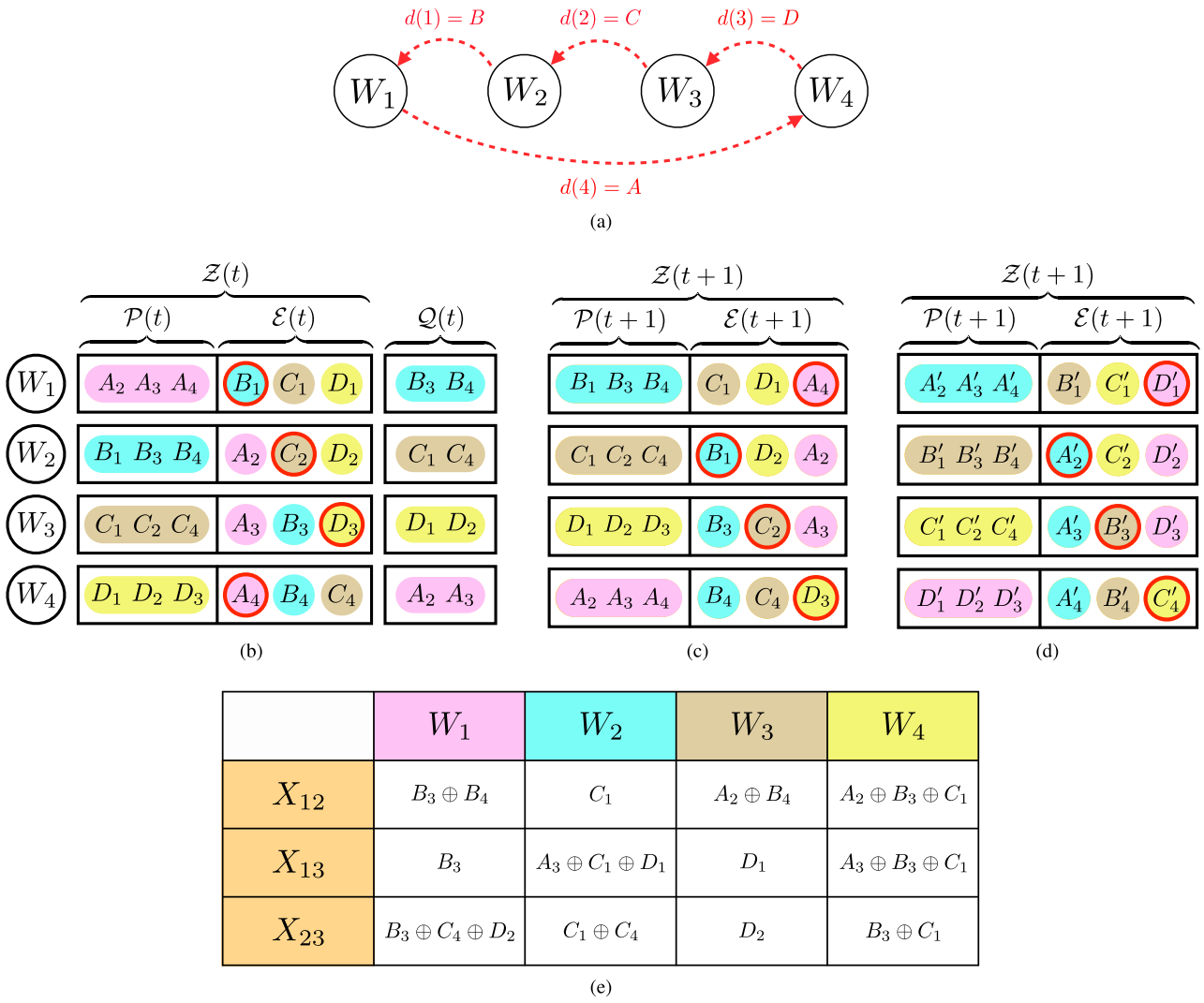
The proof of Theorem 5 is presented in Section VI-D.

Before we start discussing the results of the paper, we present an example to explain the general idea of the proposed coded shuffling scheme.

D. Illustrative Example

Example 1 (Single-Cycle File Transition Graph): Consider a shuffling system with a master node and $K = 4$ worker nodes. The size of the cache at each worker node is $S = 2$ files. There are $N = 4$ files, denoted by $\{F^1, F^2, F^3, F^4\}$. For notational simplicity, we rename the files as $\{A, B, C, D\}$. Without loss of generality, we assume that worker nodes W_1, W_2, W_3 and W_4 are processing files A, B, C, and D, respectively, at iteration t, that is $u(1) = A$, $u(2) = B$, $u(3) = C$, and $u(4) = D$. The file transition graph is $d(1) = B$, $d(2) = C$, $d(3) = D$ and $d(4) = A$, as depicted by Fig. 4a. The proposed placement strategy partitions each file into $\binom{K-1}{S-1} = 3$ subfiles of equal sizes. The subfiles are labeled with sets $\Gamma \subseteq [4]$, where $|\Gamma| = S - 1 = S/(N/K) - 1 = 1$. For instance, file A being processed by worker node W_1 is partitioned into A_2, A_3, and A_4. Accordingly, the cache Z_i of W_i is divided into two parts; P^i that is dedicated to the under-processing file F^i, and E_i that is dedicated to store parts of other files. Fig 4b captures the cache organization of worker nodes, along with the missing subfiles (i.e., the ones in Q_i, as defined in (14)) that need to be processed at iteration $t + 1$. The broadcast message X transmitted from the master node to the worker nodes is formed by the concatenation of 3 sub-messages $X = (X_{12}, X_{13}, X_{23})$, where

$$\begin{aligned} X_{12} &= A_2 \oplus B_3 \oplus B_4 \oplus C_1, \\ X_{13} &= A_3 \oplus B_3 \oplus C_1 \oplus D_1, \\ X_{23} &= B_3 \oplus C_1 \oplus C_4 \oplus D_2. \end{aligned} \tag{9}$$



	W_1	W_2	W_3	W_4
X_{12}	$B_3 \oplus B_4$	C_1	$A_2 \oplus B_4$	$A_2 \oplus B_3 \oplus C_1$
X_{13}	B_3	$A_3 \oplus C_1 \oplus D_1$	D_1	$A_3 \oplus B_3 \oplus C_1$
X_{23}	$B_3 \oplus C_4 \oplus D_2$	$C_1 \oplus C_4$	D_2	$B_3 \oplus C_1$

(e)

Fig. 4. Data Shuffling system with $N = K = 4$, $S = 2$ and $\gamma = 1$. (a) The file transition graph for a data shuffling system with $N = K = 4$, $S = 2$ and $\gamma = 1$. Worker nodes W_1, W_2, W_3 and W_4 are processing files A, B, C, and D, respectively. (b) Cache organization of worker nodes at iteration t, along with the set of subfiles which are not available in the caches at iteration t and need to be processed at iteration $t + 1$. (c) Cache organization of worker nodes at iteration $t + 1$ after updating the caches. Subfiles A_4, B_1, C_2 and D_3 in $E_4(t)$, $E_1(t)$, $E_2(t)$ and $E_3(t)$ are moved to $E_1(t + 1)$, $E_2(t + 1)$, $E_3(t + 1)$ and $E_4(t + 1)$, respectively. (d) Cache organization of worker nodes at iteration $t + 1$ after updating the caches and relabeling the subfiles of Fig. 4b. Subfiles A_4, B_1, C_2 and D_3 in $E_4(t)$, $E_1(t)$, $E_2(t)$ and $E_3(t)$ are moved to $E_1(t + 1)$, $E_2(t + 1)$, $E_3(t + 1)$ and $E_4(t + 1)$ and relabeled to D_1, A_2, B_3 and C_4, respectively. (e) Received functions by worker nodes after removing the cached subfiles. The complete received functions at worker nodes are expressed in (9).

We need to show that each worker node W_i can decode all the missing subfiles in Q_i from the broadcast message. Fig. 4e shows the received sub-messages by each worker node after removing the subfiles that exist in its cache. For instance, W_1 can decode B_3 from X_{13} and B_4 from $X_{13} \oplus X_{12}$, respectively. These, together with B_1 that is already cached in Z_1, enable W_1 to fully recover B, which is the file to be processed at iteration $t + 1$.

Decoding file A at W_4, which is the ignored worker node, is more involved. Worker node W_4 can decode A_2 and A_3 from $X_{12} \oplus X_{23}$ and $X_{13} \oplus X_{23}$, respectively, and A_4 already exists in its cache Z_4. Consequently, the proposed scheme can achieve a delivery load of $R_{\text{coded}} = 1$, due to sending 3 sub-messages, each of size 1/3. On the other hand, the delivery load achieved by the uncoded shuffling scheme, under the same placement strategy, involves sending 8 sub-messages, each of size 1/3, resulting in $R_{\text{uncoded}} = 8/3$. Hence, the proposed coded shuffling scheme can save around 62% of the communication load, compared to the uncoded shuffling scheme.

After the decoding phase, each worker node has access to (at least) 8 subfiles, including 6 subfiles pre-stored in its cache Z_i, and 2 subfiles in Q_i. However, only 6 subfiles can be stored in the cache, and the remaining ones should be discarded. It remains to show that the caches of the worker nodes can be updated using the broadcast message X to maintain a similar arrangement in preparation for the next data shuffle from iteration t+1 to t+2. This is done in two phases, namely cache updating and subfile relabeling. Fig. 4c depicts the cache organization of the worker nodes after updating the caches, while Fig. 4d captures the cache organization of the worker nodes after updating the caches and relabeling the subfiles. For example, as shown in Fig. 4c, W_1 needs to keep a full copy of $B = \{B_1, B_2, B_3\}$ at iteration $t + 1$ because $d(1) = B$.

Moreover, subfiles C_1 and D_1 already exist in $E_1(t)$, and hence they will remain in $E_1(t+1)$. Among the remaining subfiles $\{A_2, A_3, A_4\}$, only one of them can be kept in the cache. According to (20), the subfile to be kept is A_Γ such that $d^{-1}(1) \in \Gamma$, that is A_4. Finally, in order to maintain a cache configuration consistent with the original one in (11), all the subfiles in the excess storage of W_i should have a subscript of $\{i\}$, and W_i should process F_i, for $i \in [K]$, at every iteration. For example, subfiles B_1, B_3 and B_4, that are processed in $P^1(t+1)$ by W_1, in Fig. 4c are relabeled to A_2, A_3 and A_4 in Fig. 4d, respectively. Similarly, subfiles C_1, D_1, A_4, that are cached in $E_1(t+1)$ by W_1, in Fig. 4c are relabeled to B_1, C_1 and D_1 in Fig. 4d, respectively.

III. Cache Placement

In this section we introduce our proposed cache placement, in which the contents of each worker node's cache at iteration t are known. Note that the cache placement does not depend on the files to be processed by the worker nodes at iteration $t+1$, i.e., it does not depend on $\{d(i) : i \in [K]\}$.

A. File Partitioning and Labeling

Throughout this work, we assume that N/K and $S/(N/K)$ are integer numbers, unless it is specified otherwise. Let $S = S/(N/K)$. Let F^j be a file being processed by worker node W_i at iteration t, i.e., $j \in u(i)$. We partition F^j into $\binom{K-1}{S-1}$ equal-size subfiles, and label the subfiles with a subscript as

$$F^j = \{F^j_\Gamma : \Gamma \subseteq [K] \setminus \{i\}, |\Gamma| = S-1\}, \quad \forall i \in [K],\ j \in [N],\ j \in u(i). \tag{10}$$

Since the size of each file is normalized to 1, the size of each subfile will be $1/\binom{K-1}{S-1}$. For the sake of completeness, we also define dummy subfiles $F^j_\Gamma = 0$ (with size 0) for every $\Gamma \subseteq [K]$ with $j \in \Gamma$ or $|\Gamma| = S-1$.

B. Cache Placement

The cache Z_i of W_i consists of two parts: (i) the *under-processing* part P^i, in which *all subfiles* of files to be processed at iteration t are stored; (ii) the *excess* storage part E_i, which is equally distributed among all other files. We denote by E_i the portion of E_i dedicated to the file F, in which all subfiles F_Γ with $i \in \Gamma$ are cached. Hence, we have

$$Z_i = P^i \cup E_i = P^i \cup \bigcup_{\in [N]\setminus u(i)} E_i, \quad \forall i \in [K], \tag{11}$$

where

$$P^i = \{F^j_\Gamma : j \in u(i),\ \Gamma \subseteq [K] \setminus \{i\},\ |\Gamma| = S-1\}, \tag{12}$$

$$E_i = \{F_\Gamma : \in u(i),\ i \in \Gamma \subseteq [K],\ |\Gamma| = S-1\}. \tag{13}$$

For any worker node W_i, there are N/K complete files in P^i. Moreover, for each of the remaining $N - N/K$ files, there are $\binom{K-2}{S-2}$ subfiles, out of a total of $\binom{K-1}{S-1}$ subfiles, that are cached in the excess storage part. Thus, we have

$$|Z_i| = |P^i| + \left|\bigcup_{\in [N]\setminus u(i)} E_i\right| = \frac{N}{K} + \left(N - \frac{N}{K}\right)\frac{\binom{K-2}{S-2}}{\binom{K-1}{S-1}} = S,$$

which satisfies the memory constraints.

Remark 2: The proposed cache placement is different from the one in [14] (Lee et al.). The placement in [15] follows a random sampling of the files independently across the users. Moreover, file splitting is not allowed in [15], and a file (data point) is either fully stored in the storage of a worker node, or no bit of that is cached by the worker node. The proposed placement strategy, however, is deterministic and fully characterized at every shuffling iteration. On the other hand, the so-called structural invariant placement strategy introduced in [16] is deterministic, but different from the one proposed in this paper. In particular, the structural invariant placement method works for values of S satisfying $S = 1 + i\frac{K-1}{K}$ for some integer $i = 0, 1, \ldots, K$, while we need S to be some integer in $[K]$. Consequently, the communication load achieved in [16] is sub-optimum for general K. A variation of the structural invariant storage placement is proposed in [16, Appendix D] only for $S \in \{1, K-2, K-1\}$ which is identical the placement strategy proposed in our work for any integer value of S. It is worth noting that in spite of similarities between the placement strategies, the proposed encoding and decoding methods in [16] (referred to as "aligned coded shuffling") are completely different from those proposed in this paper.

Recall that the worker node W_i should be able to recover files $\{F : \in d(i)\}$ from its cache Z_i and the broadcast message X. Communicating files in $d(i)$ from the master node to a worker node W_i can be limited to sending only the desired subfiles that do not exist in the cache of W_i. For a worker node $i \in [K]$, let Q_i denote the set of subfiles to be processed by W_i at iteration $t+1$, which are not available in its cache Z_i at iteration t, that is,

$$Q_i = \{F_\Gamma : \in d(i),\ \notin u(i),\ i \notin \Gamma, \Gamma \subseteq [K],\ |\Gamma| = S-1\}. \tag{14}$$

It is evident that each worker node needs to decode at most $\binom{K-1}{S-1} - \binom{K-2}{S-2} = \binom{K-2}{S-1}$ subfiles for each of the N/K files in $d(i)$, in order to process them at iteration $t+1$.

The pseudocodes of the proposed file partitioning and labeling, and cache placement are given in Algorithm 1 and Algorithm 2, respectively, in Appendix H.

IV. Coded Shuffling for the Canonical Setting ($N = K$)

We describe two delivery strategies in this section. The first delivery scheme is universal, in the sense that it does not exploit the properties of the underlying file transition graph. By analyzing this scheme in Section IV-A, we show that the delivery load in Theorem 1 is achievable. Two illustrative examples are presented in Section IV-B to better describe the coding and decoding strategies. We then demonstrate that the size of the broadcast message can be reduced by exploiting the cycles in the file transition graph. A graph-based delivery strategy is proposed in Section IV-C. This new scheme can achieve the reduced delivery load proposed in Theorem 2. Finally, we conclude this section by presenting an illustrative example for the graph-based delivery scheme in Section IV-D.

A. A Universal Delivery Scheme for Any Shuffling: Proof of Theorem 1

Recall that for $N = K$ we have $S = S/(N/K) = S$. In order to prove Theorem 1, we propose a coded shuffling scheme to show that a delivery load of $R = \binom{K-1}{S} / \binom{K-1}{S-1}$ is achievable for the canonical setting ($N = K$) for any integer $1 \le S \le N$. We assume, without loss of generality, that W_i processes file F^i at iteration t, i.e., $u(i) = i$ for $i \in [K]$, otherwise we can relabel the files.

Encoding

Given all cache contents $\{Z_i : i \in [K]\}$, characterized by (11), and $\{d(i) : i \in [K]\}$, the broadcast message X sent from the master node to the worker nodes is obtained by the concatenation of a number of sub-messages X_Δ, each specified for a group of worker nodes Δ, that is,

$$X = \{X_\Delta : \Delta \subseteq [K-1], |\Delta| = S\}, \tag{15}$$

where

$$X_\Delta \left(\bigoplus_{i \in \Delta} F^i_{\Delta\setminus\{i\}} \oplus F^{d(i)}_{\Delta\setminus\{d(i)\}} \oplus \bigoplus_{j \in [K]\setminus\Delta} F^{d(i)}_{(\{j\}\cup\Delta)\setminus\{i,d(i)\}} \right). \tag{16}$$

The encoding design hinges on $K-1$ worker nodes. Without loss of generality, we consider $W_1, W_2, \ldots, W_{K-1}$ for whom the broadcast sub-messages are designed, and designate W_K as the *ignored* worker node. We will later show how W_K is served for free using the sub-messages designed for other worker nodes.

Remark 3: It is true that subfile $F^{d(i)}_{\Delta\setminus\{d(i)\}}$ does exist in the cache of worker node W_i, and is not needed to be broadcast, and indeed the encoded sub-message does not include this subfile. This is due to the fact that such a subfile $F^{d(i)}_{\Delta\setminus\{d(i)\}}$ appears twice in X_Δ, and hence will be canceled by the XOR operation. Note that, this subfile is not dummy only if $d(i) \in \Delta$. Therefore, one copy is attained in the second term of the summand as $F^{d(i)}_{\Delta\setminus\{d(i)\}}$, and the second copy is attained as the first term in the summand as $F^j_{\Delta\setminus\{j\}}$ for $j = d(i) \in \Delta$.

According to the proposed encoding scheme, there are a total of $\binom{K-1}{S}$ encoded sub-messages, each corresponds to one subset Δ, and the size of each sub-message is $1/\binom{K-1}{S-1}$. Hence, the overall broadcast communication load is upper bounded by

$$R \le \frac{\binom{K-1}{S}}{\binom{K-1}{S-1}},$$

as claimed in Theorem 1.

Decoding

The following lemmas demonstrate how each worker node decodes the missing subfiles, that constitute the file to be processed at iteration $t+1$, from the broadcast sub-messages and its cache contents.

Lemma 1: For a worker node W, where $\in [K-1]$, a missing subfile $F^{d()}_\Gamma \in Q$ can be decoded

- from Z and the broadcast sub-message $X_{\{\}\cup\Gamma}$, if $K \notin \Gamma$; and
- from Z, the broadcast sub-message $X_{(\Gamma\setminus\{K\})\cup\{,d()\}}$, and other subfiles previously decoded by W, if $K \in \Gamma$.

We refer to Appendix A for the proof of lemma 1.

Remark 4: Here, we provide an intuitive justification for Lemma 1. Consider a worker node W_i and a set of worker nodes Δ of size S that includes i. One can show that every subfile appearing in X_Δ belongs to either Z_i or Q_i. Therefore, worker node W_i can recover a linear equation in the subfiles in Q_i by removing the subfiles in its cache Z_i from X_Δ. It turns out that all such equations are linearly independent. The number of such equations is $\binom{K-2}{S-1}$ (because $\Delta \subseteq [K-1]$ and $i \in \Delta$). On the other hand, the number of subfiles in Q_i is (at most) $\binom{K-1}{S-1} - \binom{K-2}{S-2} = \binom{K-2}{S-1}$, since out of a total of $\binom{K-1}{S-1}$ subfiles of $F^{d(i)}$, $\binom{K-2}{S-2}$ of them are cached in $E_i^{d(i)}$, characterized by (13). Therefore, the obtained set of linearly independent equations suffices to recover all the subfiles in Q_i.

Lemma 2: For the worker node W_K, any missing subfile $F^{d(K)}_\Gamma \in Q_K$ can be decoded from the cache contents Z_K and the summation of the broadcast sub-messages $X_{\{\}\cup\Gamma}$.
$\in [K-1]\setminus\Gamma$

We refer to Appendix B for the proof of lemma 2.

Cache Updating and Subfile Relabeling

After worker nodes decode the missing subfiles, characterized by (14), the caches of worker nodes need to be updated and the subfiles need to be relabeled before processing the files at iteration $t+1$. The goal of cache updating and subfile relabeling is to maintain a similar cache configuration for the worker nodes for shuffling iteration $t+2$. First, the caches are updated as follows:

- For $i \in [K]$, all the subfiles of $F^{d(i)}$ are placed in P^i at iteration $t+1$, i.e.,

$$P^i(t+1) = \{F^{d(i)}_\Gamma : \Gamma \subseteq [K]\setminus\{d(i)\}, |\Gamma| = S-1\}. \tag{17}$$

- For $i \in [K]$, the excess storage is updated by removing all the subfiles of $F^{d(i)}$, and replacing them by the subfiles of F^i that were cached at $W_{d^{-1}(i)}$, i.e.,

$$E_i(t+1) = (E_i(t) \setminus S_i) \cup A_i, \tag{18}$$

where

$$S_i = \{F^{d(i)}_\Gamma : i \in \Gamma,\ \Gamma \subseteq [K]\setminus\{d(i)\}, |\Gamma| = S-1\}, \tag{19}$$

$$A_i = \{F^i_\Gamma : d^{-1}(i) \in \Gamma,\ \Gamma \subseteq [K]\setminus\{i\}, |\Gamma| = S-1\}. \tag{20}$$

Consequently, we have $Z_i(t+1) = P^i(t+1) \cup E_i(t+1)$ by definition. Note that the cache updating procedure is feasible, since the subfiles needed for $Z_i(t+1)$ either exist in $Z_i(t)$ or appear in the set of missing subfiles $Q_i(t)$ to be decoded after the broadcast message delivery. In particular, all the subfiles of F^i already exist in $P^i(t)$, and hence those in A_i will be simply moved from the under-processing part to the excess storage part of the cache.

Next, the subfiles are *relabeled* as follows:

1) For every subfile F^i_Γ, where $i \in [K]$, $d^{-1}(i) \in \Gamma$, $\Gamma \subseteq [K] \setminus \{i\}$ and $|\Gamma| = S - 1$, relabel the subfile's subscript to F^i_Λ, where $\Lambda = \Gamma \setminus d^{-1}(i) \cup \{i\}$.
2) For every subfile F^i_Γ, where $i \in [K]$, $\Gamma \subseteq [K] \setminus \{i\}$ and $|\Gamma| = S - 1$, relabel the subfile's superscript to $F^{d^{-1}(i)}_\Gamma$.

At the end of cache updating and subfile relabeling phase, the cache configuration of each worker node at iteration $t+1$ maintains a similar arrangement to that introduced initially at iteration t and characterized by (11). More specifically, the cache updating step ensures that the under-processing part of the cache includes all subfiles of the file to be processed at iteration $t+1$. It also guarantees that the excess storage part of the cache stores an equal share of subfiles of all other files that are not in the under-processing part. The subfile relabeling step, however, ensures that two properties are satisfied at any shuffling iteration; (i) the index of a worker node appears in the new subscript of all subfiles in the excess storage part of its cache, (ii) the name of the file to be processed by worker node W_i is changed to F^i (from $F^{d(i)}$). Therefore, the proposed scheme can be systematically applied at the following shuffling iterations. This completes the proof of Theorem 1.

The pseudocodes of the proposed encoding at the master node, decoding at the worker nodes, and cache updating and subfile relabeling are given in Algorithm 3, Algorithm 4, and Algorithm 5, respectively, in Appendix H.

Remark 5: Let S be a non-integer cache size with $1 \le S \le N$. We can always write $S = \alpha S + (1-\alpha) S$, for some $\alpha \in (0, 1)$. The data shuffling problem for non-integer cache size S can be addressed by a memory-sharing argument, similar to [17]. More precisely, we can show that the pairs $(S, R(S))$ and $(S, R(S))$ are achievable, and conclude that, for $S = \alpha S + (1-\alpha) S$, a communication load of $R(S) = \alpha R(S) + (a-\alpha) R(S)$ can be achieved. Recall that the size of each file is normalized to 1 unit. For the memory-sharing argument, each file will be partitioned into two parts of sizes α and $1-\alpha$. The cache of each worker node is also divided into two parts of sizes αS and $(1-\alpha)S$. Then, the files of size α will be cached and shuffled within the parts of the caches of size αS. Similarly, the files of size $1-\alpha$, together with the parts of the caches of size $(1-\alpha)S$, form another isolated instance of the problem. Summing the delivery loads of the two instances, we get

$$\begin{aligned} &R(\alpha S + (1-\alpha) S) \\ &\quad = \alpha R(S) + (1-\alpha) R(S) \\ &\quad \le \alpha \frac{\binom{K-1}{S} - \binom{\gamma-1}{S}}{\binom{K-1}{S-1}} + (1-\alpha) \frac{\binom{K-1}{S} - \binom{\gamma-1}{S}}{\binom{K-1}{S-1}}. \end{aligned} \tag{21}$$

This shows that the convex hull of the pairs $\{(S, R) : S \in [N]\}$ is achievable.

B. Illustrative Examples

Example 2 (Multiple-Cycle File Transition Graph): Consider a shuffling problem with parameters $K = 6$, $S = 3$, and $N = 6$. For simplicity, we rename the files $\{F^1, F^2, F^3, F^4, F^5, F^5\}$ to $\{A, B, C, D, E, F\}$. Assume worker nodes W_1, W_2, W_3, W_4, W_5 and W_6 are processing files A, B, C, D, E, and F, respectively. The file transition graph is depicted by Fig. 5. It comprises $\gamma = 3$ cycles, with cycle lengths $(\ell_1, \ell_2, \ell_3) = (3, 1, 2)$. That is, $d(1) = B$, $d(2) = C$, $d(3) = A$ in the first cycle, $d(4) = D$ in the second cycle, and $d(5) = F$, $d(6) = E$ in the third cycle. Fig. 6a captures the cache organization of worker nodes, along with the missing subfiles that need to be processed at the next iteration. Note that P^i and E_i, for $i \in [K]$, are designed according to (12) and (13), respectively. We use (15) and (16) to design the broadcast message X, which is constructed by concatenating a number of X_Δ, each intended for $S = 3$ worker nodes. For example, X_{123} is expressed as

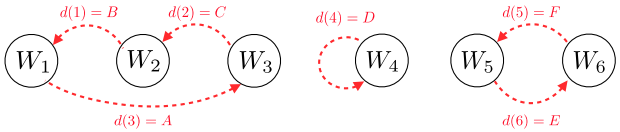


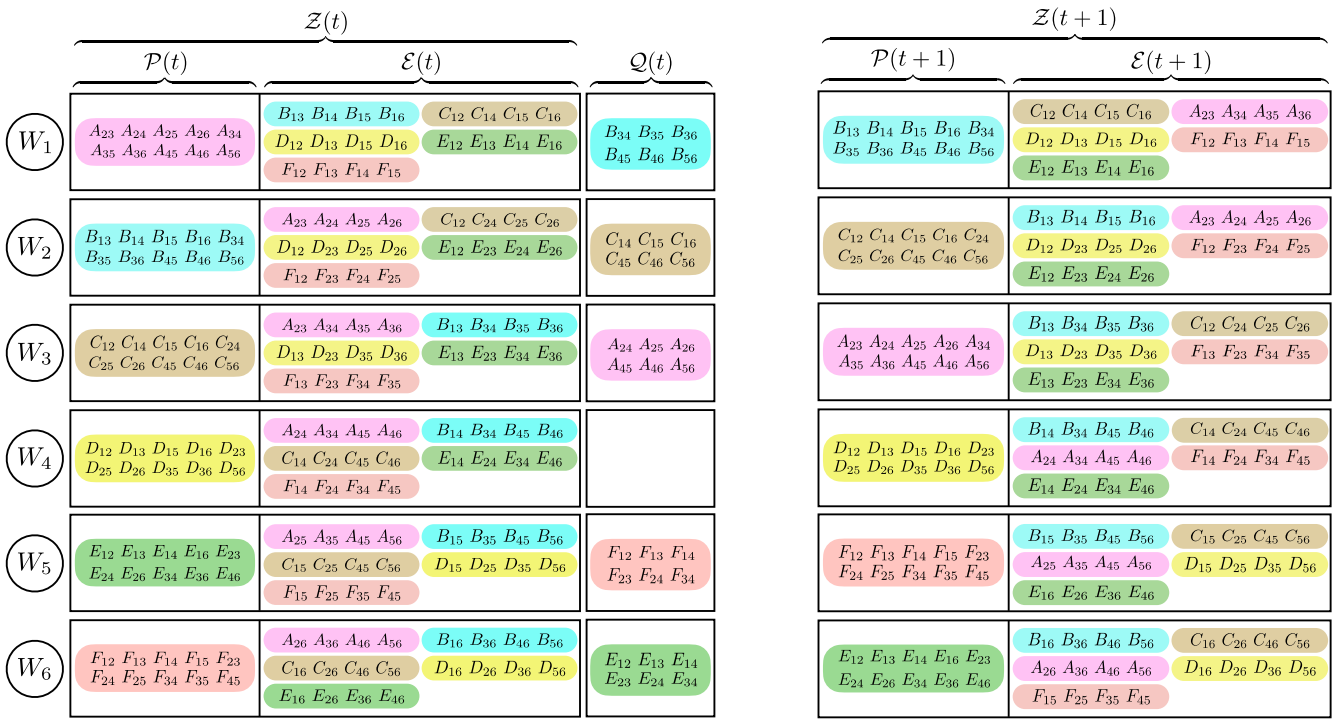
Fig. 5. The file transition graph for a data shuffling system with $N = K = 6$ and $\gamma = 3$. Worker nodes W_1, W_2, W_3, W_4, W_5 and W_6 process files A, B, C, D, E, and F at iteration t, respectively.

$$\begin{aligned} X_{123} = &\Big(\bigoplus_{i \in \{1,2,3\}} F^i_{\{1,2,3\}\setminus\{i\}} \oplus F^{d(i)}_{\{1,2,3\}\setminus\{d(i)\}} \oplus \\ &\qquad \bigoplus_{j \in [K]\setminus\{1,2,3\}} F^{d(i)}_{(\{j\}\cup\{1,2,3\})\setminus\{i,d(i)\}} \Big) \\ = &\; F^1_{23} \oplus F^2_{13} \oplus F^2_{34} \oplus F^2_{35} \oplus F^2_{36} \oplus F^3_{13} \oplus F^3_{12} \\ &\oplus F^3_{14} \oplus F^3_{15} \oplus F^3_{16} \oplus F^3_{12} \oplus F^1_{23} \oplus F^1_{24} \oplus F^1_{25} \oplus F^1_{26} \\ = &\; F^1_{24} \oplus F^1_{25} \oplus F^1_{26} \oplus F^2_{34} \oplus F^2_{35} \oplus F^2_{36} \oplus F^3_{14} \oplus F^3_{15} \oplus F^3_{16} \\ = &\; A_{24} \oplus A_{25} \oplus A_{26} \oplus B_{34} \oplus B_{35} \oplus B_{36} \oplus C_{14} \oplus C_{15} \oplus C_{16}. \end{aligned} \tag{22}$$

Similarly, the set of other broadcast sub-messages is expressed as

$$\begin{aligned} X_{124} &= A_{24} \oplus B_{34} \oplus B_{45} \oplus B_{46} \oplus C_{14}, \\ X_{125} &= A_{25} \oplus B_{35} \oplus B_{45} \oplus B_{56} \oplus C_{15} \oplus E_{12} \oplus F_{12}, \\ X_{134} &= A_{24} \oplus A_{45} \oplus A_{46} \oplus B_{34} \oplus C_{14}, \\ X_{135} &= A_{25} \oplus A_{45} \oplus A_{56} \oplus B_{35} \oplus C_{15} \oplus E_{13} \oplus F_{13}, \\ X_{145} &= A_{45} \oplus B_{45} \oplus E_{14} \oplus F_{14}, \\ X_{234} &= A_{24} \oplus B_{34} \oplus C_{14} \oplus C_{45} \oplus C_{46}, \\ X_{235} &= A_{25} \oplus B_{35} \oplus C_{15} \oplus C_{45} \oplus C_{56} \oplus E_{23} \oplus F_{23}, \\ X_{245} &= B_{45} \oplus C_{45} \oplus E_{24} \oplus F_{24}, \\ X_{345} &= A_{45} \oplus C_{45} \oplus E_{34} \oplus F_{34}. \end{aligned} \tag{23}$$

It should be noted that no subfiles of file D appears in any of the sub-messages defined (23) since $u(4) = d(4) = D$, i.e., D is processed by W_4 at iterations t and $t+1$, and hence does not need to be transmitted by the master node for this data shuffle. Moreover, note that index 6 does not appear in the subscript of the broadcast sub-messages, since W_6 is the ignored worker node. However, the subfiles assigned to W_6



	W_1	W_2	W_3	W_4	W_5	W_6
X_{123}	$B_{34} \oplus B_{35} \oplus B_{36}$	$C_{14} \oplus C_{15} \oplus C_{16}$	$A_{24} \oplus A_{25} \oplus A_{26}$	–	$A_{24} \oplus A_{26} \oplus B_{34} \oplus B_{36} \oplus C_{14} \oplus C_{16}$	$A_{24} \oplus A_{25} \oplus B_{34} \oplus B_{35} \oplus C_{14} \oplus C_{15}$
X_{124}	$B_{34} \oplus B_{45} \oplus B_{46}$	C_{14}	$A_{24} \oplus B_{45} \oplus B_{46}$	–	$A_{24} \oplus B_{34} \oplus B_{46} \oplus C_{14}$	$A_{24} \oplus B_{34} \oplus B_{45} \oplus C_{14}$
X_{125}	$B_{35} \oplus B_{45} \oplus B_{56}$	C_{15}	$A_{25} \oplus B_{45} \oplus B_{56} \oplus E_{12} \oplus F_{12}$	–	F_{12}	$A_{25} \oplus B_{35} \oplus B_{45} \oplus C_{15} \oplus E_{12}$
X_{134}	B_{34}	$A_{45} \oplus A_{46} \oplus C_{14}$	$A_{24} \oplus A_{45} \oplus A_{46}$	–	$A_{24} \oplus A_{46} \oplus B_{34} \oplus C_{14}$	$A_{24} \oplus A_{45} \oplus B_{34} \oplus C_{14}$
X_{135}	B_{35}	$A_{45} \oplus A_{56} \oplus C_{15} \oplus E_{13} \oplus F_{13}$	$A_{25} \oplus A_{45} \oplus A_{56}$	–	F_{13}	$A_{25} \oplus A_{45} \oplus B_{35} \oplus C_{15} \oplus E_{13}$
X_{145}	B_{45}	$A_{45} \oplus E_{14} \oplus F_{14}$	$A_{45} \oplus B_{45} \oplus E_{14} \oplus F_{14}$	–	F_{14}	$A_{45} \oplus B_{45} \oplus E_{14}$
X_{234}	$B_{34} \oplus C_{45} \oplus C_{46}$	$C_{14} \oplus C_{45} \oplus C_{46}$	A_{24}	–	$A_{24} \oplus B_{34} \oplus C_{14} \oplus C_{46}$	$A_{24} \oplus B_{34} \oplus C_{14} \oplus C_{45}$
X_{235}	$B_{35} \oplus C_{45} \oplus C_{56} \oplus E_{23} \oplus F_{23}$	$C_{15} \oplus C_{45} \oplus C_{56}$	A_{25}	–	F_{23}	$A_{25} \oplus B_{35} \oplus C_{15} \oplus C_{45} \oplus E_{23}$
X_{245}	$B_{45} \oplus C_{45} \oplus E_{24} \oplus F_{24}$	C_{45}	$B_{45} \oplus E_{24} \oplus F_{24}$	–	F_{24}	$B_{45} \oplus C_{45} \oplus E_{24}$
X_{345}	$C_{45} \oplus E_{34} \oplus F_{34}$	$A_{45} \oplus C_{45} \oplus E_{34} \oplus F_{34}$	A_{45}	–	F_{34}	$A_{45} \oplus C_{45} \oplus E_{34}$

(c)

Fig. 6. Data Shuffling system with $N = K = 6$, $S = 3$ and $\gamma = 3$. The file transition graph is depicted in Fig. 5. (a) Cache organization of worker nodes at iteration t, along with the set of subfiles which are not available in the caches at iteration t and need to be processed at iteration $t+1$. (b) Cache organization of worker nodes at iteration $t+1$ after updating the caches. For instance, subfiles $\{B_{13}, B_{14}, B_{15}, B_{16}\}$ in $E_1(t)$ are moved to $E_2(t+1)$. (c) Received functions by worker nodes after removing the cached subfiles. The complete received functions at worker nodes are expressed in (22) and (23).

can be recovered by linear combination of other transmitted sub-messages.

For each worker node, Fig. 6c shows the received sub-messages from the master node after removing the subfiles that already exist in its cache. The decoding procedure of the proposed coded shuffling scheme is analogous to interference mitigation techniques in wireless communications. To present this analogy, we focus on three different cases of the decoding procedure.

(i) Decoding C_{14} from X_{124} by W_2:

$$X_{124} = \underbrace{C_{14}}_{\text{Desired subfile}} \oplus \underbrace{A_{24} \oplus B_{34} \oplus B_{45} \oplus B_{46}}_{\text{Cached subfiles in } \mathcal{Z}_2}.$$

The decoding procedure is analogous to interference suppression technique. W_2 decodes C_{14} by canceling the interfering subfiles using its cache contents Z_2.

(ii) Decoding C_{16} from X_{123} by W_2:

$$X_{123} = \underbrace{C_{16}}_{\text{Desired subfile}} \oplus \underbrace{C_{14}}_{\text{Decoded subfile from } X_{124}} \oplus \underbrace{C_{15}}_{\text{Decoded subfile from } X_{125}} \oplus \underbrace{A_{24} \oplus A_{25} \oplus A_{26} \oplus B_{34} \oplus B_{35} \oplus B_{36}}_{\text{Cached subfiles in } Z_2}.$$

The decoding procedure is analogous to successive interference cancellation (SIC) technique. W_2 decodes C_{16} by first canceling the subfiles that exist in Z_2. Next, it exploits the subfiles decoded from X_{124} and X_{125} to successively cancel the remaining interfering subfiles.

(iii) Decoding E_{23} from X_{235} by W_6:

$$X_{235} = \underbrace{E_{23}}_{\text{Desired subfile}} \oplus (\underbrace{C_{56}}_{\text{Cached subfiles in } Z_6} \oplus F_{23}) \oplus (\underbrace{A_{25} \oplus B_{35} \oplus C_{15} \oplus C_{45}}_{X_{123} \oplus X_{234}}).$$

The decoding procedure is analogous to aligned interference suppression (interference alignment) technique. W_6 decodes E_{23} by first canceling the subfiles cached in Z_6. Then, the remaining interfering subfiles are the result of XORing some other received sub-messages, i.e., $X_{123} \oplus X_{234}$, and hence, they can be canceled accordingly.

As a result, the achieved delivery load is $R_{\text{coded}} = \binom{5}{3} / \binom{5}{2} = 1$. On the other hand, the delivery load achieved by the uncoded shuffling scheme, under the same placement strategy, is $R_{\text{uncoded}} = (5 \times 6)/\binom{5}{2} = 3$. That is, the proposed coded shuffling scheme can save around 66% of the communication load, and thus, it speeds up the overall run-time of the data shuffling process. When each worker node decodes all missing subfiles at iteration t, Fig. 6b depicts the cache organization of each worker node after updating the cache in preparation for the following data shuffle at iteration $t+1$. Note that the subfiles can be relabeled in a similar way as in Example 1.

C. A Graph-Based Delivery Scheme for Any Shuffling: Proof of Theorem 2

The coded shuffling scheme proposed in Section IV-A provides the worker nodes with the missing parts of their assigned files, using the broadcast message and cached subfiles. However, depending on the file transition graph, the delivery load obtained by that scheme may be sub-optimum. As an extreme and hypothetical example, consider a file transition graph where each worker node W_i is assigned the same file to process at iterations t and $t+1$, i.e., $d(i) = u(i)$. Clearly, no communication between the master node and worker nodes is needed in this case, and hence, $R = 0$ is achievable. This implies the scheme in Section IV-A is sub-optimal for this instance of the shuffling problem.

It turns out that the *number of cycles* in the file transition graph is the main characteristic to determine the optimum delivery scheme. More concretely, for a file transition graph with γ cycles where $\gamma - 1 \geq S$, we show that there are precisely $\binom{\gamma-1}{S}$ sub-messages in (15) that are linearly dependent on the other sub-messages. Thus, by refraining from broadcasting these sub-messages, we can reduce the delivery load, and achieve the one given in (3). In the decoding phase, worker nodes can first recover all the redundant sub-messages that have not been transmitted by the master node by computing linear combinations of appropriate sub-messages that have been received. Then, each worker node follows the same decoding rules, discussed in Section IV-A, to decode the assigned file at iteration $t+1$. This results in an opportunistic coded shuffling scheme based on the scheme proposed in Section IV-A. We will later show in Section V that this scheme is indeed optimum, and achieves the minimum possible delivery load.

The following lemma characterizes the linearly dependent sub-messages, and quantifies the reduced delivery load:

Lemma 3: Consider a data shuffling system with a master node, K worker nodes with storage capacity per worker node S, and $N = K$ files, and a given file transition graph that comprises γ cycles. Consider the placement strategy given in Section III and the delivery scheme provided in Section IV-A. Then, there are a total of $\binom{\gamma-1}{S}$ redundant (linearly dependent) sub-messages among the $\binom{K-1}{S}$ broadcasting sub-messages. We refer to Appendix C for the proof of lemma 3.

In fact, we can explicitly characterize the set of redundant sub-messages that are not broadcast in the delivery phase. To this end,

- consider the first $\gamma - 1$ cycles out of the total of γ cycles formed by the file transition graph (generally, we have to consider all cycles except the one that includes the ignored worker node);
- from these $\gamma - 1$ cycles, consider all possible combinations of sets of cycles that have S distinct cycles. There are $\binom{\gamma-1}{S}$ such sets;
- consider all sub-messages X_Δ (defined in (16)) where Δ has exactly one worker node from each of the chosen S cycles.

In the proof of Lemma 3, we show that the sum of all such sub-messages is zero. Thus, the master node can remove one of the sub-messages from each group and transmit the rest of them to the worker nodes. Therefore, the resulting broadcast communication load R is upper bounded by

$$R \leq \frac{\binom{K-1}{S} - \binom{\gamma-1}{S}}{\binom{K-1}{S-1}}.$$

In the decoding phase, each worker node first reconstructs the missing, redundant sub-messages by adding the other sub-messages in the group, and then follows the same decoding scheme, and cache updating and subfile relabeling strategies introduced in Section IV-A. This completes the proof of Theorem 2. In the next example, we illustrate the concept of redundancy within the the set of broadcast sub-messages.

D. Illustrative Example

Example 3 (Multiple-Cycle File Transition Graph with Opportunistic Transmission): We consider the same system parameters of Example 2 in Section IV-B with the same file

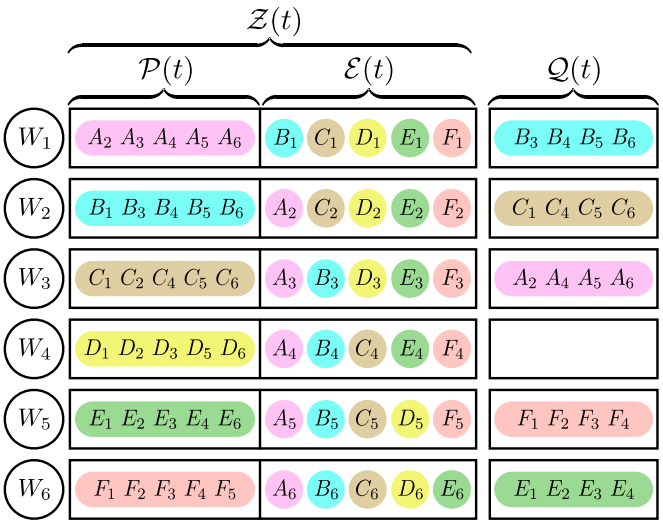


Fig. 7. Cache organization of worker nodes at iteration t, along with the set of subfiles which are not available in the caches at iteration t and need to be processed at iteration $t+1$, for a data Shuffling system with $N = K = 6$, $S = 2$ and $\gamma = 3$. The file transition graph for this system is depicted in Fig. 5.

transition graph given in Fig. 5, except that here the cache size of each worker node is $S = 2$ files. Fig. 7 captures the cache organization of worker nodes, along with the missing subfiles that need to be processed at iteration $t+1$. Following the achievable scheme proposed in IV-A, the set of sub-messages transmitted by the master node to the worker nodes is expressed as

$$\begin{aligned}
X_{12} &= A_2 \oplus B_3 \oplus B_4 \oplus B_5 \oplus B_6 \oplus C_1,\\
X_{13} &= A_2 \oplus A_4 \oplus A_5 \oplus A_6 \oplus B_3 \oplus C_1,\\
X_{14} &= A_4 \oplus B_4,\\
X_{15} &= A_5 \oplus B_5 \oplus E_1 \oplus F_1,\\
X_{23} &= A_2 \oplus B_3 \oplus C_1 \oplus C_4 \oplus C_5 \oplus C_6,\\
X_{24} &= B_4 \oplus C_4,\\
X_{25} &= B_5 \oplus C_5 \oplus E_2 \oplus F_2,\\
X_{34} &= A_4 \oplus C_4,\\
X_{35} &= A_5 \oplus C_5 \oplus E_3 \oplus F_3,\\
X_{45} &= E_4 \oplus F_4.
\end{aligned} \tag{24}$$

Note that the file transition graph consists of $\gamma = 3$ cycles, namely, $\{W_1, W_2, W_3\}$, $\{W_4\}$, and $\{W_5, W_6\}$, where the ignored worker node W_6 appears in the third cycle. Ignoring the third cycle, we have two remaining ones. The family of all Δ's with exactly one entry from each of the first and second cycle is given by $\{\{1,4\},\{2,4\},\{3,4\}\}$. It is evident that X_{14}, X_{24} and X_{34} are linearly dependent, since $X_{34} \oplus X_{14} \oplus X_{24} = 0$. Therefore, we can safely omit X_{34} from the set of the transmitted sub-messages. For decoding purposes, worker nodes can recover X_{34} from X_{14} and X_{24}. Therefore, each one of them is able to decode the assigned file at iteration $t+1$ by following the decoding procedure presented in Section IV-A.

V. Converse Proof for Data Shuffling With Canonical Setting ($N = K$)

We prove the optimality of the coded shuffling scheme proposed in Section IV-C, as stated in Theorem 3. The instance of the problem is fully determined by the indices of files to be processed by the worker nodes at iterations t and $t+1$, i.e., $u(\cdot)$ and $d(\cdot)$, respectively. Without loss of generality, we may assume $u(i) = i$ for every $i \in [K]$, otherwise we can relabel the files. These assignment functions induce a directed file transition graph $G(V, E)$, with node set $V = \{W_1, W_2, \ldots, W_K\}$, in which there is an edge from W_i to W_j if and only if $d(W_j) = F^i$.

Since the in-degrees and out-degrees of each vertex are equal to 1, such a graph consists of a number of directed cycles. Let γ denote the number of cycles in this graph with cycle lengths given by $\{\ell_1, \ell_2, \ldots, \ell_\gamma\}$. Then each node in the graph can be represented by a pair (c, p), where $c \in \{1, 2, \ldots, \gamma\}$ denoted the cycle number, and $p \in \{1, 2, \ldots, \ell_c\}$ denotes the position within cycle c. With this notation[6], we have $d(c,p) = u(c, p+1) = (c, p+1)$, i.e., the worker node at position (c,p) will process file $F^{(c,p)}$ and $F^{(c,p+1)}$ at iterations t and $t+1$, respectively[7]. Note that the *positional label* (c,p) essentially induces an *order* on the nodes and edges of the graph. For two pairs (c',p') and (c,p), we say (c',p') appears before (c,p) and denote it by $(c',p') < (c,p)$ if either (c',p') appears in a cycle with smaller index ($c' < c$) or in the same cycle but at an smaller position ($c' = c$ and $p' < p$). Similarly, $(c',p') \leq (c,p)$ indicates that either $(c',p') < (c,p)$ or $(c',p') = (c,p)$.

Consider an arbitrary *uncoded placement* of the files in worker nodes' cache, and denote by F_j^i the bits of file F^i cached at W_j. Note that we do not make any assumptions on the size or symmetry of F_j^i's. Then, the cache contents of worker node W_j is given by $Z_j = F^j \cup_{i=j} F_j^i$, which is equivalent to

$$Z_{(c,p)} = F^{(c,p)} \cup \left(\bigcup_{(c',p')=(c,p)} F_{(c,p)}^{(c',p')} \right), \tag{25}$$

using the positional labeling, where j is the worker node in the pth position of the cth cycle.

For a given file transition graph $G(V, E)$, we introduce a virtual worker node W equipped with a cache Z, in which we store

$$Z = \left(\bigcup_{c\in[\gamma]} F^{(c,1)} \right) \cup \left(\bigcup_{c\in[\gamma]} \left(\bigcup_{\substack{(c',p')\\ (c,p)\\ p'>1}} F_{(c,p)}^{(c',p')} \right) \right) \tag{26}$$

$$= \left(\bigcup_{c\in[\gamma]} F^{(c,1)} \right) \cup \left(\bigcup_{c\in[\gamma]} \left(\bigcup_{\substack{(c',p')\\ p'<\ell_c\\ (c',p')<(c,p)}} F_{(c',p')}^{(c,p)} \right) \right), \tag{27}$$

where

[6]The selection of the order of cycles, as well as the starting position within each cycle, is arbitrary.

[7]We define $(c, \ell_c + 1) = (c, 1)$ for the sake of consistency.

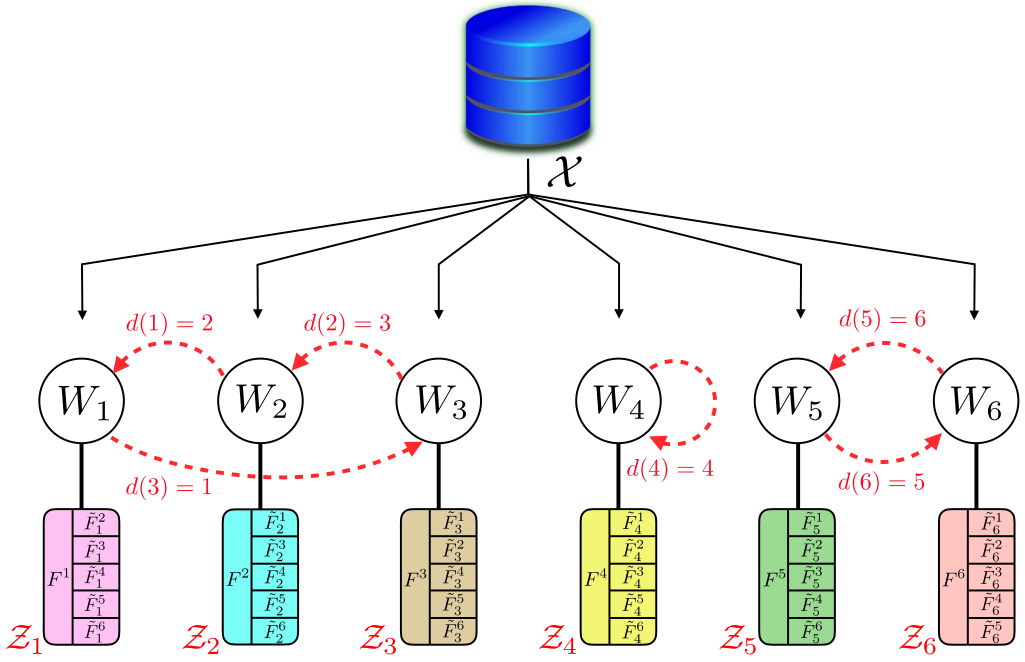


Fig. 8. The file transition graph, along with the cache organization of worker nodes, for a data Shuffling system with $N = K = 6$, and $\gamma = 3$.

- in (26), the first equality reads as follows. the cache Z of the virtual worker node W is the union of two sets. The first set is the union of files being processed by the first worker node in each cycle. The second set only consists the sub-files of all other files, and only includes sub-files that are stored at worker nodes that (i) whose rank is the circular order is less that that of the processing worker, and (ii) do not appear at the last position of their cycle.
- in (27), the second equality in (25) is just a re-arrangement of the sub-files selected in (24). That will follows form two steps applied simultaneously, i.e., (i) swapping the labels used for the subscript and superscripts of F, and (ii) changing the order of the unions.

In order to prove the optimality of the proposed coded shuffling scheme, we first show that the broadcast message X and the cache contents of the virtual worker node suffice to decode all the files (Lemma 4). Then, we lower bound the size of the broadcast message by upper bounding the size of the data cached at the virtual worker node (Lemma 5 and Lemma 6).

Lemma 4: For any file transition graph G, given the cache contents Z of the virtual worker node W and the broadcast message X, all files in the data shuffling system can be decoded, that is

$$H\left(\{F^i\}_{i=1}^{K} | X, Z\right) = 0. \tag{28}$$

We refer to Appendix D for the proof of Lemma 4.

For a file F^i and a subset of worker nodes $J \subseteq [K] \setminus \{i\}$, we define the size of a union of bits of F^i that are cached at worker nodes in J as $\mu_J^i = \left|\bigcap_{j \in J} F_j^i\right|$. For a given integer α, let μ_α denote the average (over files and worker nodes) size of a set of union of bits that are cached in the excess storage of α worker nodes, that is,

$$\begin{aligned}\mu_\alpha &= \frac{1}{K\binom{K-1}{\alpha}} \sum_{i \in [K]} \sum_{\substack{J \subseteq [K]\setminus i \\ |J|=\alpha}} \mu_J^i \\ &= \frac{1}{K\binom{K-1}{\alpha}} \sum_{i \in [K]} \sum_{\substack{J \subseteq [K]\setminus i \\ |J|=\alpha}} \left|\bigcap_{j \in J} F_j^i\right|. \end{aligned} \tag{29}$$

Next, the communication load for an instance of the shuffling problem, determined by a graph $G(V, E)$, can be lower bounded in terms of μ_i's as follows.

Lemma 5: For a data shuffling problem determined by a file transition graph $G(V, E)$, the communication load R is lower bounded by

$$R(G) \geq K - \gamma - \sum_{i=1}^{K-\gamma} \mu_i. \tag{30}$$

We refer to Appendix E for the proof of Lemma 5.

Before we continue with the formal proof, we present the defined notation and the main steps of our argument through an illustrative example.

Example 4: Consider a data shuffling system with $K = 6$ worker nodes, and $N = 6$ files, namely $F^1, F^2, F^3, F^4, F^5, F^6$. We assume worker node i is processing F^i at time t. Moreover, the file assignments for iteration $t + 1$ are given by $d(1) = 2$, $d(2) = 3$, $d(3) = 1$, $d(4) = 4$, $d(5) = 6$, and $d(6) = 5$. The file transition graph of the problem is depicted by Fig. 8, that consists of $\gamma = 3$ cycles, with cycle lengths $(\ell_1, \ell_2, \ell_3) = (3, 1, 2)$. Hence, the nodes are labeled using the (c, p) notation as follows: $1 \leftrightarrow (1, 1)$, $2 \leftrightarrow (1, 2)$, $3 \leftrightarrow (1, 3)$, $4 \leftrightarrow (2, 1)$, $5 \leftrightarrow (3, 1)$,

and $6 \leftrightarrow (3, 2)$. The cache contents of the virtual worker node will be

$$Z = F^{(1,1)}, F^{(2,1)}, F^{(3,1)} \cup F^{(1,2)}_{(1,1)}, F^{(1,3)}_{(1,1)}, F^{(1,3)}_{(1,2)}, F^{(3,2)}_{(1,1)}, F^{(3,2)}_{(1,2)}, F^{(3,2)}_{(3,1)},$$

Next, according to Lemma 4, we argue that given Z and X, the virtual worker node is able to decode all files as follows:

- W has the entire files $F^{(1,1)}$, $F^{(2,1)}$, and $F^{(3,1)}$, i.e., $H(F^{(1,1)}, F^{(2,1)}, F^{(3,1)}, Z) = 0$.
- It also has the entire data cached at W_1, that is,

$$Z_1 = F^{(1,1)}, F^{(1,2)}_{(1,1)}, F^{(1,3)}_{(1,1)}, F^{(2,1)}_{(1,1)}, F^{(3,1)}_{(1,1)}, F^{(3,2)}_{(1,1)}.$$

  Hence, it can decode $d(1) = F^{(1,2)}$ from (Z_1, X), which implies $H(F^{(1,2)} | Z, X) = 0$.
- Then, having $F^{(1,2)}$ decoded, the virtual worker node has the entire content of Z_2 which, together with X, suffices to decode the file assigned to W_2, which is $d(2) = 3 \leftrightarrow (1, 3)$. That is, $H(F^{(1,3)} | Z, X, F^{(1,2)}) = 0$.
- Finally, since files $F^{(1,1)}$, $F^{(2,1)}$, $F^{(3,1)}$ and $F^{(3,2)}_{(3,1)}$ are in Z, and files $F^{(1,2)}$, $F^{(1,3)}$ are decoded, the virtual worker node has the entire Z_5. Hence, the only remaining file $F^{(3,2)}$ can be recovered from (Z_5, X) since $d(5) = 6 \leftrightarrow (3, 2)$. This implies $H(F^{(3,2)} | Z, X, F^{(1,2)}, F^{(1,3)}) = 0$.

This argument shows that

$$\begin{aligned} 6 &= H(F^1, F^2, F^3, F^4, F^5, F^6) \\ &\leq H(X, Z) \\ &\leq H(X) + H(F^1) + H(F^4) + H(F^5) + H(F^2_1) \\ &\quad + H(F^3_1, F^3_2) + H(F^6_1, F^6_2, F^6_5), \end{aligned}$$

which implies

$$\begin{aligned} R &= H(X) \\ &\geq 6 - H(F^1) + H(F^4) + H(F^5) \\ &\quad - H(F^2_1) + H(F^3_1, F^3_2) + H(F^6_1, F^6_2, F^6_5) \\ &= 3 - H(F^2_1) + H(F^3_1, F^3_2) + H(F^6_1, F^6_2, F^6_5) \\ &\geq 3 - (\mu_1 + \mu_2 + \mu_3), \end{aligned}$$

where the last inequality follows from a similar argument on several versions of the same problem with re-labeled files, using set-theoretic operations. To avoid any repetition, we omit these steps here, and refer to the proof of Lemma 5 in E. Finally, we need to bound μ_1, μ_2, and μ_3, which is elaborated by lemma 6.

Finally, we seek an upper bound on μ_i's using a set theoretic argument, as stated in the following lemma:

Lemma 6: The variables μ_i's introduced in (29) satisfy

$$\mu_i \leq 1 - \frac{\binom{K-i-1}{S-1}}{\binom{K-1}{S-1}}, \tag{31}$$

for $\alpha \in \{0, 1, 2, \ldots, K-1\}$.

The proof of this lemma is presented in Appendix F. Having the lemmas above, we are ready to prove the optimality of the proposed coded shuffling scheme.

Proof of Theorem 3: We start with Lemma 5, and use Lemma 6 to upper bound μ_i's. We have

$$\begin{aligned} R(G) &\geq K - \gamma - \sum_{i=1}^{K-\gamma} \mu_i \\ &\geq K - \gamma - \left(K - \gamma - \sum_{i=1}^{K-\gamma} \frac{\binom{K-i-1}{S-1}}{\binom{K-1}{S-1}}\right). \\ &= \frac{1}{\binom{K-1}{S-1}} \sum_{i=1}^{K-\gamma} \binom{K-i-1}{S-1} \\ &= \frac{1}{\binom{K-1}{S-1}} \sum_{j=\gamma-1}^{K-2} \binom{j}{S-1} \\ &= \frac{1}{\binom{K-1}{S-1}} \left[\sum_{j=0}^{K-2} \binom{j}{S-1} - \sum_{j=0}^{\gamma-2} \binom{j}{S-1}\right] \\ &= \frac{1}{\binom{K-1}{S-1}} \left[\binom{K-1}{S} - \binom{\gamma-1}{S}\right]. \end{aligned} \tag{32}$$

This completes the proof of Theorem 3.

VI. Coded Shuffling Scheme for General Setting ($N \geq K$)

We shift our attention to the general and practical setting of the data shuffling problem when $N \geq K$. We assume N/K, the number of files to be processed by each worker node, is integer. Furthermore, unless otherwise mentioned, we assume $S/(N/K)$ is integer. It should be noted that the proposed coded shuffling algorithm for the general setting of $N \geq K$ is an extension to the one developed for the canonical setting of $N = K$.

A. A Universal Delivery Scheme for General Setting: Proof of Theorem 4

In the following, we extend the achievable scheme proposed in Section IV-C for the canonical setting ($N = K$) in order to develop an achievable scheme for the general setting of the shuffling problem, that is, for any $N \geq K$. To this end, we follow the cache placement scheme proposed in Section III. Then, for the delivery phase, we *decompose* the data shuffling problem into N/K instances of sub-problems, where each sub-problem consists of K worker nodes, K files, and cache size of $S = S/(N/K)$ per worker node. Therefore, the resulting sub-problems lie in the class of the canonical setting discussed earlier in Section IV.

We first present the following lemma that is essential for the proof of Theorem 4:

Lemma 7: For any data shuffling problem with K worker nodes and N files (where K divides N), $G(V, E)$ can be decomposed into N/K subgraphs $G_i(V, E_i)$ with $|E_i| = K$, for $i \in [N/K]$, such that

- For each subgraph $G_i(V, E_i)$, the in-degree and out-degree of each vertex in V are 1.

- The edge sets $\{E_i : i \in [N/K]\}$ provide a partitioning for E, i.e., $E_i \cap E_j = \emptyset$ for any distinct pair of $i, j \in [N/K]$, and $\bigcup_{i=1}^{N/K} E_i = E$.

The proof of this lemma is based on Hall's theorem, and presented in Appendix G.

Now, we prove Theorem 4 as follows. Recall from Section II that the file transition graph is defined as a directed graph $G(V, E)$, where an edge $e_j = (i,) \in E$ with $j \in [N]$ indicates that $j \in u(i) \cap d()$, i.e., file F^j is being processed by worker node W_i at iteration t, and will be processed by worker node W at iteration $t + 1$. Since each worker node processes N/K files at every iteration of the distributed algorithm, the directed graph $G(V, E)$ is regular since the in-degree and out-degree of each vertex in V are N/K. The decomposition of the shuffling problem is inspired by decomposing the file transition graph. More precisely, we decompose $G(V, E)$ into N/K subgraphs, namely $G_i(V, E_i)$ for $i \in [N/K]$, such that each subgraph induces one *canonical* instance of the problem. The existence of such a decomposition is guaranteed by Lemma 7.

Each resulting subgraph $G_i(V, E_i)$ induces a data shuffling system with K files (corresponding to the edges appear in the subgraph), K worker nodes, and storage capacity per worker node $S = S/(N/K)$. Then, we can apply the universal coded shuffling scheme proposed in Section IV-C to achieve a delivery load of $\binom{K-1}{S} / \binom{K-1}{S-1}$ for each sub-problem. As a result, an overall delivery load of

$$R = \frac{N}{K} \frac{\binom{K-1}{S}}{\binom{K-1}{S-1}}$$

is achievable for any file transition graph. This completes the proof of Theorem 4.

The scheme is proved to be optimal for the *worst-case shuffling scenario* with K worker nodes and N files in Section VI-D.

B. A Graph-Based Delivery Scheme for General Setting

The delivery scheme proposed in Section VI-A is based on applying the universal delivery scheme on each subgraph of the file transition graph after decomposing it. We discussed in Section IV-C that the delivery load obtained by this scheme can be sub-optimum, depending on the topology of the graph, and then we proposed an opportunistic approach to slash the delivery load. Consequently, we can apply the graph-based delivery scheme of Section IV-C on each subgraph $G_i(V, E_i)$, for $i \in [N/K]$, of the file transition graph, to reduce the delivery load. Let γ_i be the number of cycles in the subgraph $G_i(V, E_i)$. Theorem 2 implies that the delivery load of $\frac{\binom{K-1}{S} - \binom{\gamma_i - 1}{S}}{\binom{K-1}{S-1}}$ is achievable for the i-th subgraph. Therefore, we have the following corollary:

Corollary 2: Consider a data shuffling system that processes N files using K worker nodes, each equipped with a cache of size S. Assume that the file transition graph $G(V, E)$ is decomposed into N/K subgraphs $G_i(V, E_i)$, for $i \in [N/K]$, and denote by γ_i the number of cycles in subgraph G_i. Then, a total delivery load of

$$R = \sum_{i=1}^{N/K} \frac{\binom{K-1}{S} - \binom{\gamma_i - 1}{S}}{\binom{K-1}{S-1}} = \frac{1}{\binom{K-1}{S-1}} \left(\frac{N}{K} \binom{K-1}{S} - \sum_{i=1}^{N/K} \binom{\gamma_i - 1}{S} \right) \tag{33}$$

is achievable for the file assignments given by the file transition graph.

Next, we explain the details of the proposed shuffling schemes through an illustrative example.

C. Illustrative Example

Example 5: We consider a data shuffling system with $K = 4$ worker nodes with cache size of $S = 4$ files, and $N = 8$ files, denoted by $\{F^1, F^2, F^3, F^4, F^5, F^6, F^7, F^8\}$. For notational simplicity, we rename the files as $\{A, B, C, D, E, F, G, H\}$, respectively. Each worker node stores $N/K = 2$ files to process at iteration t, and caches $S - N/K = 2$ files in the excess storage part. Fig. 9a depicts the underlying file transition graph $G(V, E)$. For instance, worker node W_1 processes two files A and E at iteration t. At iteration t+1, file A will be again processed by W_1, while file E will be processed by W_4. Therefore, there are two directed edges outgoing from W_1; one from W_1 to W_1 (labeled by A), and the other from W_1 to W_4 (labeled by E). Fig. 9b depicts the cache organization of worker nodes at iteration t, along with the missing subfiles that need to be processed at iteration $t + 1$. Note that cache placement for the excess storage is symmetric across files and worker nodes, and does not depend on the file transition graph.

After constructing of $G(V, E)$, we decompose it into N/K subgraphs as discussed in Section VI-A. Fig. 10a shows a decomposition of $G(V, E)$ into $N/K = 2$ subgraphs, designated G_1 and G_2. Each of such subgraphs induces an instance of the coded shuffling problem with parameters $N_i = K = 4$, for $i \in \{1, 2\}$, and $S = 2$. The corresponding cache contents and file assignments for the problems induced by G_1 and G_2 are given in Fig. 11. It is clear from Theorem 1 that a communication load of $\binom{K-1}{S} / \binom{K-1}{S-1} = \binom{3}{2} / \binom{3}{1} = 1$ is achievable for each subgraph, and hence the total delivery load is $R = R_1 + R_2 = 2$.

The delivery load can be potentially reduced by exploiting the cycles in the subgraphs of file transition graph, and refraining from broadcasting redundant sub-messages. However, for subgraphs G_1 and G_2 with $\gamma_1 = \gamma_2 = 2$ cycles, there is no communication load reduction due to a graph-based delivery scheme, since, according to Theorem 2, we have

$$R_1 = \frac{\binom{3}{2} - \binom{1}{2}}{\binom{3}{1}} = 1, \quad R_2 = \frac{\binom{3}{2} - \binom{1}{2}}{\binom{3}{1}} = 1,$$

and $R = R_1 + R_2 = 2$.

It is worth noting that the graph decomposition proposed here is not unique. In particular, another possible decomposition of $G(V, E)$ is depicted in Fig. 10b. Here, G_1 and G_2 are subgraphs obtained by decomposing G, and they consist of

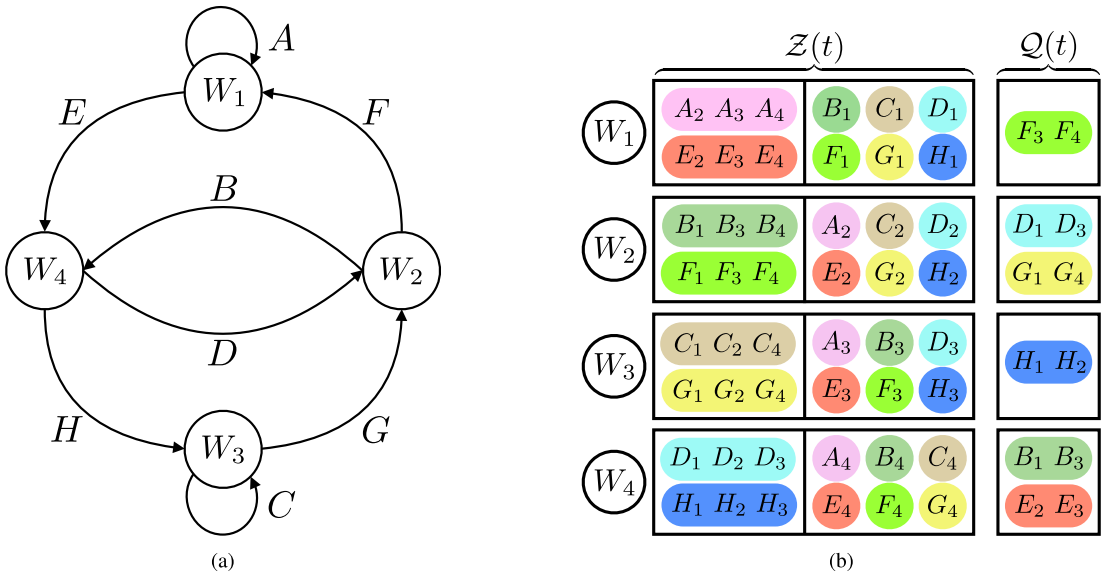


Fig. 9. Data Shuffling system with $N = 8$, $K = 4$ and $S = 4$. (a) The file transition graph $G(V, E)$. (b) Cache organization of worker nodes at iteration t, along with the set of subfiles which are not available in the caches at iteration t and need to be processed at iteration $t + 1$.

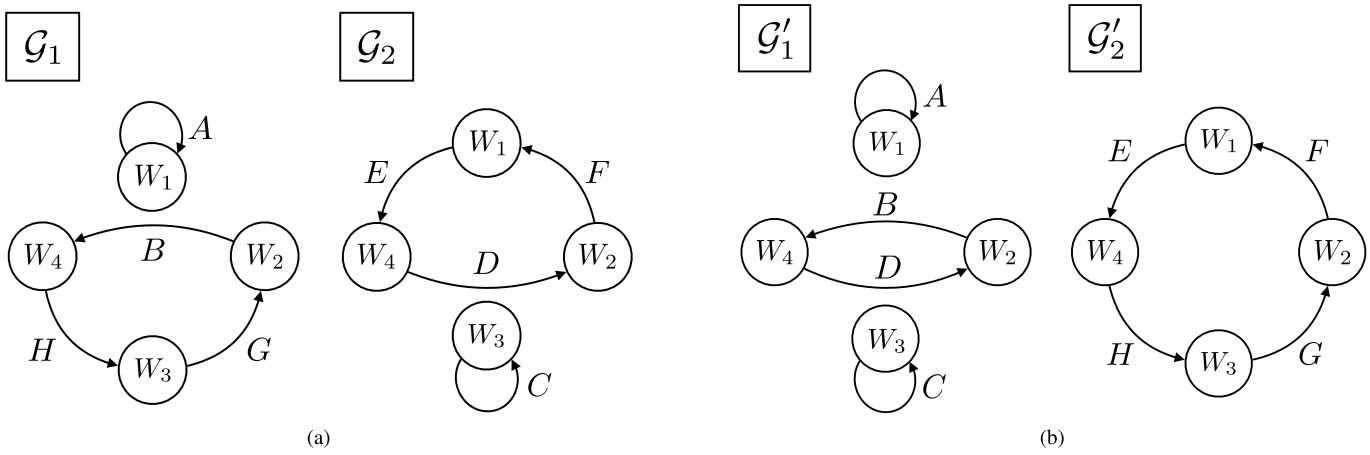


Fig. 10. (a) One possible decomposition of $G(V, E)$, depicted in Fig. 9, into $N/K = 2$ subgraphs; G_1 and G_2. (b) Another possible decomposition of $G(V, E)$ into $N/K = 2$ subgraphs; G'_1 and G'_2.

$\gamma_1 = 3$ and $\gamma_2 = 1$ cycles, respectively. Therefore, applying the graph-based delivery scheme of Section IV-C, the delivery loads of the subgraphs are given by

$$R_1 = \frac{\binom{3}{2} - \binom{2}{2}}{\binom{3}{1}} = \frac{2}{3}, \quad R_2 = \frac{\binom{3}{2} - \binom{0}{2}}{\binom{3}{1}} = 1,$$

and $R = R_1 + R_2 = 5/3 < R$. As a result, the second decomposition provides a lower communication load than the first decomposition.

D. Optimality for the Worst Case Shuffling: Proof of Theorem 5

In order to prove Theorem 5, we present one instance of the shuffling problem, for which the delivery load of Theorem 4 is indeed required, and cannot be further reduced. This shows that the upper bound on the delivery load given in Theorem 4 is optimum for such worst-case shuffling scenarios.

Let us consider a data shuffling problem $\mathbf{P}(N, K, S)$ with N files, K worker nodes, and cache memory size of S files. The size of each file is normalized to 1 unit. At each iteration, each worker node processes N/K files out of the N files, and hence $|u(i)| = |d(i)| = N/K$ for $i \in [K]$. The shuffling scenario we consider here is given by $d(i) = u(i + 1)$ for $i \in [K - 1]$ and $d(K) = u(1)$, i.e., all the files being processed by one worker node at iteration t are assigned to another worker node at iteration $t + 1$. For each worker node W_i, where $i \in [K]$, the expression for the cache contents Z_i, that comprises under-processing part P^i and excess storage part E_i, at iteration t is defined by (11). Moreover, the set of subfiles Q_i to be processed by W_i at iteration $t + 1$ is defined by (14). Suppose, for contradiction, that there exists a shuffling scheme that achieves a delivery load R where

$$R(\mathbf{P}) < \frac{N}{K} \frac{\binom{K-1}{S}}{\binom{K-1}{S-1}}. \tag{34}$$

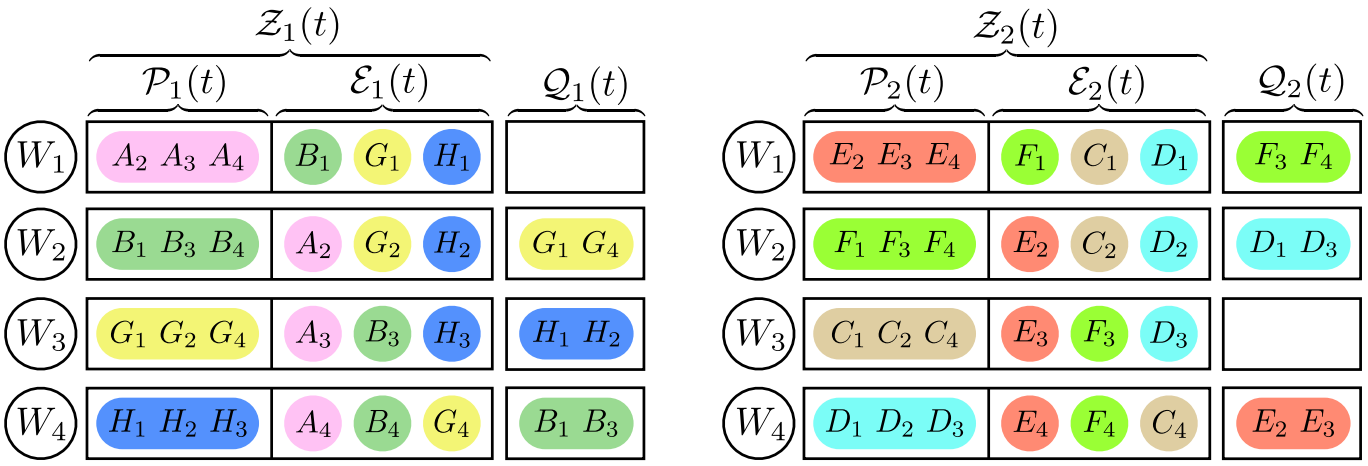


Fig. 11. Cache organization of worker nodes at iteration t, along with the set of subfiles which are not available in the caches at iteration t and need to be processed at iteration $t+1$, for the two subgraphs; G_1 and G_2, of the first decomposition, captured by Fig. 10a, of $G(V, E)$.

Now, let us consider another data shuffling problem $\mathbf{P}(N, K, S)$ with $N = K$ files, $K = K$ worker nodes, and $S = S/(N/K) = S$ files. We denote the files of this instance of the problem by $\{F^1, F^2, \ldots, F^K\}$. Each worker node processes 1 file at each iteration, i.e., $|u(i)| = |d(i)| = N/K = 1$ for $i \in [K]$. Let us divide each file in $\mathbf{P}(N, K, S)$ into N/K mini-files, each of which is of size $1/(N/K)$. More precisely, let $F^i(r)$ be the r^{th} mini-file of file F^i for $i \in [K]$ and $r \in [N/K]$. Consequently, for $i \in [K]$, the corresponding expressions of Z_i and Q_i are given by

$$Z_i = P^i \cup E_i = P^i \cup \bigcup_{\in [K]\backslash i} E_i \quad , \tag{35}$$

where

$$P^i = \quad F^i_\Gamma(r) : \Gamma \subseteq [K]\backslash\{i\}, |\Gamma| = \quad S-1, \ r \in [N/K] \quad , \tag{36}$$

$$E_i = \quad F_\Gamma(r) : \ = i, \quad i \in \Gamma \subseteq [K], |\Gamma| = S-1, \ r \in [N/K] \quad , \tag{37}$$

and

$$Q_i = \quad F_\Gamma(r) : \in d(i), \quad = i, \ i \notin \Gamma, \ \Gamma \subseteq [K], \\ |\Gamma| = S-1, \ r \in [N/K] \quad . \tag{38}$$

Therefore, $\mathbf{P}$ can be viewed as a data shuffling problem with $N \times (N/K) \quad = N$ mini-files each of size $1/(N/K)$, and $K = K$ worker nodes where each has a cache memory to store $S \times (N/K) = S$ mini-files. Viewing the data shuffling problem $\mathbf{P}(N, K, S)$ as a problem similar to $\mathbf{P}$, we can apply the delivery scheme of $\mathbf{P}$ (with mini-files of size $1/(N/K)$ instead of 1) and achieve a delivery load of

$$R(\mathbf{P}) = \frac{1}{N/K} R(\mathbf{P}) \\ < \frac{1}{N/K} \frac{N}{K} \frac{\binom{K-1}{S}}{\binom{K-1}{S-1}} = \frac{\binom{K-1}{S}}{\binom{K-1}{S-1}} = \frac{\binom{K-1}{S}}{\binom{K-1}{S-1}}, \tag{39}$$

which contradicts Corollary 1.

E. On the Sub-Optimality of Decomposition-Based Delivery

The delivery load proposed in Corollary 2 depends on the decomposition of the file transition graph $G(V, E)$. As discussed in Example 5, such decomposition is not unique. Therefore, one can minimize the delivery load in (33) by exhaustively searching over all possible decompositions of the file transition graph $G(V, E)$. A natural question is whether the delivery load obtained by such a best decomposition is optimum.

In the following example, we show that the answer to this question is "No", and a decomposition-based delivery scheme can be sub-optimum for a general file transition graph. Consider a data shuffling system with $K = 5$ worker nodes, and $N = 10$ files, denoted by $\{A, B, C, D, E, F, G, H, H, I, J\}$. The cache available at each worker node is $S = 2$, i.e., $S = S/(N/K) = 1$, which implies that there is no excess storage. The file transition graph of this problem is depicted in Fig 12a. It turns out that there is only one possible decomposition of this graph, shown in Fig. 12b, that satisfies the conditions given in Section VI-A. It is clear that each subgraph has only one cycle, i.e., $\gamma_1 = \gamma_2$. Hence, from Corollary 2, the delivery load is given by

$$R = \frac{\binom{K-1}{S} - \binom{\gamma_1 - 1}{S}}{\binom{K-1}{S-1}} + \frac{\binom{K-1}{S} - \binom{\gamma_2 - 1}{S}}{\binom{K-1}{S-1}} \\ = \frac{\binom{4}{1} - \binom{0}{1}}{\binom{4}{0}} + \frac{\binom{4}{1} - \binom{0}{1}}{\binom{4}{0}} = 8.$$

Now, let us consider an alternative transmission strategy as follows:

$$X = \{A \oplus D, \quad B \oplus I, \ C \oplus F, \ E \oplus H, \ G \oplus J\}.$$

It is easy to check that all the worker nodes can recover their assigned files from their cache contents and X. Since we transmit 5 sub-messages, each of which is of size 1, then the corresponding delivery load is $R = 5$, which is strictly less than $R = 8$, can be achieved. Indeed, this delivery scheme is inspired by a different decomposition of the graph $G(V, E)$, as depicted in Fig. 13. This decomposition consists

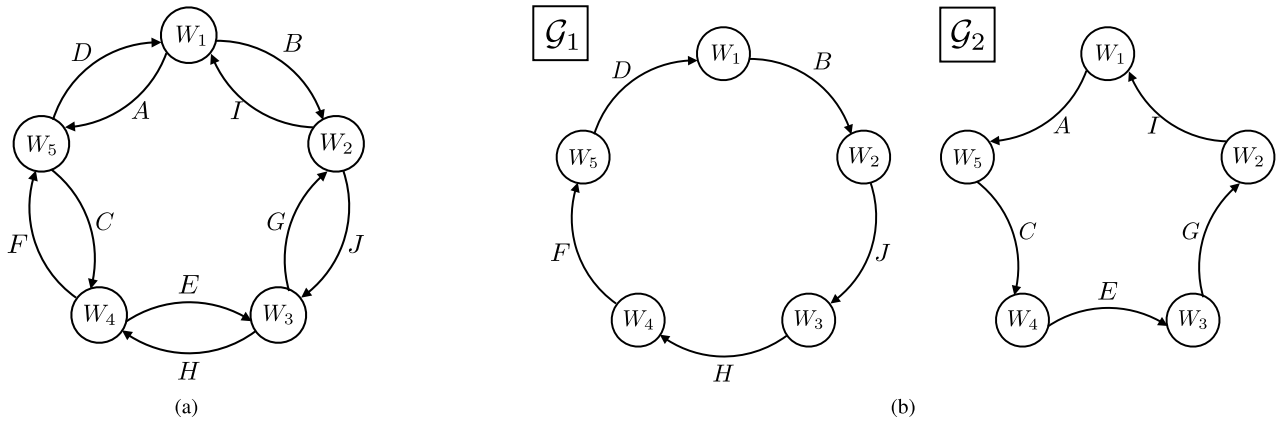


Fig. 12. Data Shuffling system with $N = 10$, $K = 5$ and $S = 2$. (a) The file transition graph $G(V, E)$. (b) Decomposition of $G(V, E)$ into $N/K = 2$ subgraphs; G_1 and G_2.

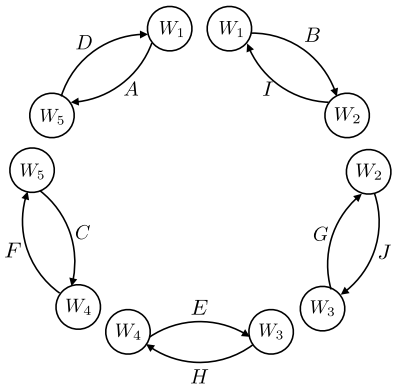


Fig. 13. A *different* decomposition of the file transition graph $G(V, E)$, shown in Fig.12a.

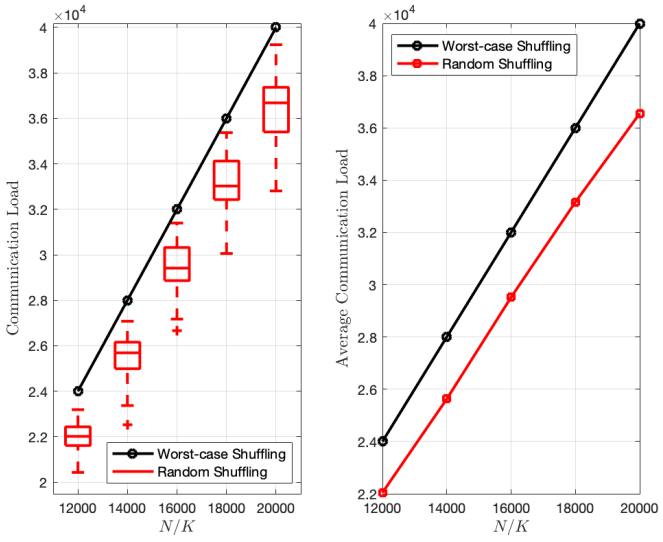


Fig. 14. The trade-off between the communication load versus N/K for a data shuffling problem with $K = 6$ and $S = S/(N/K) = 2$. The left subplot depicts range of achieved communication loads over 10^3 shuffling iterations, while the right subplot shows the corresponding average communication loads. The black curve is for the worst-case shuffling, while the red curve is for the random shuffling.

of 5 subgraphs, and the corresponding data shuffling problems do not lie in the class of canonical problems discussed in Section VI-A. This example shows that the upper bound on the delivery load, given in Corollary 2, is loose in general.

VII. Simulation Results

Next, we present simulation results for the achieved communication load when $N > K$ for random shuffling. In the below figures, we plot the communication load for different values of N/K. We set the number of worker nodes to $K = 6$, and evaluate the communication load as a function of N/K. We also set the normalized storage size to $S = 2$ in Fig. 14, and $S = 3$ in Fig. 15. The red box-plot in the left subplots of the figures depict the range of the achieved communication loads over 10^3 random shuffling scenarios. This shows that the communication load can significantly vary depending on the underlying random shuffling. The black curves, however, are the achieved communication loads for the worst-case shuffling scenario. On the other hand, the right subplots depict the corresponding average communication loads over 10^3 random shuffling scenarios (red curves), in comparison with the achieved communication loads for the worst-case shuffling scenario (black curves).

It is clearly evident that the proposed algorithm achieves a lower communication load compared to the one achieved for worst-case scenario in all figures. Moreover, the performance gap is more significant for smaller values of S. This is consistent with our theoretical result in Corollary 2: The saving in the communication load (compared to the worst-case shuffling) is proportional to

$$\sum_{i=1}^{N/K} \frac{\gamma_i - 1}{S}.$$

Therefore, each term in the summation gets smaller as S increases. However, as N/K increases, we have a larger number of terms contributing to the saving.

It should be noted that the algorithm used in our simulations for N/K graph decompositions (or more specifically

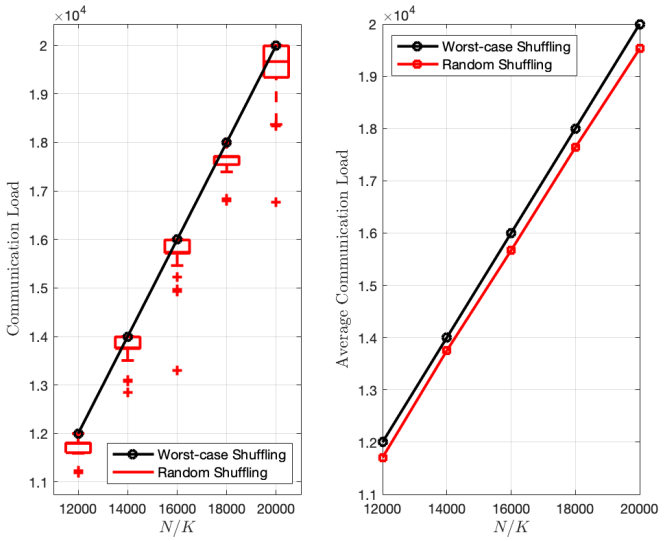


Fig. 15. The trade-off between the communication load versus N/K for a data shuffling problem with $K = 6$ and $S = S/(N/K) = 3$. The left subplot depicts range of achieved communication loads over 10^3 shuffling iterations, while the right subplot shows the corresponding average communication loads. The black curve is for the worst-case shuffling, while the red curve is for the random shuffling.

N/K perfect matchings) hinges on the Hungarian algorithm [28], [29]. The pseudocode of the graph decomposition algorithm is given in Algorithm 6, presented in Appendix H. In Remark 6, given in Appendix G, we present a brief discussion about the different algorithms for finding a perfect matching, along with their time complexities.

VIII. Conclusion

In this paper, we proposed a novel deterministic coded shuffling scheme which improves the state of the art by achieving a lower communication load for any shuffling when the number of files is equal to the number of worker nodes. Furthermore, the optimality of the proposed coded shuffling scheme was demonstrated through a matching converse proof. We showed that the placement phase of the proposed scheme, assuming uncoded prefetching, is optimal. Then, we exploited this canonical setting as a building block, and proposed a shuffling strategy for the general problem setting when the number of files is greater than or equal to the number of worker nodes. Moreover, we proved that the delivery load is optimum for worst-case shuffling. The characterization of the optimum trade-off for a given file transition graph is, however, still an open problem.

Promising directions for future research include the following. To complete our understanding of the problem, a topic of future work is to characterize the exact load-memory trade-off for any shuffling for the general setting of the shuffling problem. Moreover, in this work, we optimized the communication load of the data shuffling procedure for any two consecutive iterations, i.e., one-round shuffling. A more general framework consists of multiple consecutive shuffling iterations, and the design of a shuffling mechanism in order to achieve an enhanced overall delivery load would be of practical interest.

Appendix A
Proof of Lemma 1

We will prove that all the subfiles intended for W_i, $i \in [K-1]$, can be recovered from the family of broadcast sub-messages X and cache contents Z. Recall that all such subfiles are indexed by $F^{d(i)}_{\Gamma}$ for some $\Gamma \subseteq [K] \setminus \{i, d(i)\}$ with $|\Gamma| = S-1$, otherwise either the subfile is already cached at the worker node (when $i \in \Gamma$), or the subfile is zero (if $d(i) \in \Gamma$). We distinguish the following two cases:

A. $K \notin \Gamma$

The condition $K \notin \Gamma$ implies that $\{i\} \cup \Gamma \subseteq [K-1]$ and $|\{i\} \cup \Gamma| = S$. Hence, $X_{\{i\}\cup\Gamma} \in X$, as defined in (15), i.e., $X_{\{i\}\cup\Gamma}$ is one of the sub-messages broadcast by the master node. We can recover $F^{d(i)}_{\Gamma}$ from $X_{\{i\}\cup\Gamma}$ as follows:

$$X_{\{i\}\cup\Gamma} = \bigoplus_{i\in\{i\}\cup\Gamma} \Big(F^{i}_{(\{i\}\cup\Gamma)\setminus\{i\}} \oplus F^{d(i)}_{(\{i\}\cup\Gamma)\setminus\{d(i)\}} \oplus \Big(\bigoplus_{j\in[K]\setminus(\{i\}\cup\Gamma)} F^{d(i)}_{(\{j,i\}\cup\Gamma)\setminus\{i,d(i)\}} \Big) \Big) = \bigoplus_{i\in\{i\}\cup\Gamma} Y_i, \tag{40}$$

where we denote the summand for i by Y_i.

- First note that if $i = d(i)$ and $d(i) = i$ then we have $i \in (\{i\}\cup\Gamma)\setminus\{i\}$, $i \in (\{i\}\cup\Gamma)\setminus\{d(i)\}$, and $i \in (\{j, i\}\cup\Gamma)\setminus\{i, d(i)\}$. Hence, each term in
$$Y_i = F^{i}_{(\{i\}\cup\Gamma)\setminus\{i\}} \oplus F^{d(i)}_{(\{i\}\cup\Gamma)\setminus\{d(i)\}} \oplus \Big(\bigoplus_{j\in[K]\setminus(\{i\}\cup\Gamma)} F^{d(i)}_{(\{j,i\}\cup\Gamma)\setminus\{i,d(i)\}} \Big) \tag{41}$$
exists in the excess storage of worker node i (see definition of E in (13)) and hence Y_i can be removed from $X_{\{i\}\cup\Gamma}$ using the cache Z_i.
- Next, for i with $i \neq d(i)$ but $d(i) = i$ we have $i \in (\{i\}\cup\Gamma)\setminus\{i\}$ which implies $F^{i}_{(\{i\}\cup\Gamma)\setminus\{i\}} \in E \subset Z_i$. Moreover, from $d(i) = i$ we have
$$Y_i = \underbrace{F^{i}_{(\{i\}\cup\Gamma)\setminus\{i\}}}_{\in E} \oplus \underbrace{F_{\Gamma}}_{\in P} \oplus \Big(\bigoplus_{j\in[K]\setminus(\{i\}\cup\Gamma)} \underbrace{F_{(\{j\}\cup\Gamma)\setminus\{i\}}}_{\in P} \Big),$$
which shows Y_i can be fully removed using the cache contents Z_i.

- Finally, for $i =$ with $d(i) =$, we have

$$Y_i = F^{i}_{(\{\}\cup\Gamma)\setminus\{i\}} \oplus F^{d(i)}_{(\{\}\cup\Gamma)\setminus\{d(i)\}} \oplus \Big(\bigoplus_{j\in[K]\setminus(\{\}\cup\Gamma)} F^{d(i)}_{(\{j,\}\cup\Gamma)\setminus\{i,d(i)\}} \Big)$$
$$= F_\Gamma \oplus F^{d()}_{(\{\}\cup\Gamma)\setminus\{d()\}} \oplus \bigoplus_{j\in[K]\setminus(\{\}\cup\Gamma)} F^{d()}_{(\{j\}\cup\Gamma)\setminus\{d()\}}$$
$$= F_\Gamma \oplus \bigoplus_{j\in[K]\setminus\Gamma} \underbrace{F^{d()}_{(\{j\}\cup\Gamma)\setminus\{d()\}}}_{=0 \text{ if } d() \notin \{j\}\cup\Gamma}$$
$$= F_\Gamma \oplus \bigoplus_{j=d()} F^{d()}_{(\{j\}\cup\Gamma)\setminus\{d()\}} \quad (42)$$
$$= F_\Gamma \oplus F^{d()}_{\Gamma}, \quad (43)$$

where in (42) we have used the fact that $F^{d()}_{(\{j\}\cup\Gamma)\setminus\{d()\}}$ is non-zero only if $d() \in \{j\}\cup\Gamma$, because otherwise $|(\{j\}\cup\Gamma)\setminus\{d()\}| = S > S-1$. On the other hand, we know $d() \notin \Gamma$. Therefore, the only non-zero term is the one corresponding to $j = d()$. Also note that $F_\Gamma \in P$ exists in the cache of worker node .

Therefore, $X_{\{\}\cup\Gamma}$ can be written as

$$X_{\{\}\cup\Gamma} = \zeta_0 + F^{d()}_{\Gamma},$$

where the interference term ζ_0 can be completely removed using the cache contents Z. This implies $F^{d()}_{\Gamma}$ can be recovered from the received sub-message $X_{\{\}\cup\Gamma}$ and cache contents of W.

B. $K \in \Gamma$

When Γ is a set of indices that includes K, the desired subfile cannot be directly recovered from one single transmit sub-message, since the proposed broadcast strategy does not send any sub-message X_Δ with $K \in \Delta$. However, we will show that $F^{d()}_{\Gamma}$ can be still recovered from the summation of subfiles from the cache contents and previously sub-messages decoded by W. Define $\Gamma = (\Gamma\setminus\{K\})\cup\{d()\}$. Recall that we assume $d() = K$, otherwise $F^{d()}_{\Gamma} = 0$ is a dummy subfile. This implies $|\{\}\cup\Gamma| = S$. Then we have

$$X_{\{\}\cup\Gamma} = \bigoplus_{i\in\{\}\cup\Gamma} \Big(F^{i}_{(\{\}\cup\Gamma)\setminus\{i\}} \oplus F^{d(i)}_{(\{\}\cup\Gamma)\setminus\{d(i)\}} \oplus \Big(\bigoplus_{j\in[K]\setminus(\{\}\cup\Gamma)} F^{d(i)}_{(\{j,\}\cup\Gamma)\setminus\{i,d(i)\}} \Big) \Big)$$
$$= \bigoplus_{i\in\{\}\cup\Gamma} Y_i, \quad (44)$$

where Y_i is the summand for the corresponding i in (44). Similar to Appendix A-A, we can identify the following three cases:

- If i is such that $i =$ and $d(i) =$, then $\in (\{\}\cup\Gamma)\setminus\{i\}$, $\in (\{\}\cup\Gamma)\setminus\{d(i)\}$ and $\in (\{j,\}\cup\Gamma)\setminus\{i, d(i)\}$, i.e., all the subfiles added in Y_i are indexed by a set that includes . All such subfiles are cached in Z (see (11)), and hence Y_i can be reconstructed and removed from the summation in (40).
- For i with $d(i) =$ but $i =$, we have

$$Y_i = \underbrace{F^{i}_{(\{\}\cup\Gamma)\setminus\{i\}}}_{\in E} \oplus \underbrace{F_\Gamma}_{\in P} \oplus \Big(\bigoplus_{j\in[K]\setminus(\{\}\cup\Gamma)} \underbrace{F_{(\{j,\}\cup\Gamma)\setminus\{i,d(i)\}}}_{\in P} \Big),$$

which implies that the entire Y_i can be reconstructed from $Z = E \cup P$.

- Finally, for $i =$ with $d(i) = i$, we can simplify Y_i as

$$Y_i = F_\Gamma \oplus F^{d()}_{(\{\}\cup\Gamma)\setminus\{d()\}} \oplus \Big(\bigoplus_{j\in[K]\setminus(\{\}\cup\Gamma)} F^{d()}_{(\{j\}\cup\Gamma)\setminus\{d()\}} \Big)$$
$$= F_\Gamma \oplus \Big(\bigoplus_{j\in[K]\setminus\Gamma} F^{d()}_{(\{j\}\cup\Gamma)\setminus\{d()\}} \Big) \quad (45)$$
$$= F_\Gamma \oplus \Big(\bigoplus_{j\in[K-1]\setminus\Gamma} F^{d()}_{(\{j\}\cup\Gamma)\setminus\{d()\}} \Big) \oplus \Big(\bigoplus_{j=K} F^{d()}_{(\{j\}\cup\Gamma)\setminus\{d()\}} \Big) \quad (46)$$
$$= F_\Gamma \oplus \Big(\bigoplus_{j\in[K-1]\setminus\Gamma} F^{d()}_{(\{j\}\cup\Gamma)\setminus\{d()\}} \Big) \oplus F^{d()}_{\Gamma}, \quad (47)$$

where in (45) we have merged two summations over $j =$ and $j \in [K]\setminus(\{\}\cup\Gamma)$, and again split it to $j \in [K-1]\setminus\Gamma$ and $j = K$ in (46). In (47), we use the fact that

$$(\{K\}\cup\Gamma)\setminus\{d()\} = (\{K, d()\}\cup\Gamma\setminus\{K\})\setminus\{d()\} = \Gamma.$$

Also note that $F_\Gamma \in P \subseteq Z$.

Therefore, we get

$$X_{\{\}\cup\Gamma} = \zeta_1 + \underbrace{\Big(\bigoplus_{j\in[K-1]\setminus\Gamma} F^{d()}_{(\{j\}\cup\Gamma)\setminus\{d()\}} \Big)}_{\text{already decoded}} + F^{d()}_{\Gamma},$$

where ζ_1 is the sum of some subfiles that are cached in Z. Note that all subfiles in the second term are indexed by sets $(\{j\}\cup\Gamma)\setminus\{d()\}$ which do not include K, and hence already decoded by W as explained in Appendix A-A. Thus, the desired subfile $F^{d()}_{\Gamma}$ can be recovered from $X_{\{\}\cup\Gamma}$ by removing the interference using cached data (interference suppression), as well as the previously decoded subfiles (successive interference cancellation). This completes the proof of Lemma 1.

Appendix B
Proof of Lemma 2

We will show the decodability of subfiles assigned to the ignored worker node W_K by the master node at iteration $t+1$. More precisely, we need to show $F^{d(K)}$ can be recovered from

(X, Z_K). First, note that if $d(K) = K$, then $F^{d(K)} = F^{K}$ is already cached at W_K and the claim clearly holds. Next, note that the subfiles of $F^{d(K)}$ indexed by Γ with $K \in \Gamma$ are cached in $E_K^{d(K)} = \{F_{\Gamma}^{d(K)} : K \in \Gamma, \Gamma \subseteq [K] \setminus \{d(K)\}, |\Gamma| = S - 1\}$. Therefore, we can assume $d(K) = K$ and restrict our attention to $F_{\Gamma}^{d(K)}$ with $K \notin \Gamma$. Worker node W_K can add up the received sub-messages $X_{\{\}\cup\Gamma}$ over all $\in [K-1] \setminus \Gamma$. This summation can be simplified as (50), shown at the bottom of this page. Note that

- in (48), shown at the bottom of this page, the inner summation over $i \in \{\} \cup \Gamma$ is broken into $i \in \Gamma$ and $i =$;
- in (49), shown at the bottom of this page, the order of summation over and i is reversed;
- and finally, the double sum in (49) is decomposed into $\mathsf{Term}_1 \oplus \mathsf{Term}_2 \oplus \mathsf{Term}_3$, and the last summation in (49) is decomposed into $\mathsf{Term}_4 \oplus \mathsf{Term}_5$.

Next, we can rewrite Term_2 in (50) as

$$\mathsf{Term}_2 = \Big(\bigoplus_{i \in \Gamma} \bigoplus_{\in [K-1]\setminus\Gamma} F^{d(i)}_{(\{\}\cup\Gamma)\setminus\{d(i)\}} \Big) \tag{51}$$

$$= \Big(\bigoplus_{\in\Gamma} \bigoplus_{j \in [K-1]\setminus\Gamma} F^{d()}_{(\{j\}\cup\Gamma)\setminus\{d()\}} \Big) \tag{52}$$

where

- in (51), we change the name of variables i and to and j, respectively;
- and in (52), the order of summation over and j is reversed.

Moreover, Term_5 in (50) can be expanded as

$$\mathsf{Term}_5 = \Big(\bigoplus_{\in[K-1]\setminus\Gamma} F^{d()}_{(\{\}\cup\Gamma)\setminus\{d()\}} \Big) \oplus \Big(\bigoplus_{j \in [K]\setminus(\{\}\cup\Gamma)} F^{d()}_{(\{j\}\cup\Gamma)\setminus\{d()\}} \Big)$$

$$= \Big(\bigoplus_{\in[K-1]\setminus\Gamma} \bigoplus_{j \in [K]\setminus\Gamma} F^{d()}_{(\{j\}\cup\Gamma)\setminus\{d()\}} \Big) \tag{53}$$

$$= \Big(\bigoplus_{j \in [K]\setminus\Gamma} \bigoplus_{\in [K-1]\setminus\Gamma} F^{d()}_{(\{j\}\cup\Gamma)\setminus\{d()\}} \Big) \tag{54}$$

$$= \Big(\bigoplus_{j \in [K]\setminus\Gamma} \bigoplus_{\in [K-1]\setminus\Gamma} F^{d()}_{(\{j\}\cup\Gamma)\setminus\{d()\}} \Big) \oplus \Big(\bigoplus_{\in [K-1]\setminus\Gamma} F^{d()}_{(\{K\}\cup\Gamma)\setminus\{d()\}} \Big), \tag{55}$$

where

- in (53), the inner summations over $j =$ and $j \in [K]\setminus(\{\}\cup\Gamma)$ are merged into a single summation over $j \in [K] \setminus \Gamma$;

$$X_{\{\}\cup\Gamma} = \Big(\bigoplus_{\in[K-1]\setminus\Gamma} \bigoplus_{i \in \{\}\cup\Gamma} F^{i}_{(\{\}\cup\Gamma)\setminus\{i\}} \oplus F^{d(i)}_{(\{\}\cup\Gamma)\setminus\{d(i)\}} \oplus \bigoplus_{j \in [K]\setminus(\{\}\cup\Gamma)} F^{d(i)}_{(\{j\}\cup\{\}\cup\Gamma)\setminus\{i,d(i)\}} \Big)$$

$$= \Big(\bigoplus_{\in[K-1]\setminus\Gamma} \bigoplus_{i \in \Gamma} F^{i}_{(\{\}\cup\Gamma)\setminus\{i\}} \oplus F^{d(i)}_{(\{\}\cup\Gamma)\setminus\{d(i)\}} \oplus \bigoplus_{j \in [K]\setminus(\{\}\cup\Gamma)} F^{d(i)}_{(\{j,\}\cup\Gamma)\setminus\{i,d(i)\}} \Big)$$

$$\oplus \Big(\bigoplus_{\in[K-1]\setminus\Gamma} \bigoplus_{i \in \{\}} F^{i}_{(\{\}\cup\Gamma)\setminus\{i\}} \oplus F^{d(i)}_{(\{\}\cup\Gamma)\setminus\{d(i)\}} \oplus \bigoplus_{j \in [K]\setminus(\{\}\cup\Gamma)} F^{d(i)}_{(\{j,\}\cup\Gamma)\setminus\{i,d(i)\}} \Big) \tag{48}$$

$$= \Big(\bigoplus_{i \in \Gamma} \bigoplus_{\in[K-1]\setminus\Gamma} F^{i}_{(\{\}\cup\Gamma)\setminus\{i\}} \oplus F^{d(i)}_{(\{\}\cup\Gamma)\setminus\{d(i)\}} \oplus \bigoplus_{j \in [K]\setminus(\{\}\cup\Gamma)} F^{d(i)}_{(\{j,\}\cup\Gamma)\setminus\{i,d(i)\}} \Big)$$

$$\oplus \Big(\bigoplus_{\in[K-1]\setminus\Gamma} F_{(\{\}\cup\Gamma)\setminus\{\}} \oplus F^{d()}_{(\{\}\cup\Gamma)\setminus\{d()\}} \oplus \bigoplus_{j \in [K]\setminus(\{\}\cup\Gamma)} F^{d()}_{(\{j,\}\cup\Gamma)\setminus\{,d()\}} \Big) \tag{49}$$

$$= \underbrace{\Big(\bigoplus_{i \in \Gamma} \bigoplus_{\in[K-1]\setminus\Gamma} F^{i}_{(\{\}\cup\Gamma)\setminus\{i\}} \Big)}_{\mathsf{Term}_1} \oplus \underbrace{\Big(\bigoplus_{i \in \Gamma} \bigoplus_{\in[K-1]\setminus\Gamma} F^{d(i)}_{(\{\}\cup\Gamma)\setminus\{d(i)\}} \Big)}_{\mathsf{Term}_2} \oplus \underbrace{\Big(\bigoplus_{i \in \Gamma} \bigoplus_{\in[K-1]\setminus\Gamma} \bigoplus_{j \in [K]\setminus(\{\}\cup\Gamma)} F^{d(i)}_{(\{j,\}\cup\Gamma)\setminus\{i,d(i)\}} \Big)}_{\mathsf{Term}_3}$$

$$\oplus \Big(\underbrace{\bigoplus_{\in[K-1]\setminus\Gamma} F_{\Gamma}}_{\mathsf{Term}_4} \oplus \underbrace{\Big(\bigoplus_{\in[K-1]\setminus\Gamma} F^{d()}_{(\{\}\cup\Gamma)\setminus\{d()\}} \oplus \bigoplus_{j \in [K]\setminus(\{\}\cup\Gamma)} F^{d()}_{(\{j\}\cup\Gamma)\setminus\{d()\}} \Big)}_{\mathsf{Term}_5} \Big) \tag{50}$$

- in (54), the order of summation over and j is reversed;
- and finally in (55), the summation over j is broken into $j \in [K-1] \setminus \Gamma$ and $j = K$.

Combining Term_2 and Term_5, we obtain (62), shown at the bottom of this page. Note that

- in (56), shown at the bottom of this page, Term_2 with summation over $\in \Gamma$ and the first parentheses of Term_5 with summation over $\in [K-1] \setminus \Gamma$ are merged to a single summation over $\in [K-1]$;
- in (57), shown at the bottom of this page, the term $F^{d()}_{(\{j\}\cup\Gamma)\setminus\{d()\}}$ is added and subtracted, $j\in[K-1]\setminus\Gamma \quad =K$ i.e., the term is XORed twice;
- in (58), shown at the bottom of this page, we use the fact that $d(\cdot)$ is a bijective map over $[K]$, and hence replaced summation over $\in [K]$ with another summation over $u \in [K]$ where $u = d()$;
- in (59), shown at the bottom of this page, we know that $|\Gamma| = S-1$ and $j \notin \Gamma$. Hence, $|(\{j\} \cup \Gamma) \setminus \{u\}| = S-1$ if and only if $u \in \{j\} \cup \Gamma$. Similarly, $|(\{j\} \cup \Gamma) \setminus \{d(K)\}| = S-1$ if and only if $d(K) \in \{j\} \cup \Gamma$. Moreover, we know that $d(K) \notin \Gamma$, since we assume $F^{d(K)}_{\Gamma}$ is a non-trivial subfile requested by W_K at iteration $t+1$. Hence, the only valid choice for $d(K)$ is $d(K) = j$. After all simplifications, we get the expression in (59);
- in (60), shown at the bottom of this page, we break the summation over $u \in \{j\} \cup \Gamma$ into two summations, one over $u \in \Gamma$ and the other over $u = j$;
- and finally in (61), shown at the bottom of this page, the order of the two summations of the first term is reversed in order to identify expressions similar to Term_1 and Term_4 in (50).

It should be noted that every non-zero subfile that appears in ζ_2 in (62), shown at the bottom of this page, is either a subfile of F^K (for satisfying $d() = K$) or a subfile indexed by set $(\{K\} \cup \Gamma) \setminus \{d()\}$ that includes K. Both groups of such subfiles are cached at worker node W_K by definition of Z_K in (11). Therefore, ζ_2 can be completely recovered from Z_K.

Furthermore, for each $i \in \Gamma$, the inner term in Term_3 can be rewritten as

$$\begin{aligned}
&\in[K-1]\setminus\Gamma \quad \Big(\underset{j\in[K]\setminus(\{j\}\cup\Gamma)}{\oplus} F^{d(i)}_{(\{j,\}\cup\Gamma)\setminus\{i,d(i)\}} \Big) \\
&= \Big(\underset{\in[K-1]\setminus\Gamma \quad j\in[K-1]\setminus(\{j\}\cup\Gamma)}{\oplus} F^{d(i)}_{(\{j,\}\cup\Gamma)\setminus\{i,d(i)\}} \Big) \\
&\quad \underbrace{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}_{=0} \\
&\quad \oplus \Big(\underbrace{\underset{\in[K-1]\setminus\Gamma}{\oplus} F^{d(i)}_{(\{K,\}\cup\Gamma)\setminus\{i,d(i)\}}}_{\zeta_3(i)} \Big),
\end{aligned} \tag{63}$$

where the first term is zero since every subfile $F^{d(i)}_{(\{a,b\}\cup\Gamma)\setminus\{i,d(i)\}}$ appears exactly twice in the summation: once for $(= a, j = b)$ and another time for $(= b, j = a)$, where $a = b$. Hence, their contributions will be canceled when they are XORed. Moreover, we can show that all the subfiles appearing in $\zeta_3(i)$ are already cached in Z_K, and can be recovered by W_K. To see this, recall that $i \in \Gamma$ and $K \notin \Gamma$, which imply $i = K$. Therefore, either $d(i) = K$, or subscript $(\{K,\}\cup\Gamma)\setminus\{i, d(i)\}$ includes K. In the former case we have

$$\mathsf{Term}_2 \oplus \mathsf{Term}_5 = \Big(\underset{j\in[K-1]\setminus\Gamma \quad \in[K-1]}{\oplus} F^{d()}_{(\{j\}\cup\Gamma)\setminus\{d()\}} \Big) \oplus \underbrace{\Big(\underset{\in[K-1]\setminus\Gamma}{\oplus} F^{d()}_{(\{K\}\cup\Gamma)\setminus\{d()\}} \Big)}_{\zeta_2} \tag{56}$$

$$= \Big(\underset{j\in[K-1]\setminus\Gamma \quad \in[K]}{\oplus} F^{d()}_{(\{j\}\cup\Gamma)\setminus\{d()\}} \Big) \oplus \Big(\underset{j\in[K-1]\setminus\Gamma}{\oplus} F^{d(K)}_{(\{j\}\cup\Gamma)\setminus\{d(K)\}} \Big) \oplus \zeta_2 \tag{57}$$

$$= \Big(\underset{j\in[K-1]\setminus\Gamma \quad u\in[K]}{\oplus} \underbrace{F^{u}_{(\{j\}\cup\Gamma)\setminus\{u\}}}_{=0 \text{ if } u\in\{j\}\cup\Gamma} \Big) \oplus \Big(\underset{j\in[K-1]\setminus\Gamma}{\oplus} \underbrace{F^{d(K)}_{(\{j\}\cup\Gamma)\setminus\{d(K)\}}}_{=0 \text{ if } d(K)\notin\{j\}\cup\Gamma} \Big) \oplus \zeta_2 \tag{58}$$

$$= \Big(\underset{j\in[K-1]\setminus\Gamma \quad u\in\{j\}\cup\Gamma}{\oplus} F^{u}_{(\{j\}\cup\Gamma)\setminus\{u\}} \Big) \oplus F^{d(K)}_{\Gamma} \oplus \zeta_2 \tag{59}$$

$$= \Big(\underset{j\in[K-1]\setminus\Gamma \quad u\in\Gamma}{\oplus} F^{u}_{(\{j\}\cup\Gamma)\setminus\{u\}} \Big) \oplus \Big(\underset{j\in[K-1]\setminus\Gamma}{\oplus} F^{j}_{\Gamma} \Big) \oplus F^{d(K)}_{\Gamma} \oplus \zeta_2 \tag{60}$$

$$= \Big(\underset{u\in\Gamma \quad j\in[K-1]\setminus\Gamma}{\oplus} F^{u}_{(\{j\}\cup\Gamma)\setminus\{u\}} \Big) \oplus \Big(\underset{j\in[K-1]\setminus\Gamma}{\oplus} F^{j}_{\Gamma} \Big) \oplus F^{d(K)}_{\Gamma} \oplus \zeta_2 \tag{61}$$

$$= \mathsf{Term}_1 \oplus \mathsf{Term}_4 \oplus F^{d(K)}_{\Gamma} \oplus \zeta_2 \tag{62}$$

$F^{d(i)}_{(\{K,\}\cup\Gamma)\setminus\{i,d(i)\}} = F^{K}_{(\{K,\}\cup\Gamma)\setminus\{i,d(i)\}} \in P^K \subseteq Z_K$, while in the latter case we have $F^{d(i)}_{(\{K,\}\cup\Gamma)\setminus\{i,d(i)\}} \in E_K \subset Z_K$.

Plugging (62) and (63) into (50), we get

$$\begin{aligned} & X_{\{\}\cup\Gamma} \\ &\in[K-1]\setminus\Gamma \\ &= (\text{Term}_2 \oplus \text{Term}_5) \oplus (\text{Term}_1 \oplus \text{Term}_4) \oplus \text{Term}_3 \\ &= \text{Term}_1 \oplus \text{Term}_4 \oplus F^{d(K)}_{\Gamma} \oplus \zeta_2 \\ &\quad \oplus (\text{Term}_1 \oplus \text{Term}_4) \oplus \zeta_3(i) \\ &\quad \& \quad , \quad i\in\Gamma \\ &= \zeta_2 \oplus \zeta_3(i) \oplus F^{d(K)}_{\Gamma}, \end{aligned}$$

where $\zeta_2 \oplus (\zeta_3(i)$ can be reconstructed from Z_K. Consequently, the subfile $F^{d(K)}_{\Gamma}$ can be recovered from the cache content Z_K and $(X_{\{\}\cup\Gamma}$. This completes the proof of Lemma 2.

Appendix C
Proof of Lemma 3

We first define some notation. Without loss of generality, we assume $u(i) = i$ for $i \in [K]$. Let us partition W, the set of K worker nodes, into γ disjoint subsets W_i according to the cycles of the file transition graph, where W_i is the set of worker nodes that belong to cycle i, for $i \in [\gamma]$. Hence, we have

$$W = \bigcup_{i=1}^{\gamma} W_i, \tag{64}$$

where

$$W_i \cap W_j = \varnothing, \quad \forall i = j, \quad i, j \in [\gamma]. \tag{65}$$

In what follows, without loss of generality, let us consider the first $\gamma - 1$ cycles, and designate the γth cycle as the *ignored cycle*. Moreover, assume that the ignored worker node W_K belongs to the ignored cycle γ. Let $\gamma - 1 \geq S$, and consider an arbitrary subset Ψ of S cycles, that is $\Psi \subseteq [\gamma - 1]$ and $|\Psi| = S$. We define $W^{\otimes\Psi}$ to be the Cartesian product of the corresponding S sets of worker nodes, that is defined as

$$W^{\otimes\Psi} = \bigtimes_{i\in\Psi} W_i = \{\Delta \subseteq [K] : \Delta \cap W_i = 1, \forall i \in \Psi\},$$

for every $\Psi \subseteq [\gamma-1]$ with $|\Psi| = S$. Note that each element set in $W^{\otimes\Psi}$ consists of a tuple of S worker nodes, each belongs to a different cycle.

Recall that each sub-message X_Δ, defined in (16), is designed for a subset of S worker nodes, i.e., $\Delta \subseteq [K-1]$ and $|\Delta| = S$. The sub-message X_Δ is given by

$$X_\Delta = \Big(\bigoplus_{i\in\Delta} F^{i}_{\Delta\setminus\{i\}} \oplus F^{d(i)}_{\Delta\setminus\{d(i)\}} \oplus \bigoplus_{j\in[K]\setminus\Delta} F^{d(i)}_{(\{j\}\cup\Delta)\setminus\{i,d(i)\}} \Big). \tag{66}$$

In the following, we consider some $\Delta \in W^{\otimes\Psi}$, and expand the corresponding sub-message X_Δ. Note that since $\Delta \in W^{\otimes\Psi}$, it has only one worker node from each cycle. hence, for any $i \in \Delta$, we have either $d(i) \notin \Delta$ or $d(i) = i$ (because otherwise W_i and $W_{d(i)}$ will be two distinct and consecutive worker nodes that belong to the same cycle). This allows us to expand X_Δ as (71), given at the top of the next page. Note that

- in (67), shown at the top of the next page, the first summation over $i \in \Delta$ in (66) is split into $(i \in \Delta, d(i) \notin \Delta)$ and $(i \in \Delta, d(i) = i)$;
- in (68), shown at the top of the next page, we have used the fact that if $i, d(i) \in \Delta$ then $i = d(i)$ in the forth and sixth summations;
- in (69), shown at the top of the next page, we have canceled out the identical quantities in the second and the forth summations of (68). Moreover, in the third term of (68), each subfile is indexed by $\Delta \setminus \{d(i)\}$, while $d(i) \notin \Delta$. Note that $|\Delta \setminus \{d(i)\}| = S = S - 1$, and thus such subfiles are defined to be zero. This implies that the third summation is zero. Similarly, the sixth summation in (68) is zero, due the fact that $|(\{j\} \cup \Delta) \setminus \{i\}| = 1 + S - 1 = S > S - 1$, and hence all the subfiles in the sixth summation are zero. Therefore, only the first and the fifth terms of (68) survive;
- in (70), shown at the top of the next page, the summation over j is split into $j \in [K] \setminus \Delta, j = d(i)$ and $j = d(i)$;
- and finally, the second term in (70) is set to zero in (71) shown at the top of the next page. This is due to the fact that for $d(i) \notin \Delta$ and $d(i) = j$, we have $|(\{j\} \cup \Delta) \setminus \{i, d(i)\}| = |(\{j\} \cup \Delta) \setminus \{i\}| = S > S - 1$, and hence, $F^{d(i)}_{(\{j\}\cup\Delta)\setminus\{i,d(i)\}} = 0$.

Next, we prove that there exists one linearly dependent sub-message in the set of sub-messages $\{X_\Delta : \Delta \in W^{\otimes\Psi}\}$ for each group of S cycles, determined by $\Psi \subseteq [\gamma - 1]$. More precisely, we claim that

$$\begin{aligned} \bigoplus_{\Delta\in W^{\otimes\Psi}} X_\Delta &= \bigoplus_{\Delta\in W^{\otimes\Psi}} \Big(\Big(\bigoplus_{\substack{i\in\Delta \\ d(i)=i}} F^{i}_{\Delta\setminus\{i\}} \Big) \oplus \Big(\bigoplus_{\substack{i\in\Delta \\ d(i)\notin\Delta}} F^{d(i)}_{\Delta\setminus\{i\}} \Big) \Big) \\ &= \bigoplus_{\Gamma\in W^{\otimes\Psi}} \Big(\bigoplus_{\substack{j\in\Gamma \\ d(j)=j}} F^{j}_{\Gamma\setminus\{j\}} \Big) \oplus \bigoplus_{\Delta\in W^{\otimes}_{\Psi}} \Big(\bigoplus_{\substack{i\in\Delta \\ d(i)\notin\Delta}} F^{d(i)}_{\Delta\setminus\{i\}} \Big) \\ &= 0. \end{aligned} \tag{72}$$

We prove (72) by showing that each subfile appears exactly twice in the XOR equation which immediately yields that the summation in (72) equals to zero. To this end, consider a pair of (Δ, i) with $\Delta \in W^{\otimes\Psi}$, $i \in \Delta$ and $d(i) = i$. Therefore, we have a subfile $F^{d(i)}_{\Delta\setminus\{i\}}$ that appears in the second summation in (72). Now, define $j = d(i)$ and $\Gamma = (\Delta \cup \{j\}) \setminus \{i\}$. It is clear that i and $d(i)$ belong to the same cycle, and hence $\Gamma \in W^{\otimes\Psi}$. On the other hand, since $d(\cdot)$ is a bijective map, the fact that $d(i) = i$ implies that $d(d(i)) = d(i)$, or $d(j) = j$. Therefore, the pair (Γ, j) satisfies the three conditions in the first summation, namely $\Gamma \in W^{\otimes\Psi}$, $j \in \Gamma$, and $d(j) = j$. Hence, the corresponding term $F^{j}_{\Gamma\setminus\{j\}}$ appears in the second summation. However, by plugging in $j = d(i)$ and $\Gamma = (\Delta \cup \{j\}) \setminus \{i\}$, we get $F^{j}_{\Gamma\setminus\{j\}} = F^{d(i)}_{\Delta\setminus\{i\}}$. This shows that the terms $F^{j}_{\Gamma\setminus\{j\}}$ in the first summation and $F^{d(i)}_{\Delta\setminus\{i\}}$ in the second summation cancel out each other in the entire summation. This applies to each term, and shows that (72) holds.

$$X_\Delta = \Big(\bigoplus_{\substack{i\in\Delta\\ d(i)\notin\Delta}} F^i_{\Delta\setminus\{i\}}\Big) \oplus \Big(\bigoplus_{\substack{i\in\Delta\\ d(i)=i}} F^i_{\Delta\setminus\{i\}}\Big) \oplus \Big(\bigoplus_{\substack{i\in\Delta\\ d(i)\notin\Delta}} F^{d(i)}_{\Delta\setminus\{d(i)\}}\Big) \oplus \Big(\bigoplus_{\substack{i\in\Delta\\ d(i)=i}} F^{d(i)}_{\Delta\setminus\{d(i)\}}\Big)$$
$$\oplus \Big(\bigoplus_{\substack{i\in\Delta\\ d(i)\notin\Delta}} \bigoplus_{j\in[K]\setminus\Delta} F^{d(i)}_{(\{j\}\cup\Delta)\setminus\{i,d(i)\}}\Big) \oplus \Big(\bigoplus_{\substack{i\in\Delta\\ d(i)=i}} \bigoplus_{j\in[K]\setminus\Delta} F^{d(i)}_{(\{j\}\cup\Delta)\setminus\{i,d(i)\}}\Big) \tag{67}$$

$$= \Big(\bigoplus_{\substack{i\in\Delta\\ d(i)\notin\Delta}} F^i_{\Delta\setminus\{i\}}\Big) \oplus \Big(\bigoplus_{\substack{i\in\Delta\\ d(i)=i}} F^i_{\Delta\setminus\{i\}}\Big) \oplus \Big(\bigoplus_{\substack{i\in\Delta\\ d(i)\notin\Delta}} F^{d(i)}_{\Delta\setminus\{d(i)\}}\Big) \oplus \Big(\bigoplus_{\substack{i\in\Delta\\ d(i)=i}} F^i_{\Delta\setminus\{i\}}\Big)$$
$$\oplus \Big(\bigoplus_{\substack{i\in\Delta\\ d(i)\notin\Delta}} \bigoplus_{j\in[K]\setminus\Delta} F^{d(i)}_{(\{j\}\cup\Delta)\setminus\{i,d(i)\}}\Big) \oplus \Big(\bigoplus_{\substack{i\in\Delta\\ d(i)=i}} \bigoplus_{j\in[K]\setminus\Delta} F^{i}_{(\{j\}\cup\Delta)\setminus\{i\}}\Big) \tag{68}$$

$$= \Big(\bigoplus_{\substack{i\in\Delta\\ d(i)\notin\Delta}} F^i_{\Delta\setminus\{i\}}\Big) \oplus \Big(\bigoplus_{\substack{i\in\Delta\\ d(i)\notin\Delta}} \bigoplus_{j\in[K]\setminus\Delta} F^{d(i)}_{(\{j\}\cup\Delta)\setminus\{i,d(i)\}}\Big) \tag{69}$$

$$= \Big(\bigoplus_{\substack{i\in\Delta\\ d(i)\notin\Delta}} F^i_{\Delta\setminus\{i\}}\Big) \oplus \Big(\bigoplus_{\substack{i\in\Delta\\ d(i)\notin\Delta}} \bigoplus_{\substack{j\in[K]\setminus\Delta\\ j\neq d(i)}} F^{d(i)}_{(\{j\}\cup\Delta)\setminus\{i,d(i)\}}\Big) \oplus \Big(\bigoplus_{\substack{i\in\Delta\\ d(i)\notin\Delta}} F^{d(i)}_{(\{j\}\cup\Delta)\setminus\{i,d(i)\}}\Big) \tag{70}$$

$$= \Big(\bigoplus_{\substack{i\in\Delta\\ d(i)\notin\Delta}} F^i_{\Delta\setminus\{i\}}\Big) \oplus \Big(\bigoplus_{\substack{i\in\Delta\\ d(i)\notin\Delta}} F^{d(i)}_{\Delta\setminus\{i\}}\Big) \tag{71}$$

As a result, there is (at least) one linearly dependent sub-message in the set of sub-messages $\{X_\Delta : \Delta \in W_\Psi^{\otimes}\}$, and any of these sub-messages can be reconstructed by summing the others. Therefore, we can refrain from sending one of them in the delivery phase. Moreover, the groups of sub-messages $\{X_\Delta : \Delta \in W_\Psi^{\otimes}\}$ are disjoint for different groups of cycles Ψ. Since there are $\binom{\gamma-1}{S}$ such groups of cycles, therefore one can refrain from sending $\binom{\gamma-1}{S}$ redundant sub-messages out of the $\binom{K-1}{S}$ sub-messages that the master node should broadcast to the worker nodes (according to the proposed delivery scheme in Section IV-A). Consequently, the delivery load given by (3) is achievable. This completes the proof of Lemma 3.

Appendix D
Proof of Lemma 4

We want to prove that the broadcast message X and the cache contents of the virtual worker node Z enable us to perfectly recover all the files of the shuffling system. First, note that according to the one-to-one positional labeling, we have

$$\{F^i : i \in \{1, 2, \ldots, K\}\} \equiv \{F^{(c,p)} : c \in \{1, 2, \ldots, \gamma\}, p \in \{1, 2, \ldots, \ell_c\}\}.$$

Hence, in order to prove the lemma, it suffices to show that

$$H\left(\{\{F^{(c,p)}\}_{p=1}^{\ell_c}\}_{c=1}^{\gamma} \,|\, X, Z\right) = 0.$$

To this end, we consider two cases: $p = 1$ and $p > 1$. For $p = 1$, from (27) we simply have $F^{(c,1)} \subseteq Z$ for each $c \in [\gamma]$, which implies $H\left(\{F^{(c,1)}\}_{c=1}^{\gamma} \,|\, X, Z\right) \leq H\left(\{F^{(c,1)}\}_{c=1}^{\gamma} \,|\, Z\right) = 0$. Therefore, we have

$$\begin{aligned} &H\left(\{\{F^{(c,p)}\}_{p=1}^{\ell_c}\}_{c=1}^{\gamma} \,|\, X, Z\right) \\ &= H\left(\{\{F^{(c,p)}\}_{p=2}^{\ell_c}\}_{c=1}^{\gamma} \{F^{(x,1)}\}_{x=1}^{\gamma}, X, Z\right) \\ &= \sum_{c=1}^{\gamma}\sum_{p=1}^{\ell_c-1} H\left(F^{(c,p+1)} \,|\, \{F^{(c',p')}\}_{(c',p')\neq(c,p)}, \{F^{(x,1)}\}_{x=1}^{\gamma}, X, Z\right), \end{aligned} \tag{73}$$

where we have used the chain rule in the last equation. Recall from (25) that for any pair (c, p) with $p < \ell_c$, we have

$$\begin{aligned} Z_{(c,p)} &= F^{(c,p)} \cup \Big(\bigcup_{(c',p')=(c,p)} F^{(c',p')}_{(c,p)}\Big) \\ &= F^{(c,p)} \cup \underbrace{\Big(\bigcup_{(c',p')<(c,p)} F^{(c',p')}_{(c,p)}\Big)}_{\subseteq\, F^{(c',p')}_{(c,p)}} \cup \Big(\bigcup_{c\in[\gamma]} F^{(c,1)}_{(c,p)}\Big) \cup \underbrace{\Big(\bigcup_{\substack{(c',p')<(c,p)\\ p>1}} F^{(c',p')}_{(c,p)}\Big)}_{\subseteq Z}. \end{aligned} \tag{74}$$

Therefore, each term in (73) can be bounded by

$$\begin{aligned} H\left(F^{(c,p+1)} \mid \{F^{(c',p')}\}_{(c',p')\prec(c,p)}, X, Z\right) &\le H\left(F^{(c,p+1)}, Z_{(c,p)} \mid \{F^{(c',p')}\}_{(c',p')\prec(c,p)}, X, Z\right) \\ &\le H\left(Z_{(c,p)} \mid \{F^{(c',p')}\}_{(c',p')\prec(c,p)}, Z\right) \\ &\quad + H\left(F^{(c,p+1)} \mid Z_{(c,p)}, X\right) = 0, \end{aligned} \tag{75}$$

where the first term in (75) is zero due to (74), and the second term is zero due to the fact that the worker node at position (c, p) should be able to recover its assigned file $F^{(c,p+1)}$ from its cache $Z_{(c,p)}$ and the broadcast message X. Plugging (75) into (73), we conclude the claim of Lemma 4.

Appendix E
Proof of Lemma 5

The next step in the converse proof is to provide a lower bound on the communication load for a given instance of the shuffling problem as follows.

Consider a directed file transition graph $G(V, E)$, that characterizes one instance of the problem. We can start with

$$\begin{aligned} K = H\left(\{F^i\}_{i=1}^K\right) &\le H\left(\{F^i\}_{i=1}^K, X, Z\right) \\ &= H(X, Z) + H\left(\{F^i\}_{i=1}^K \mid X, Z\right) \\ &= H\left(X, Z, \left(F^{(c,p)}_{(c',p')}\right)_{(c',p')\prec(c,p)}\right) \qquad (76) \\ &= H\left(\left(X, \left(F^{(c,1)}\right)_{c\in[\gamma]}, \left(F^{(c,p)}_{(c',p')}\right)_{(c',p')\prec(c,p)}\right)_{c\in[\gamma],p>1}\right) \qquad (77) \\ &\le H(X) + \sum_{c=1}^{\gamma} H(F^{(c,1)}) + \sum_{c=1}^{\gamma}\sum_{p=2}^{\alpha_c} H\left(F^{(c,p)}_{(c',p')}\right)_{(c',p')\prec(c,p)} \\ &= R(d) + \gamma + \sum_{c=1}^{\gamma}\sum_{p=2}^{\alpha_c} \mu^{(c,p)}_{\Phi(c,p)}, \qquad (78) \end{aligned}$$

where $\Phi(c, p) = \{(c', p') : (c', p') \prec (c, p), p' < \alpha_{c'}\}$. Note that (76) holds due to Lemma 4, and in (77) we use the definition of Z in (27). This implies a lower bound on the communication load R that is given by

$$R(d) \ge K - \gamma - \sum_{c=1}^{\gamma}\sum_{p=2}^{\alpha_c} \mu^{(c,p)}_{\Phi(c,p)}. \tag{79}$$

A similar argument holds for any instance of the problem, characterized by an assignment function $d(\cdot)$ whose file transition graph is isomorphic to $G(V, E)$. Let $V = \{(c, p) : c \in [\gamma], p \in [\alpha_c]\}$ be the set of vertices of the graph, and $\Pi : V \to V$ be the set of all possible permutations on the vertices of the graph. For each $\pi \in \Pi$, we have an instance of the shuffling problem characterized by the same file transition graph $G(V, E)$, in which node W_i which was at position (c, p) in the original problem, is now positioned at $\pi(c, p)$. For the new instance of the problem, we have

$$Z_{\pi(c,p)} = \left(F^{\pi(c,p)}, \left(F^{\pi(c',p')}_{\pi(c,p)}\right)_{(c',p')=(c,p)}\right).$$

Hence, following an argument similar to that of (79), we obtain

$$\begin{aligned} R(G) &= \frac{1}{|D_G|}\sum_{d\in D_G} R(d) \\ &\ge K - \gamma - \sum_{c=1}^{\gamma}\sum_{p=2}^{\alpha_c} \mu^{\pi(c,p)}_{\pi(\Phi(c,p))}, \end{aligned} \tag{80}$$

where $\pi(\Phi(c, p)) = \{\pi(c', p') : (c', p') \in \Phi(c, p)\}$. Next, by averaging (80) over all $\pi \in \Pi$, we get

$$R(G) \ge K - \gamma - \frac{1}{K!}\sum_{\pi\in\Pi}\sum_{c=1}^{\gamma}\sum_{p=2}^{\alpha_c} \mu^{\pi(c,p)}_{\pi(\Phi(c,p))}, \tag{81}$$

where

$$\begin{aligned} &\frac{1}{K!}\sum_{\pi\in\Pi}\sum_{c=1}^{\gamma}\sum_{p=2}^{\alpha_c} \mu^{\pi(c,p)}_{\pi(\Phi(c,p))} \\ &= \frac{1}{K!}\sum_{c=1}^{\gamma}\sum_{p=2}^{\alpha_c}\sum_{k\in V}\sum_{\substack{A:A\subseteq V\setminus\{k\}\\|A|=|\Phi(c,p)|}}\sum_{\substack{\pi\in\Pi:\pi(c,p)=k\\ \pi(\Phi(c,p))=A}} \mu_A^k \\ &= \frac{1}{K!}\sum_{c=1}^{\gamma}\sum_{p=2}^{\alpha_c}\sum_{k\in[K]}\sum_{\substack{A:A\subseteq V\setminus\{k\}\\|A|=|\Phi(c,p)|}} 1!|A|!(K-|A|-1)!\mu_A^k \qquad (82) \\ &= \frac{1}{K}\binom{K-1}{|A|}^{-1}\sum_{c=1}^{\gamma}\sum_{p=2}^{\alpha_c}\sum_{k\in[K]}\sum_{\substack{A:A\subseteq V\setminus\{k\}\\|A|=|\Phi(c,p)|}} \mu_A^k \\ &= \sum_{c=1}^{\gamma}\sum_{p=2}^{\alpha_c} \mu_{|\Phi(c,p)|}, \qquad (83) \end{aligned}$$

where in (82) the innermost summation is evaluated by counting the number of permutations π that satisfy $\pi(c, p) = k$ and $\pi(\Phi(c, p)) = A$. More precisely, there are $|A|!$ ways to map entries of $\Phi(c, p)$ to A, and there is only one way to map (c, p) to k. There are a total of $K - |A| - 1$ remaining entries, which can be mapped in $(K - |A| - 1)!$ ways. Moreover, in (83) we have used the definition of μ_η in (29). Recall from the definition of $\Phi(c, p)$ that $|\Phi(c, p)| = \sum_{t=1}^{c-1} \alpha_t - 1 + (p-1)$. It is easy to see that $|\Phi(c, p)| = |\Phi(c', p')|$ for any distinct pair of (c, p) and (c', p'). Moreover, $|\Phi(c, p)| \ge 1$ (for $p \ge 1$) and $|\Phi(c, p)| \le K - \gamma$. These together imply that $\{|\Phi(c, p)| : c \in \{1, 2, \ldots, \gamma\}, p \in \{2, 3, \ldots, \alpha_c\}\} = \{1, 2, \ldots, K - \gamma\}$. In other words, for each integer i, there exists exactly one pair of (c, p) such that $|\Phi(c, p)| = i$. Hence, the RHS of (83) can be simplified to $\sum_{i=1}^{K-\gamma} \mu_i$. Plugging this into (81), we obtain

$$R(G) \ge K - \gamma - \sum_{i=1}^{K-\gamma} \mu_i. \tag{84}$$

This completes the proof of Lemma 5.

APPENDIX F
PROOF OF LEMMA 6

This appendix is dedicated to prove an upper bound on μ_α's in order to obtain a lower bound on $R(G)$. Note that variables $\{\mu_\alpha\}_{\alpha=0}^{K-1}$ defined in (29) can be generalized to any arbitrary family of sets.

Consider an arbitrary family of sets $\{A_1, A_2, \ldots, A_{K-1}\}$, and define

$$\theta_\alpha = \frac{1}{\binom{K-1}{\alpha}} \sum_{\substack{J \subseteq [K-1] \\ |J|=\alpha}} |A_J|,$$

where $A_J = \bigcap_{j \in J} A_j$.

We need to derive some preliminary results in order to prove the lemma. The proofs of these claims are presented at the end of this appendix. The first lemma below establishes a core inequality on linear combinations of θ_α's.

Lemma 8: For any family of sets $\{A_1, A_2, \ldots, A_{K-1}\}$, and an integer $T \in \{0, 1, 2, \ldots, K-2\}$, we have

$$\sum_{j=T}^{K-1} (-1)^{j-T} \binom{K-1-T}{j-T} \theta_j \le 0. \tag{85}$$

The following propositions will be used later in the proof of Lemma 6:

Proposition 1: For given integers $0 \le q \le p$ and any sequence of real numbers $(\theta_0, \theta_2, \ldots, \theta_p)$, we have

$$\sum_{x=0}^{p} \sum_{y=0}^{p} (-1)^{x-y} \binom{p-q}{x-y,\ p-x,\ y-q} \theta_x = \theta_q. \tag{86}$$

Proposition 2: For given $T, \alpha, \beta \in \mathbb{Z}^+$, we have

$$(\alpha - 1)\binom{T}{\alpha} - \beta \binom{T-1}{\alpha-1} \ge -\binom{\beta}{\alpha}. \tag{87}$$

For $\alpha \in \{1, 2, \ldots, K-1\}$ and $\beta \in \{0, 1, \ldots, K-1\}$, we define $V_{\alpha,\beta}$ as

$$V_{\alpha,\beta} \binom{K-\alpha-1}{K-\beta-1} + \frac{\sum_{S-1}^{K-\alpha-1}\binom{K-1}{K-\beta-1}}{\binom{K-1}{S-1}}(\alpha-1) - \frac{\alpha\beta}{K-S}. \tag{88}$$

The following proposition shows that $V_{\alpha,\beta}$ is non-negative for any choice of α and β:

Proposition 3: For $\alpha \in \{1, 2, \ldots, K-1\}$ and $\beta \in \{0, 1, \ldots, K-1\}$, we have $V_{\alpha,\beta} \ge 0$.

Now, we are ready to prove Lemma 6.

Proof of Lemma 6: We start from Lemma 8, and write

$$\sum_{i=\beta}^{K-1} (-1)^{i-\beta} \binom{K-\beta-1}{i-\beta} \theta_i \le 0, \quad \beta \in \{0, 1, \ldots, K-2\}.$$

Now, fix some $\alpha \in \{1, 2, \ldots, K-1\}$ and recall from Proposition 3 that $V_{\alpha,\beta} \ge 0$. Multiplying both sides of this inequality by non-negative coefficients $V_{\alpha,\beta}$ and summing over all values of β to obtain

$$\sum_{\beta=0}^{K-2} V_{\alpha,\beta} \sum_{i=\beta}^{K-1} (-1)^{i-\beta} \binom{K-\beta-1}{i-\beta} \theta_i \le \sum_{\beta=0}^{K-2} V_{\alpha,\beta} \times 0 = 0. \tag{89}$$

Moreover, when $\beta = K-1$, we have

$$V_{\alpha,K-1} \sum_{i=K-1}^{K-1} (-1)^{i-(K-1)} \binom{0}{i-(K-1)} \theta_i = V_{\alpha,K-1}\ \theta_{K-1}. \tag{90}$$

Summing (89) and (90), we get (93), given at the top of the next page. Note that (91), shown at the top of the next page, holds because the binomial term is zero for $i < \beta$; and in (92), shown at the top of the next page, we used the definition of $V_{\alpha,\beta}$ given by (88). Each term in (93) can be simplified as follows. First, we can use Proposition 1 for $(p, q, x, y) = (K-1, \alpha, i, \beta)$ to get

$$\text{Term}_1 = \sum_{i=0}^{K-1}\sum_{\beta=0}^{K-1} (-1)^{i-\beta} \binom{K-1-\alpha}{i-\beta,\ K-1-i,\ \beta-\alpha} \theta_i = \theta_\alpha. \tag{94}$$

Similarly, setting $(p, q, x, y) = (K-1, 0, i, \beta)$ in Proposition 1, we obtain

$$\text{Term}_2 = \sum_{i=0}^{K-1}\sum_{\beta=0}^{K-1} (-1)^{i-\beta} \binom{K-1}{i-\beta,\ K-1-i,\ \beta} \theta_i = \theta_0 = 0. \tag{95}$$

Lastly, for Term_3, we have

$$\begin{aligned}
\text{Term}_3 &= \sum_{i=0}^{K-1}\sum_{\beta=0}^{K-1} (-1)^{i-\beta} \binom{K-\beta-1}{i-\beta} \binom{K-1}{\beta} \beta\theta_i \\
&= \sum_{i=0}^{K-1}\sum_{\beta=0}^{K-1} (-1)^{i-\beta} \binom{K-\beta-1}{i-\beta} \binom{K-2}{\beta-1} \frac{K-1}{\beta}\beta\theta_i \\
&= (K-1) \sum_{i=0}^{K-1}\sum_{\beta=0}^{K-1} (-1)^{i-\beta} \binom{K-2}{i-\beta,\ K-1-i,\ \beta-1} \theta_i \\
&= (K-1)\theta_1,
\end{aligned} \tag{96}$$

where (96) follows from Proposition 1 by setting $(p, q, x, y) = (K-1, 1, i, \beta)$. Plugging (94), (95) and (96) into (93), we obtain

$$\theta_\alpha - \frac{\alpha}{K-S}\frac{\sum_{S-1}^{K-\alpha-1}}{\binom{K-1}{S-1}}(K-1)\theta_1 \le V_{\alpha,K-1}\ \theta_{K-1},$$

$$V_{\alpha,K-1}\ \theta_{K-1}$$

$$\geq \sum_{\beta=0}^{K-1} V_{\alpha,\beta} \sum_{i=0}^{K-1} (-1)^{i-\beta} \binom{K-\beta-1}{i-\beta} \theta_i \tag{91}$$

$$= \sum_{i=0}^{K-1}\sum_{\beta=0}^{K-1} (-1)^{i-\beta} \binom{K-\beta-1}{i-\beta} V_{\alpha,\beta}\ \theta_i$$

$$= \sum_{i=0}^{K-1}\sum_{\beta=0}^{K-1} (-1)^{i-\beta} \binom{K-\beta-1}{i-\beta}\binom{K-\alpha-1}{K-\beta-1} \theta_i + (\alpha-1)\frac{\binom{K-\alpha-1}{S-1}}{\binom{K-1}{S-1}} \sum_{i=0}^{K-1}\sum_{\beta=0}^{K-1} (-1)^{i-\beta} \binom{K-\beta-1}{i-\beta}\binom{K-1}{K-\beta-1} \theta_i$$

$$- \frac{\alpha}{K-S}\frac{\binom{K-\alpha-1}{S-1}}{\binom{K-1}{S-1}} \sum_{i=0}^{K-1}\sum_{\beta=0}^{K-1} (-1)^{i-\beta} \binom{K-\beta-1}{i-\beta}\binom{K-1}{K-\beta-1} \beta\theta_i \tag{92}$$

$$= \underbrace{\sum_{i=0}^{K-1}\sum_{\beta=0}^{K-1} (-1)^{i-\beta} \binom{K-1-\alpha}{i-\beta, K-1-i, \beta-\alpha} \theta_i}_{\text{Term}_1} + (\alpha-1)\frac{\binom{K-\alpha-1}{S-1}}{\binom{K-1}{S-1}} \underbrace{\sum_{i=0}^{K-1}\sum_{\beta=0}^{K-1} (-1)^{i-\beta} \binom{K-1}{i-\beta, K-1-i, \beta} \theta_i}_{\text{Term}_2}$$

$$- \frac{\alpha}{K-S}\frac{\binom{K-\alpha-1}{S-1}}{\binom{K-1}{S-1}} \underbrace{\sum_{i=0}^{K-1}\sum_{\beta=0}^{K-1} (-1)^{i-\beta} \binom{K-\beta-1}{i-\beta}\binom{K-1}{\beta} \beta\theta_i}_{\text{Term}_3} \tag{93}$$

or equivalently,

$$\frac{1}{\binom{K-1}{\alpha}} \sum_{J\subseteq[K-1],\,|J|=\alpha} \sum_{j\in J} A_j \leq \frac{\alpha(K-1)}{K-S}\frac{\binom{K-\alpha-1}{S-1}}{\binom{K-1}{S-1}}\frac{1}{\binom{K-1}{1}} \sum_{j\in[K-1]} |A_j| + V_{\alpha,K-1}\ \frac{1}{\binom{K-1}{K-1}} \sum_{j\in[K-1]} A_j\ . \tag{97}$$

It should be noted that (97) holds for any choice of $\{A_1, A_2, \ldots, A_K\}$. In particular, applying (97) to the family of sets $F_j^i : j \in [K]\setminus\{i\}$ for a fixed i, we get

$$\frac{1}{\binom{K-1}{\alpha}} \sum_{J\subseteq[K]\setminus\{i\},\,|J|=\alpha} \sum_{j\in J} F_j^i \leq \frac{\alpha(K-1)}{K-S}\frac{\binom{K-\alpha-1}{S-1}}{\binom{K-1}{S-1}}\frac{1}{\binom{K-1}{1}} \sum_{j\in[K]\setminus\{i\}} F_j^i + V_{\alpha,K-1}\ \frac{1}{\binom{K-1}{K-1}} \sum_{j\in[K]\setminus\{i\}} F_j^i\ , \tag{98}$$

for $i \in \{1, 2, \ldots, K\}$. Averaging (98) over $i \in \{1, 2, \ldots, K\}$, we obtain

$$\frac{1}{K\binom{K-1}{\alpha}} \sum_{i\in[K]} \sum_{J\subseteq[K]\setminus\{i\},\,|J|=\alpha} \sum_{j\in J} F_j^i \leq \frac{\alpha(K-1)}{K-S}\frac{\binom{K-\alpha-1}{S-1}}{\binom{K-1}{S-1}}\frac{1}{K\binom{K-1}{1}} \sum_{i\in[K]} \sum_{j\in[K]\setminus\{i\}} F_j^i + V_{\alpha,K-1}\ \frac{1}{K\binom{K-1}{K-1}} \sum_{i\in[K]} \sum_{j\in[K]\setminus\{i\}} F_j^i\ ,$$

or equivalently,

$$\mu_\alpha \leq \frac{\alpha(K-1)}{K-S}\frac{\binom{K-\alpha-1}{S-1}}{\binom{K-1}{S-1}}\mu_1 + V_{\alpha,K-1}\ \mu_{K-1}\ . \tag{99}$$

Hence, it remains to upper bound μ_1 and μ_{K-1}. Recall from definition of μ_i's in (29) that

$$\mu_1 = \frac{1}{K(K-1)} \sum_{i\in[K]} \sum_{j\in[K]\setminus\{i\}} F_j^i = \frac{1}{K(K-1)} \sum_{j\in[K]} \sum_{i\in[K]\setminus\{j\}} F_j^i = \frac{1}{K(K-1)} \sum_{j\in[K]} |Z_j \setminus P^j| \leq \frac{1}{K(K-1)} \sum_{j\in[K]} (S-1) = \frac{S-1}{K-1}, \tag{100}$$

and

$$\mu_{K-1} = \frac{1}{K} \sum_{i\in[K]} \sum_{j\in[K]\setminus\{i\}} F_j^i \leq \frac{1}{K} \sum_{i\in[K]} \min\left\{ \sum_{j\in[K]\setminus\{i\}} |F_j^i|,\ |F^i| \right\} = \frac{1}{K} \min\left\{ \frac{1}{K}\sum_{j\in[K]}\sum_{i\in[K]\setminus\{j\}} |F_j^i|,\ \frac{1}{K}\sum_{i\in[K]} 1 \right\} = \min\{S-1, 1\}. \tag{101}$$

Plugging (101) and (101) into (99), we get

$$\begin{aligned}\mu_\alpha &\le \frac{\alpha(K-1)}{K-S}\frac{\binom{K-\alpha-1}{S-1}}{\binom{K-1}{S-1}}\frac{S-1}{K-1} + V_{\alpha,K-1}\cdot 1\\ &= \frac{\binom{K-\alpha-1}{S-1}}{\binom{K-1}{S-1}}\frac{S-1}{K-S}\alpha\\ &\quad + \left(1+\frac{\binom{K-\alpha-1}{S-1}}{\binom{K-1}{S-1}}\right)(\alpha-1) - \frac{\alpha(K-1)}{K-S}\\ &= 1 + \frac{\binom{K-\alpha-1}{S-1}}{\binom{K-1}{S-1}}\frac{S-1}{K-S}\alpha - \frac{K-1}{K-S}\alpha + \alpha - 1\\ &= 1 - \frac{\binom{K-\alpha-1}{S-1}}{\binom{K-1}{S-1}}. \end{aligned} \tag{102}$$

This completes the proof of Lemma 6.

It remains to prove Lemma 8 and Propositions 1, 2, and 3, whose proofs are presented as follows.

Proof of Lemma 8: Fix a $T \subseteq [K-1]$ be a set of indices of size $|T| = T$. Define $X \triangleq \bigcap_{t\in T} A_t$. Clearly, we have $X \cap \bigcap_{t\in T} A_t \subseteq \bigcap_{t\in T} A_t$, which implies

$$X \cap \bigcap_{t\in T} A_t - \bigcap_{t\in T} A_t \le 0. \tag{103}$$

Each term in the LHS of (103) can be expanded using the inclusion-exclusion principle as follows:

$$\begin{aligned}\bigcap_{t\in T} A_t &= \sum_{U\subset T:\,|U|=1} |A_U| - \sum_{U\subset T:\,|U|=2} |A_U| + \sum_{U\subset T:\,|U|=3} |A_U| - \cdots\\ &\quad + (-1)^{|T|-1} \sum_{U\subset T:\,|U|=|T|} |A_U|. \end{aligned} \tag{104}$$

In a similar way, we have (105) as shown at the bottom of this page.

Substituting (104) and (105) in (103), we get

$$\begin{aligned}0 &\ge X \cap \bigcap_{t\in T} A_t - \bigcap_{t\in T} A_t\\ &= |X| - \sum_{U\subset T:\,|U|=1} |X\cup A_U| + \sum_{U\subset T:\,|U|=2} |X\cup A_U| - \cdots\\ &\quad + (-1)^{|T|} \sum_{U\subset T:\,|U|=|T^c|} |X\cup A_U|\\ &= |A_T| - \sum_{U\subset T:\,|U|=1} |A_T\cup A_U| + \sum_{U\subset T:\,|U|=2} |A_T\cup A_U| - \cdots\\ &\quad + (-1)^{|T|} \sum_{U\subset T:\,|U|=|T^c|} |A_T\cup A_U|. \end{aligned} \tag{106}$$

Note that (106) holds for any subset of indices T of size $|T| = T$. Averaging this inequality over all choices of $T \subseteq [K-1]$ with $|T| = T$, we obtain the chain of equations in (109), shown at the top of the next page. Note that in (107), shown at the top of the next page, we have replaced $A_T \cup A_U$ by $A_V = A_{T\cup U}$; in (108), shown at the top of the next page, equality holds since $\binom{x}{y}\binom{y}{z} = \binom{x}{z}\binom{x-z}{y-z}$ for any triple of integers (x, y, z); and in (109) we have $|T| = K-1-T$. Hence, (85) readily follows for all values of $T \in \{0, 1, \ldots, K-2\}$. This completes the proof of Lemma 8.

Proof of Proposition 1:

$$\begin{aligned}&\sum_{x=0}^{p}\sum_{y=0}^{p} (-1)^{x-y}\binom{p-q}{x-y, p-x, y-q}\theta_x\\ &= \sum_{x=0}^{p}\sum_{y=q}^{x} (-1)^{x-y}\binom{p-q}{p-x}\binom{x-q}{x-y}\theta_x \qquad (110)\\ &= \sum_{x=0}^{p}\sum_{z=0}^{x-q} (-1)^{z}\binom{x-q}{z}\binom{p-q}{p-x}\theta_x \qquad (111)\\ &= \sum_{x=0}^{p} (1-1)^{x-q}\binom{p-q}{p-x}\theta_x \qquad (112)\\ &= \binom{p-q}{p-q}\theta_q = \theta_q, \qquad (113)\end{aligned}$$

where in (110) we have used the fact that the multinomial term is zero whenever $y < q$ or $y > x$; in (111) we substitute $x-y$ by z that takes values in $\{0, 1, \ldots, x-q\}$; and (113) holds since $(1-1)^{x-q}$ is zero except for $x = q$. This complete the proof of Proposition 1.

Proof of Proposition 2: Let us define

$$f(T) = (\alpha-1)\binom{T}{\alpha} - \beta\binom{T-1}{\alpha-1},$$

for fixed α and β. We prove that $f(T) \ge -\binom{\beta}{\alpha}$ by a two-sided induction on T, i.e., for $T \ge \beta$ and $T \le \beta$. First, note that

$$\begin{aligned}X \cap \bigcap_{t\in T} A_t &= \left\lfloor |X| + \sum_{U\subset T:\,|U|=1} |A_U| \right\rfloor - \left\lceil \sum_{U\subset T:\,|U|=1} |X\cup A_U| + \sum_{U\subset T:\,|U|=2} |A_U| \right\rceil + \left\lfloor \sum_{U\subset T:\,|U|=2} |X\cup A_U| + \sum_{U\subset T:\,|U|=3} |A_U| \right\rfloor\\ &\quad - \cdots + (-1)^{|T|-1}\left\lceil \sum_{U\subset T:\,|U|=|T|-1} |X\cup A_U| + \sum_{U\subset T:\,|U|=|T|} |A_U| \right\rceil + (-1)^{|T|} \sum_{U\subset T:\,|U|=|T|} |X\cup A_U| \end{aligned} \tag{105}$$

$$0 \geq \frac{1}{\binom{K-1}{T}} \sum_{\substack{T\subseteq[K-1]\\|T|=T}} \left[|A_T| - \sum_{\substack{U\subset T\\|U|=1}} |A_T \cup A_U| + \sum_{\substack{U\subset T\\|U|=2}} |A_T \cup A_U| - \cdots + (-1)^{|T|} \sum_{\substack{U\subset T\\|U|=|T^c|}} |A_T \cup A_U| \right]$$

$$= \frac{1}{\binom{K-1}{T}} \sum_{\substack{T\subseteq[K-1]\\|T|=T}} |A_T| - \frac{1}{\binom{K-1}{T}} \sum_{\substack{T\subseteq[K-1]\\|T|=T}} \sum_{\substack{U\subset T\\|U|=1}} |A_T \cup A_U| + \frac{1}{\binom{K-1}{T}} \sum_{\substack{T\subseteq[K-1]\\|T|=T}} \sum_{\substack{U\subset T\\|U|=2}} |A_T \cup A_U| - \cdots$$

$$+ (-1)^{|T|} \frac{1}{\binom{K-1}{T}} \sum_{\substack{T\subseteq[K-1]\\|T|=T}} \sum_{\substack{U\subset T\\|U|=|T|}} |A_T \cup A_U|$$

$$= \frac{1}{\binom{K-1}{T}} \sum_{\substack{V\subseteq[K-1]\\|V|=T}} |A_V| - \frac{\binom{T+1}{T}}{\binom{K-1}{T}} \sum_{\substack{V\subseteq[K-1]\\|V|=T+1}} |A_V| + \frac{\binom{T+2}{T}}{\binom{K-1}{T}} \sum_{\substack{V\subseteq[K-1]\\|V|=T+2}} |A_V| - \cdots + (-1)^{|T|} \frac{\binom{T+|T|}{T}}{\binom{K-1}{T}} \sum_{\substack{V\subseteq[K-1]\\|V|=T+|T|}} |A_V| \quad (107)$$

$$= \frac{1}{\binom{K-1}{T}} \sum_{\substack{V\subseteq[K-1]\\|V|=T}} |A_V| - \frac{\binom{K-1-T}{1}}{\binom{K-1}{T+1}} \sum_{\substack{V\subseteq[K-1]\\|V|=T+1}} |A_V| + \frac{\binom{K-1-T}{2}}{\binom{K-1}{T+2}} \sum_{\substack{V\subseteq[K-1]\\|V|=T+2}} |A_V| - \cdots + (-1)^{|T|} \frac{\binom{K-1-T}{|T|}}{\binom{K-1}{T+|T|}} \sum_{\substack{V\subseteq[K-1]\\|V|=T+|T|}} |A_V| \quad (108)$$

$$= \theta_T - \binom{K-1-T}{1} \theta_{T+1} + \binom{K-1-T}{2} \theta_{T+2} - \cdots + (-1)^{|T|} \binom{K-1-T}{|T|} \theta_{T+|T|}$$

$$= \sum_{j=T}^{K-1} (-1)^{j-T} \binom{K-1-T}{j-T} \theta_j \quad (109)$$

for $T = \beta$, we have

$$f(\beta) = (\alpha - 1) \left[\binom{\beta}{\alpha} - \beta \binom{\beta-1}{\alpha-1} \right]$$
$$= (\alpha - 1) \left[\binom{\beta}{\alpha} - \alpha \binom{\beta}{\alpha} \right] = -\binom{\beta}{\alpha}, \quad (114)$$

which shows the inequality in (87) holds with equality. Let (114) be the base case for induction. Next, we consider two individual cases for $T \geq \beta$ and $T \leq \beta$.

a) $T \geq \beta$ *:* Assume that (87) holds for $T = t \geq \beta$, i.e, $f(t) \geq -\binom{\beta}{\alpha}$. In what follows, we prove that (87) holds for $T = t + 1$:

$$f(t+1)$$
$$= (\alpha-1) \left[\binom{t+1}{\alpha} - \beta \binom{t}{\alpha-1} \right]$$
$$= (\alpha-1) \left[\left(\binom{t}{\alpha} + \binom{t}{\alpha-1} \right) - \beta \left(\binom{t-1}{\alpha-1} + \binom{t-1}{\alpha-2} \right) \right]$$
$$= (\alpha-1) \left[\binom{t}{\alpha} - \beta \binom{t-1}{\alpha-1} \right] + (\alpha-1) \left[\binom{t}{\alpha-1} - \beta \binom{t-1}{\alpha-2} \right]$$
$$= (\alpha-1) \left[\binom{t}{\alpha} - \beta \binom{t-1}{\alpha-1} \right] + t \left[\binom{t-1}{\alpha-2} - \beta \binom{t-1}{\alpha-2} \right]$$
$$= f(t) + (t-\beta) \binom{t-2}{\alpha-2} \geq -\binom{\beta}{\alpha}, \quad (115)$$

where the inequality in (115) holds due to $f(t) \geq -\binom{\beta}{\alpha}$ by the induction hypothesis, and the fact that $t \geq \beta$.

b) $T \leq \beta$ *:* Similar to the previous case, assume that (87) holds for $T = t \leq \beta$, i.e., $f(t) \geq -\binom{\beta}{\alpha}$. For $T = t-1$, we can write

$$f(t-1)$$
$$= (\alpha-1) \left[\binom{t-1}{\alpha} - \beta \binom{t-2}{\alpha-1} \right]$$
$$= (\alpha-1) \left[\left(\binom{t}{\alpha} - \binom{t-1}{\alpha-1} \right) - \beta \left(\binom{t-1}{\alpha-1} - \binom{t-2}{\alpha-2} \right) \right]$$
$$= (\alpha-1) \left[\binom{t}{\alpha} - \beta \binom{t-1}{\alpha-1} \right]$$
$$+ \beta \binom{t-2}{\alpha-2} - (\alpha-1) \binom{t-1}{\alpha-1}$$
$$= (\alpha-1) \left[\binom{t}{\alpha} - \beta \binom{t-1}{\alpha-1} \right]$$
$$+ \beta \binom{t-2}{\alpha-2} - (t-1) \binom{t-2}{\alpha-2}$$
$$= f(t) + (\beta - (t-1)) \binom{t-2}{\alpha-2} \geq -\binom{\beta}{\alpha}, \quad (116)$$

where the inequality in (116) holds since $f(t) \geq -\binom{\beta}{\alpha}$ by the induction hypothesis, and $\beta \geq t - 1$. This completes the proof of Proposition 2 for any arbitrary T.

Proof of Proposition 3: In order to prove the inequality, we can rewrite $V_{\alpha,\beta}$ defined in (88) as

$$\begin{aligned}
V_{\alpha,\beta} &= \binom{K-\alpha-1}{K-\beta-1} + \frac{\binom{K-\alpha-1}{S-1}\binom{K-1}{K-\beta-1}}{\binom{K-1}{S-1}}(\alpha-1) - \frac{\alpha\beta}{K-S} \\
&= \binom{K-\alpha-1}{K-\beta-1} + \frac{(K-\alpha-1)!\,(K-S)!}{(K-\alpha-S)!\,(K-\beta-1)!\,\beta!} \\
&\quad \cdot \frac{\alpha!\,(\beta-\alpha)!}{\alpha!\,(\beta-\alpha)!}(\alpha-1) - \frac{\alpha\beta}{K-S} \\
&= \binom{K-\alpha-1}{K-\beta-1} + \frac{\binom{K-\alpha-1}{K-\beta-1}\binom{K-S}{\alpha}}{\binom{\beta}{\alpha}}(\alpha-1) - \frac{\alpha\beta}{K-S} \\
&= \binom{K-\alpha-1}{K-\beta-1}\left(1 + \frac{\binom{K-S}{\alpha}}{\binom{\beta}{\alpha}}(\alpha-1)\right) - \frac{\alpha\beta}{K-S} \\
&= \frac{\binom{K-\alpha-1}{K-\beta-1}}{\binom{\beta}{\alpha}}\binom{\beta}{\alpha} + (\alpha-1)\binom{K-S}{\alpha} - \frac{\alpha\beta}{K-S}\binom{K-S}{\alpha} \\
&= \frac{\binom{K-\alpha-1}{K-\beta-1}}{\binom{\beta}{\alpha}}\binom{\beta}{\alpha} + (\alpha-1)\binom{K-S}{\alpha} - \beta\binom{K-S-1}{\alpha-1} \\
&\geq 0, \qquad (117)
\end{aligned}$$

where the inequality in (117) holds due to the claim of Proposition 2 for $T = K - S$, that is

$$f(T-S) = (\alpha-1)\binom{K-S}{\alpha} - \beta\binom{K-S-1}{\alpha-1} \geq -\frac{\beta}{\alpha}.$$

This completes the proof of Proposition 3.

Appendix G
Decomposition of File Transition Graph: Proof of Lemma 7

Given a file transition graph $G(V, E)$, let us construct an undirected bipartite graph $H(V, E)$ (or more accurately bipartite multigraph, since there might be multiple edges between two nodes) by distinguishing worker nodes at iterations t and $t+1$. More precisely, the vertex set V is partitioned into two disjoint subsets $W^{(t)} = \{W_i^{(t)} : i \in [K]\}$ and $W^{(t+1)} = \{W_i^{(t+1)} : i \in [K]\}$. For every file F^j with $j \in u(i) \cap d()$, we include one edge e_j between $W_i^{(t)}$ and $W^{(t+1)}$. Note that if there is more than one file processed by W_i at iteration t and assigned to W at iteration $t+1$, then there are multiple edges between the two vertices $W_i^{(t)}$ and $W^{(t+1)}$ in the graph, one for each file. From the aforementioned description of $H(V, E)$, it is evident that $H(V, E)$ is an N/K-regular graph bipartite multigraph. It is easy to see that the original file transition graph $G(V, E)$ can be recovered from $H(V, E)$ by collapsing nodes $W_i^{(t)}$ and $W_i^{(t+1)}$ for each $i \in [K]$. More precisely, each edge between $W_i^{(t)}$ and $W^{(t+1)}$ in $H(V, E)$ is corresponding to a directed edge from W_i to W in $G(V, E)$. This shows a one-to-one mapping between the directed graph $G(V, E)$ and the undirected bipartite graph $H(V, E)$.

Our proposed decomposition for the graph $G(V, E)$ is based on the decomposition of the bipartite graph $H(V, E)$ into perfect matchings. Assume $H(V, E)$ can be decomposed into N/K perfect matchings between $W^{(t)}$ and $W^{(t+1)}$, namely $H_j(V, E_j)$, where $j \in [N/K]$ and $|E_j| = K$, and the degree of each node in $H_j(V, E_j)$ is exactly one. Then, by collapsing nodes $W_i^{(t)}$ and $W_i^{(t+1)}$, for $i \in [K]$, in $H_j(V, E_j)$, we get a directed graph $G_j(V, E_j)$ over $V = \{W_1, W_2, \ldots, W_K\}$ where the in-degree and out-degree of each node W_i in $G_j(V, E_j)$ are equal to those of $W_i^{(t)}$ and $W_i^{(t+1)}$ in $H_j(V, E_j)$, respectively, which are both equal to 1.

The fact that the bipartite graph $H(V, E)$ can be decomposed into N/K perfect matchings follows from a recursive application of Hall's theorem. We use Hall's theorem to find perfect matchings in $H(V, E)$ as follows.

Theorem 6 (Theorem 1 in [30]): A bipartite graph $H(V, E)$, with vertex set $V = W^{(t)} \cup W^{(t+1)}$, contains a complete matching from $W^{(t)}$ to $W^{(t+1)}$ if and only if

$$|N(T)| \geq |T|, \qquad (118)$$

for every non-empty subset $T \subseteq W^{(t)}$.

In order to apply Hall's theorem to $H(V, E)$, we need to test the necessary and sufficient condition. Let T be an arbitrary set of vertices in $W^{(t)}$, i.e., $T \subset W^{(t)}$, and $N(T)$ denote the set of neighbors of T in $W^{(t+1)}$. Recall that the degree of each vertex in T is N/K. Hence, there are a total of $|T|N/K$ edges exit T to reach nodes in $N(T)$. However, note that the total number of edges incoming to $N(T)$ cannot exceed $|N(T)|N/K$, because each node in $N(T)$ has degree N/K. The set of edges outgoing T is a subset of edges incoming to $N(T)$. Therefore, we have $|T|N/K \leq |N(T)|N/K$, which implies $|T| \leq |N(T)|$. Thus, the condition of Hall's theorem holds and there exists one perfect matching between the nodes in $W^{(t)}$ and $W^{(t+1)}$. Including the set of edges that constitute such a perfect matching in the set $E_{N/K}$, we obtain the subgraph $H_{N/K}(V, E_{N/K})$.

Next, one can remove the edges in $E_{N/K}$ from E. This reduces the degree of each node by 1, resulting in a new regular bipartite graph, with degree $N/K - 1$. Repeating the same argument on each residual graph, we can find a perfect matching, and then we remove its edges from the bipartite graph $H(V, E)$ until no edge is left. This provides us with N/K perfect matchings in $H(V, E)$, and each perfect matching corresponds to a subgraph of the file transition graph $G(V, E)$. This completes the proof of Lemma 7.

Remark 6: The problem of finding a perfect matching in an N/K-regular bipartite graph $H(V, E)$, with $|V| = 2K$ vertices and $|E| = K \times N/K = N$ edges, is well-studied in the graph theory literature. For regular bipartite graphs, a perfect matching is known to be computable in $O(N)$ time [31]. Recently, the authors in [32] have proposed a randomized algorithm that finds a perfect matching in an N/K-regular bipartite graph. The resulting runtime is $O(K \log K)$ (both in expectation and with high probability). We refer the interested reader to the aforementioned references for further details.

Example 6 (Example 5 Continued): Let us revisit the data shuffling system studied in Example 5, with $K = 4$ worker

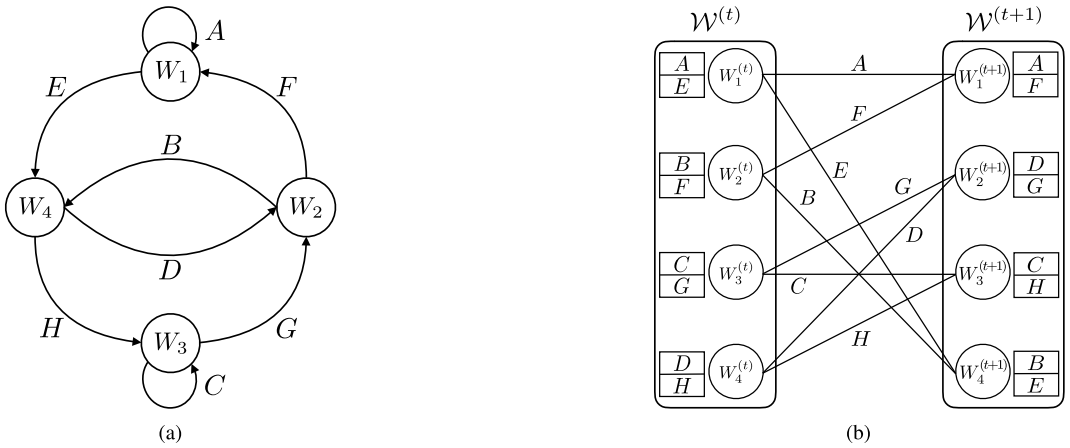


Fig. 16. (a) The file transition graph $G(V, E)$ for the data shuffling system of Example 5. (b) The corresponding bipartite graph $H(V, E)$.

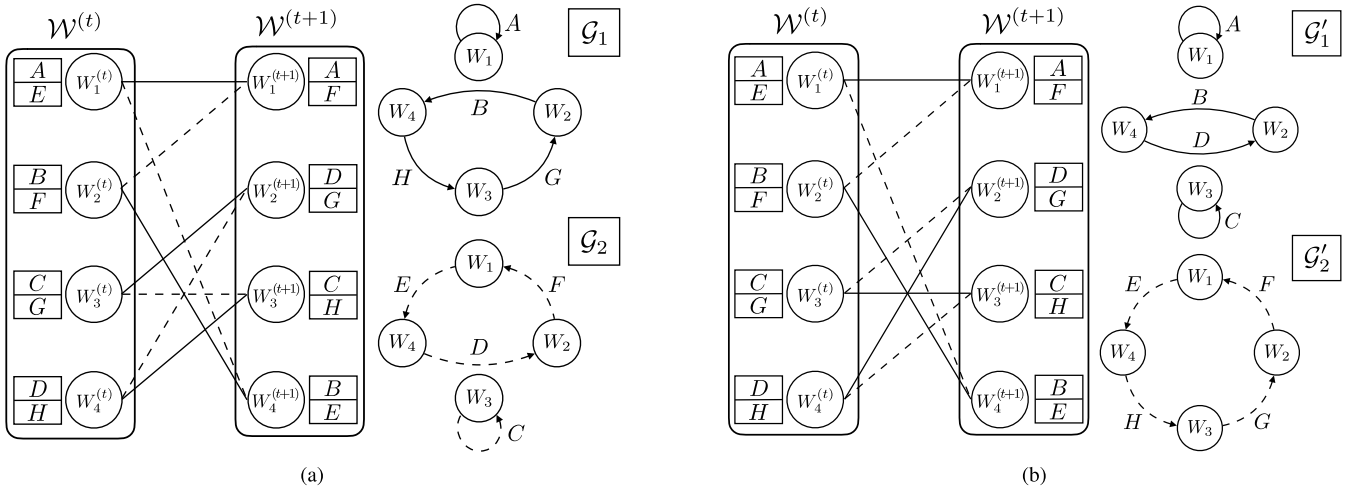


Fig. 17. (a) One decomposition of $H(V, E)$, shown in Fig. 12b, into $N/K = 2$ perfect matchings, designated by solid and dashed lines, and the corresponding decomposition of $G(V, E)$, shown in Fig. Fig. 12a, into two canonical subgraphs. (b) Another decomposition of $H(V, E)$ into $N/K = 2$ perfect matchings, and the corresponding decomposition of $G(V, E)$ into canonical subgraphs.

nodes, and $N = 8$ files, denoted by $\{A, B, C, D, E, F, G, H\}$. Fig. 16a captures the file transition graph $G(V, E)$, and Fig. 16b depicts the corresponding bipartite graph $H(V, E)$. For example, the directed edge, labeled by E, from W_1 to W_4 in $G(V, E)$ indicates that file E is processed by worker nodes W_1 and W_4 at iterations t and $t + 1$, respectively. Accordingly, there is an edge, labeled by E, between $W_1^{(t)}$ and $W_4^{(t+1)}$ that shows an assignment of file E to worker node W_1 at iteration t, and to worker node W_4 at iteration $t + 1$.

After constructing $H(V, E)$, we decompose it into $N/K = 2$ perfect matchings between $W^{(t)}$ and $W^{(t+1)}$, designated by solid and dashed lines in Fig. 17a. The corresponding canonical subgraphs of $G(V, E)$ are also shown in the figure. Another possible decomposition of $H(V, E)$ is depicted by Fig. 17b, along with the corresponding decomposition of $G(V, E)$. The existence of the decomposition of $H(V, E)$ is guaranteed by Hall's theorem, which results in the feasibility of the decomposition of $G(V, E)$. However, the graph decomposition is not unique, and there may be more than one decomposition for one instance of the data shuffling problem.

Appendix H
Pseudocodes

Algorithm 1 describes the file partitioning and labeling, while Algorithm 2 presents the cache placement. Next, Algorithms 3, 4, and 5 describe the encoding, decoding, and cache updating and subfile relabeling, respectively. Finally, Algorithm 6 presents the graph decomposition that is based on the Hungarian algorithm [28], [29].

Algorithm 1 `partitionFiles`

Input: $N, K, S, F^j : j \in [N]$
Output: $F^j_\Gamma : j \in [N], i \in [K], j \in u(i), \Gamma \subseteq [K] \setminus \{i\}, |\Gamma| = S/(N/K) - 1$
$S \leftarrow S/(N/K)$
for $i \leftarrow 1$ **to** K **do**
  for all $j \in u(i)$ **do**
    Worker node W_i partitions F^j into $\binom{K-1}{S-1}$ subfiles of equal sizes: $F^j_\Gamma : \Gamma \subseteq [K] \setminus \{i\}, |\Gamma| = S - 1$
  end for
end for

Algorithm 2 `placeCache`

Input: $N, K, S, t, \quad F_{\Gamma}^{j} : j \in [N], i \in [K], j \in u(i), \Gamma \subseteq [K] \setminus \{i\}, |\Gamma| = S/(N/K) - 1$
Output: $\{Z_i(t) : i \in [K]\}$
$S \leftarrow S/(N/K)$
for $i \leftarrow 1$ **to** K **do**
  $P^i(t) \leftarrow F_{\Gamma}^{j} : j \in u(i), \ \Gamma \subseteq [K] \setminus \{i\}, \ |\Gamma| = S - 1$
    i.e., $P^i(t) \leftarrow F^j : j \in u(i)$
  for all $\in [N] \setminus u(i)$ **do**
    $E_i \leftarrow F_{\Gamma} : i \in \Gamma \subseteq [K], \ |\Gamma| = S - 1$
  end for
  $E_i(t) \leftarrow \in [N] \setminus u(i) \quad E_i$
  $Z_i(t) \leftarrow P^i(t) \cup E_i(t)$
end for

Algorithm 3 `encodeSubmessages`

Input: $K, S, \quad F_{\Gamma}^{i} : i \in [K], \ \Gamma \subseteq [K]\backslash\{i\}, |\Gamma| = S - 1$, $\{d(i) : i \in [K]\}$
Output: $\{X_{\Delta} : \Delta \in D\_SET\}$
for all $\Delta \subseteq [K - 1]$ **and** $|\Delta| = S$ **do** $\backslash$
  $X_{\Delta} \leftarrow \big(F^{i}_{\Delta\backslash\{i\}} \oplus F^{d(i)}_{\Delta\backslash\{d(i)\}} \oplus F^{d(i)}_{(\{j\}\cup\Delta)\backslash\{i,d(i)\}} \big)$ $/$
    $i \in \Delta \quad j \in [K]\backslash\Delta$
end for
$X \leftarrow \{X_{\Delta} : \Delta \subseteq [K - 1], \ |\Delta| = S\}$
The master node broadcasts the message X to the worker nodes

Algorithm 4 `decodeSubfiles`

Input: $K, S, t, \{d(i) : i \in [K]\}, \{Z_i(t) : i \in [K]\}, \{X_{\Delta} : \Delta \subseteq [K - 1], |\Delta| = S\}$
Output: $\{Q_i(t) : i \in [K]\}$
for $i \leftarrow 1$ **to** K **do**
  if $i < K$ **then**
    for all $\Gamma \subseteq [K - 1]$ **and** $F_{\Gamma}^{d(i)} \in Q_i$ **do**
      Worker node W_i decodes subfile $F_{\Gamma}^{d(i)}$ from the sub-message $X_{\{i\}\cup\Gamma}$ using its cache contents Z_i
    end for
    for all $\Gamma \subseteq [K]$ **and** $K \in \Gamma$ **and** $F_{\Gamma}^{d(i)} \in Q_i$ **do**
      Worker node W_i decodes subfile $F_{\Gamma}^{d(i)}$ from the sub-message $X_{(\Gamma\backslash\{K\})\cup\{i,d(i)\}}$ using its cache contents Z_i and other subfiles already decoded by W_i
    end for
  else
    for all $\Gamma \subseteq [K - 1]$ **and** $F_{\Gamma}^{d(K)} \in Q_K$ **do**
      Worker node W_K decodes subfile $F_{\Gamma}^{d(K)}$ from the sub-messages $\big(j \in [K-1]\backslash\Gamma \quad X_{\{j\}\cup\Gamma}$ using its cache contents Z_K
    end for
  end if
  $Q_i(t) \leftarrow F_{\Gamma}^{d(i)} : \Gamma \subseteq [K]\backslash\{d(i)\}, |\Gamma| = S - 1 \quad \backslash Z_i(t)$
end for

Algorithm 5 `updateCaches`

Input: $K, S, t, \{d(i) : i \in [K]\}, \{Z_i(t) : i \in [K]\}, \{Q_i(t) : i \in [K]\}$
Output: $\{Z_i(t + 1) : i \in [K]\}$
for $i \leftarrow 1$ **to** K **do** Updating caches of worker nodes before iteration $t + 1$
  $j \leftarrow d^{-1}(i)$
  $A \leftarrow F_{\Gamma}^{i} : j \in \Gamma, \ \Gamma \subseteq [K]\backslash\{i\}, |\Gamma| = S - 1$.
  $S \leftarrow F_{\Gamma}^{d(i)} : i \in \Gamma, \ \Gamma \subseteq [K]\backslash\{d(i)\}, |\Gamma| = S - 1$
  $P^i(t + 1) \leftarrow F_{\Gamma}^{d(i)} : \Gamma \subseteq [K]\backslash\{d(i)\}, |\Gamma| = S - 1$
  $E_i(t + 1) \leftarrow (E_i(t) \setminus S) \cup A$
  $Z_i(t + 1) \leftarrow P^i(t + 1) \cup E_i(t + 1)$
end for
for $i \leftarrow 1$ **to** K **do** Relabeling subscripts of a set of subfiles of each worker node
  $j \leftarrow d^{-1}(i)$
  for all $F_{\Gamma}^{i} : j \in \Gamma, \ \Gamma \subseteq [K]\backslash\{i\}, |\Gamma| = S - 1$ **do**
    $\Lambda \leftarrow (\Gamma \setminus \{j\}) \cup \{i\}$
    Replace Γ in F_{Γ}^{i} by Λ
  end for
end for
for $i \leftarrow 1$ **to** K **do** Relabeling superscripts of all subfiles of each worker node
  $j \leftarrow d^{-1}(i)$
  for all $F_{\Gamma}^{i} : \Gamma \subseteq [K]\backslash\{i\}, |\Gamma| = S - 1$ **do**
    Replace i in F_{Γ}^{i} by j
  end for
end for

Algorithm 6 `decomposeGraph`

Input: $N, K, S, \{u(i), i \in [K]\}, \{d(i), i \in [K]\}$
Output: N/K perfect matchings
Construct a file transition graph matrix G that has size $K \times K$, where $G(i, j)$ is the number of files where $u(i) = d(j)$, otherwise, $G(i, j) = \infty$, for $i, j \in [K]$.
$H \leftarrow G$.
for $p \leftarrow 1$ **to** N/K **do**
  $[v, cost] =$ `Hungarian`(H).
    v denotes the matching vector of size $K \times 1$, where W_i is matched with $W_{v(i)}$ for $i \in [K]$.
  for $q \leftarrow 1$ **to** K **do**
    if $H(q, v(q)) = 1$ **then**
      $H(q, v(q)) \leftarrow \infty$
    else if $H(q, v(q)) > 1$ **then**
      $H(q, v(q)) \leftarrow H(q, v(q)) - 1$
    end if
  end for
end for

REFERENCES

[1] A. Elmahdy and S. Mohajer, "On the fundamental limits of coded data shuffling," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2018, pp. 716–720.

[2] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets," in *Proc. 2nd USENIX Workshop Hot Topics Cloud Comput. (HotCloud)*, 2010.

[3] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," in *Proc. 6th Symp. Oper. Syst. Design Implement. (OSDI)*, 2004.
[4] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly, "Dryad: Distributed data-parallel programs from sequential building blocks," in *Proc. 2nd ACM SIGOPS Eur. Conf. Comput. Syst. (EuroSys)*, 2007, vol. 41, no. 3, pp. 59–72.
[5] D. G. Murray, M. Schwarzkopf, C. Smowton, S. Smith, A. Madhavapeddy, and S. Hand, "CIEL: A universal execution engine for distributed data-flow computing," in *Proc. 8th USENIX Conf. Netw. Syst. Design Implement. (NSDI)*, 2011, pp. 113–126.
[6] J. Chung, K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Ubershuffle: Communication-efficient data shuffling for SGD via coding theory," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2017.
[7] B. Recht and C. Ré, "Toward a noncommutative arithmetic-geometric mean inequality: Conjectures, case-studies, and consequences," in *Proc. Conf. Learn. Theory (COLT)*, 2012.
[8] L. Bottou, "Stochastic gradient descent tricks," in *Neural Networks: Tricks of the Trade*. Berlin, Germany: Springer, 2012, pp. 421–436.
[9] M. Gürbüzbalaban, A. Ozdaglar, and P. Parrilo, "Why random reshuffling beats stochastic gradient descent," 2015, *arXiv:1510.08560*. [Online]. Available: https://arxiv.org/abs/1510.08560
[10] O. Shamir, "Without-replacement sampling for stochastic gradient methods," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2016, pp. 46–54.
[11] B. Ying, K. Yuan, S. Vlaski, and A. H. Sayed, "Stochastic learning under random reshuffling with constant step-sizes," *IEEE Trans. Signal Process.*, vol. 67, no. 2, pp. 474–489, Jan. 2019.
[12] J. HaoChen and S. Sra, "Random shuffling beats SGD after finite epochs," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2019, pp. 2624–2633.
[13] Q. Meng, W. Chen, Y. Wang, Z.-M. Ma, and T.-Y. Liu, "Convergence analysis of distributed stochastic gradient descent with shuffling," *Neurocomputing*, vol. 337, pp. 46–57, Apr. 2019.
[14] I. Safran and O. Shamir, "How good is SGD with random shuffling?" 2019, *arXiv:1908.00045*. [Online]. Available: https://arxiv.org/abs/1908.00045
[15] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Speeding up distributed machine learning using codes," *IEEE Trans. Inf. Theory*, vol. 64, no. 3, pp. 1514–1529, Mar. 2018.
[16] M. A. Attia and R. Tandon, "Near optimal coded data shuffling for distributed learning," *IEEE Trans. Inf. Theory*, vol. 65, no. 11, pp. 7325–7349, Nov. 2019.
[17] M. A. Attia and R. Tandon, "Information theoretic limits of data shuffling for distributed learning," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2016, pp. 1–6.
[18] M. A. Attia and R. Tandon, "On the worst-case communication overhead for distributed data shuffling," in *Proc. 54th Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, 2016, pp. 961–968.
[19] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *IEEE Trans. Inf. Theory*, vol. 60, no. 5, pp. 2856–2867, May 2014.
[20] S. A. Jafar, "Interference alignment: A new look at signal dimensions in a communication network," *Found. Trends Commun. Inf. Theory*, vol. 7, no. 1, pp. 1–134, 2011.
[21] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "The exact rate-memory tradeoff for caching with uncoded prefetching," *IEEE Trans. Inf. Theory*, vol. 64, no. 2, pp. 1281–1296, Feb. 2018.
[22] L. Song, C. Fragouli, and T. Zhao, "A pliable index coding approach to data shuffling," *IEEE Trans. Inf. Theory*, to be published.
[23] Z. Bar-Yossef, Y. Birk, T. S. Jayram, and T. Kol, "Index coding with side information," *IEEE Trans. Inf. Theory*, vol. 57, no. 3, pp. 1479–1494, Mar. 2011.
[24] K. Wan, D. Tuninetti, M. Ji, and P. Piantanida, "Fundamental limits of distributed data shuffling," in *Proc. 56th Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, 2018, pp. 662–669.
[25] S. Li, M. A. Maddah-Ali, and A. S. Avestimehr, "Coded MapReduce," in *Proc. 53rd Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, 2015, pp. 964–971.
[26] S. Li, M. A. Maddah-Ali, Q. Yu, and A. S. Avestimehr, "A fundamental tradeoff between computation and communication in distributed computing," *IEEE Trans. Inf. Theory*, vol. 64, no. 1, pp. 109–128, Jan. 2018.
[27] S. Li, M. A. Maddah-Ali, and A. S. Avestimehr, "A unified coding framework for distributed computing with straggling servers," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Dec. 2016, pp. 1–6.
[28] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval Res. Logist. Quart.*, vol. 52, no. 1, pp. 7–21, Feb. 2005.
[29] J. Munkres, "Algorithms for the assignment and transportation problems," *J. Soc. Ind. Appl. Math.*, vol. 5, no. 1, pp. 32–38, 1957.
[30] P. Hall, "On representatives of subsets," *J. London Math. Soc.*, vol. 1, no. 1, pp. 26–30, 1935.
[31] R. Cole, K. Ost, and S. Schirra, "Edge-coloring bipartite multigraphs in $O(E \log D)$ time," *Combinatorica*, vol. 21, no. 1, pp. 5–12, 2001.
[32] A. Goel, M. Kapralov, and S. Khanna, "Perfect matchings in $O(n \log n)$ time in regular bipartite graphs," *SIAM J. Comput.*, vol. 42, no. 3, pp. 1392–1404, 2013.

Adel Elmahdy received the B.Sc. degree in electronics and communications engineering from Alexandria University, Egypt, in 2011, and the M.Sc. degree in wireless communications from Nile University, Egypt, in 2015. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, University of Minnesota (UMN), USA. He was a Graduate Research Assistant with The American University in Cairo, Egypt, from 2011 to 2012. He was with the Wireless Intelligent Networks Center (WINC), Nile University, as a Research Assistant, from 2013 to 2015. He was a Research Associate with Qatar University, Qatar, from 2015 to 2016. His research interests span a broad range of topics focusing largely on information theory and coding, distributed machine-learning algorithms, and wireless communications. He was a recipient of the Graduate Fellowship from Nile University in 2013 and 2014 and the ADC Fellowship from the University of Minnesota Digital Technology Center (DTC) in 2016.

Soheil Mohajer received the B.Sc. degree in electrical engineering from the Sharif University of Technology, Iran, in 2004, and the M.Sc. and Ph.D. degrees in communication systems from the Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland, in 2005 and 2010, respectively. He was affiliated as a Post-Doctoral Researcher with Princeton University from 2010 to 2011 and the University of California at Berkeley, Berkeley, from 2011 to 2013. He is currently an Assistant Professor with the Department of Electrical and Computer Engineering, University of Minnesota, Twin Cities. He is broadly interested in information theory and its applications in distributed storage systems, wireless networks, bioinformatics, and statistical machine learning. He received the NSF CAREER Award in 2018.