Approximately Optimal Distributed Data Shuffling

Mohamed Adel Attia  Ravi Tandon
Department of Electrical and Computer Engineering
University of Arizona, Tucson, AZ
Email: {madel, tandonr}@email.arizona.edu

Abstract—Data shuffling between distributed workers is one of the critical steps in implementing large-scale learning algorithms. The focus of this work is to understand the fundamental trade-off between the amount of storage and the communication overhead for distributed data shuffling. We first present an information theoretic formulation for the data shuffling problem, accounting for the underlying problem parameters (i.e., number of workers, \( K \), number of data points, \( N \), and the available storage, \( S \) per node). Then, we derive an information theoretic lower bound on the communication overhead for data shuffling as a function of these parameters. Next, we present a novel coded communication scheme and show that the resulting communication overhead of the proposed scheme is within a multiplicative factor of at most 2 from the lower bound. Furthermore, we introduce an improved aligned coded shuffling scheme, which achieves the optimal storage vs communication trade-off for \( K < 5 \), and further reduces the maximum multiplicative gap down to 7/6, for \( K \geq 5 \).

I. INTRODUCTION

Distributed computing comes at the unavoidable communication cost due to data shuffling among distributed workers. Data shuffling can arise in many applications such as: a) random shuffling of the data-set across different points before each learning epoch so that each worker is assigned new training data, which provides statistical benefits, e.g., distributed gradient descent algorithm and its stochastic variations [1]; b) shuffling the data-set across attributes to assign different features to each worker, e.g., in mobile cloud gaming systems; and c) shuffling the data between the mappers and the reducers in the MapReduce framework. The application of coding theory to the data shuffling problem was first considered in [2], using excess storage at the workers in a probabilistic fashion, to create coded multicasting opportunities. In [3], the authors presented coded data shuffling in the MapReduce setting, with more redundant computations at the mappers, leading to a communication vs computation trade-off.

In this paper, we focus on the distributed data shuffling problem in a master-worker setting. At each learning epoch, the data set is randomly shuffled at the master node, and different data chunks need to be sent to the workers for distributed computation, which leads to the communication overhead. On one extreme, when all the workers can store the whole data set, no communication is needed for any random shuffle. On the other hand, when the storage is just enough to store the assigned data, also referred to as the no excess storage case, the communication is expected to be maximal. The goal of this work is to characterize the fundamental information-theoretic trade-off between the communication overhead and the available storage at the distributed workers.

In our prior work [4], we characterized the optimal tradeoff for \( K = 2 \) and \( K = 3 \) workers. In [5], it was shown that even for the no excess storage case, coding opportunities can still be leveraged. In [6], a pliable index coding approach for data shuffling was considered for semi-random shuffles. In this work however, we consider the tradeoff between worst-case rate over all possible shuffles and storage. The data shuffling problem is also related to the index coding problem [7]. The key difference is that the side information in the data shuffling problem (i.e., data stored at distributed workers) is generally not static, and the side information itself can change over time.

Summary of contributions: We first derive an information theoretic lower bound on the worst-case communication overhead for the data shuffling problem, based on a novel bounding methodology, similar in spirit to the recent converse proofs in the coded caching literature [8], [9]. Next, we introduce our achievable scheme based on a “structurally invariant placement and update” procedure that maintains the structure of the storage and allows the use of coded data delivery similar to [10]. We show that the rate of this scheme is within a multiplicative gap of 2 for all problem parameters. In order to close this gap, we then introduce a novel aligned coded shuffling scheme combining the ideas of coding and interference alignment. This scheme matches with the lower bound for all \( K < 5 \), and reduces the multiplicative gap to 7/6, for all \( K \geq 5 \).

II. SYSTEM MODEL

We assume a master node which has a data-set \( \mathcal{A} \), containing \( N \) data points, \( D_1, \ldots, D_N \), of size \( d \) bits each, and \( K \) distributed workers connected to the master node via a shared link. Treating the data points \( D_n \) as i.i.d. random variables, we therefore have \( H(\mathcal{A}) = N \times H(D_n) = N d \). At each iteration, indexed by \( t \), the master node divides the data-set \( \mathcal{A} \) into \( K \) equal sized data batches (assuming \( N \) is divisible by \( K \)), which correspond to a random permutation of the data-set, \( \pi^t : \mathcal{A} \rightarrow \{\mathcal{A}^t(1), \ldots, \mathcal{A}^t(K)\} \), where \( \mathcal{A}^t(k) \) denotes the data partition designated to be processed by worker \( w_k \) at time \( t \).

Since the data batches are disjoint and span the whole data-set, i.e., \( \cup_{k \in [1:K]} \mathcal{A}^t(k) = \mathcal{A} \), and \( \mathcal{A}^t(i) \cap \mathcal{A}^t(j) = \phi \), for all \( i \neq j \), we have \( H(\mathcal{A}^t(k)) = \frac{N}{K} d \), for all \( k \in [1:K] \).

After getting the data batch, each worker locally computes a function (such as the gradient or sub-gradients of the assigned...
data points) to be processed subsequently at the master node. We assume that each worker \( w_k \) has a storage \( Z^i_k \) of size \( Sd \) bits, which is used to store some function of the data-set, and \( S \) denotes the storage parameter. Considering \( Z^i_k \) as a random variable, we then have

\[
H(Z^i_k) = Sd, \quad H(Z^i_k | A) = 0, \quad \forall k \in [1 : K]. \tag{1}
\]

For processing purposes, each worker \( w_k \) must at least store the assigned data batch \( A^i(k) \) of size \( \frac{N}{K} d \) bits at time \( t \) in \( Z^i_k \), which gives the range of storage parameter as \( N/K \leq S \leq N \). Also, we get the processing constraint as

\[
H(A^i(k) | Z^i_k) = 0, \quad \forall k \in [1 : K]. \tag{2}
\]

In the next epoch \( t + 1 \), the data-set is randomly reshuffled at the master node according to a random permutation \( \pi_{t+1} : A \rightarrow \{A^{t+1}(1), \ldots, A^{t+1}(K)\} \). The two phases of the overall process, namely Data Delivery and Storage Update, are described next.

### A. Data Delivery Phase

At time \( t + 1 \), the master node sends a function of the data batches for the subsequent shuffles \((\pi_t, \pi_{t+1})\), \( X(\pi_t, \pi_{t+1}) = \phi(A^1, \ldots, A^K, A^{t+1}(1), \ldots, A^{t+1}(K)) \) over the shared link, where \( \phi(.) \) is the data delivery encoding function. We also define \( R(\pi_t, \pi_{t+1}) \) as the normalized rate of the shared link based on the shuffles \((\pi_t, \pi_{t+1})\). We then have

\[
H(X(\pi_t, \pi_{t+1}) | A) = 0, \quad H(X(\pi_t, \pi_{t+1})) = R(\pi_t, \pi_{t+1})d. \tag{3}
\]

Each worker \( w_k \) should reliably decode the desired batch \( A^{t+1}(k) \) out of the transmitted function \( X(\pi_t, \pi_{t+1}) \), as well as the data stored in the previous time slot \( Z^i_k \), i.e., \( A^{t+1}(k) = \psi(X(\pi_t, \pi_{t+1}), Z^i_k) \), where \( \psi(.) \) is the decoding function at the workers. Therefore, for reliable decoding, we have the following decodability constraint at each worker:

\[
H(A^{t+1}(k) | Z^i_k, X(\pi_t, \pi_{t+1})) = 0, \quad \forall k \in [1 : K]. \tag{4}
\]

### B. Storage Update Phase

At the next iteration \( t + 1 \), the storage for each worker \( w_k \) is updated to \( Z^{t+1}_k \), which is a function of the old storage content \( Z^i_k \) as well as transmitted function \( X(\pi_t, \pi_{t+1}) \), i.e., \( Z^{t+1}_k = \mu(X(\pi_t, \pi_{t+1}), Z^i_k) \), where \( \mu \) is the update function. Therefore, we have the following storage-update constraint:

\[
H(Z^{t+1}_k | Z^i_k, X(\pi_t, \pi_{t+1})) = 0, \quad \forall k \in [1 : K]. \tag{5}
\]

The excess storage after storing \( A^{t+1}(k) \) in \( Z^{t+1}_k \), given by \( S - \frac{N}{K} \), can be used to opportunistically store a function of the remaining \( K - 1 \) data batches. For the scope of this work, we focus on uncoded storage schemes, meaning that the excess storage is dedicated to store uncoded functions of the remaining \( K - 1 \) batches. We give the notation \( A^{t+1}(i,k) \), where \( i \neq k \), as that part of \( A^{t+1}(i) \) which worker \( w_k \) stores in its excess storage at time \( t + 1 \). As a result, we can write the content of \( Z^{t+1}_k \) for uncoded storage placement as

\[
Z^{t+1}_k = \{A^{t+1}(k), \cup_{i \in [1 : K] \setminus k} A^{t+1}(i,k)\}. \tag{6}
\]
Fig. 2. Structural invariant storage placement, (a), and update, (b), for $K = N = 4$, and $S = 7/4$. Above the dotted line are assigned data points, and below is the excess storage used to store the sub-points labeled with the worker’s index.

$\{D_2, D_3, D_4\}$, and every sub-point is available at least in one of the remaining workers. Therefore, the master node sends six coded symbols, each being useful for two workers at the same time as follows: $D_2, D_3, D_4, (w_1, w_2), (w_1, w_3), (w_2, w_3)$. The rate of this transmission is $6 \times d/4 = 3d/2$ bits, and the pair $(S = 7/4, R = 3/2)$ is achieved.

Storage Update: The storage update at time $t + 1$ follows the storage placement at time $t$, and is shown Figure 2b. For example, $w_1$ stores $D_2$ completely, and keeps from $D_1$ only the sub-point $D_1, (1)$.

Our second main result in Theorem 2 gives an information theoretical lower bound on $R_{\text{worst-case}}^{\text{upper}}$.

Theorem 2: For the data shuffling problem, a lower bound on $R_{\text{worst-case}}^{\text{upper}}$ is given by the lower convex envelope of the following $K$ storage-rate pairs: for $m \in [1: K],$

$$S = m \frac{N}{K}, \quad R_{\text{worst-case}}^{\text{upper}} = \frac{N(K - m)}{Km}.$$

The complete proof of Theorem 2 is in [11, Appendix C]. We present the key ideas behind the converse proof in Example 2 which is presented after stating Theorem 3.

Remark 1 (Basic idea for the converse): A lower bound over the optimal rate $R_{\text{worst-case}}^{*}(\pi, \pi + 1)$ of a shuffling $(\pi, \pi + 1)$ also lowers the worst-case since $R_{\text{worst-case}}^{\text{upper}} \geq R_{\text{worst-case}}^{*}(\pi, \pi + 1)$. Therefore, we get lower bounds over $R_{\text{worst-case}}^{\text{upper}}$ by averaging out a set of lower bounds for a sequence of shuffles. The novel part in our proof is to carefully choose the right shuffles which lead to the highest lower bound. In addition, we leverage a novel bounding methodology similar to [8], [9], where the optimal uncoded cache placement system is considered. The difference in the data shuffling problem, is that the workers in addition to storing the data under processing, have excess storage and can also update their storage over time.

In our next result, we compare the upper and lower bounds in Theorems 1 and 2, respectively.

Theorem 3: For the data shuffling problem, the multiplicative gap between upper and lower bounds on $R_{\text{worst-case}}^{\text{upper}}$ given by Theorems 1, and 2, respectively, is bounded as follows:

$$\frac{R_{\text{upper}}^{\text{worst-case}}}{R_{\text{lower}}^{\text{worst-case}}} \leq \frac{K}{K - 1} \leq 2.$$  

The complete proof of Theorem 3 can be found in [11, Appendix C]. This Theorem shows that the gap between the bounds vanishes as $K$ increases, i.e., $\lim_{K \to \infty} \left(\frac{K}{K - 1}\right) = 1$.

Next, we discuss the example of $N = K = 4$ to present the ideas behind the converse proof of Theorem 2.

Example 2: Assume the $N = 4$ data points are assigned at time $t$ according to $\pi_t = (1, 2, 3, 4)$. From (2), each worker should store the assigned data point at time $t$, therefore

$$H(A^{(k)}(\pi_t)) = H(D_k | Z_k^t) = 0, \quad \forall k \in [1: 4].$$

Following Remark 1, we consider the following cyclic shuffle: for a permutation $\sigma : (1, 2, 3, 4) \rightarrow (\sigma_1, \sigma_2, \sigma_3, \sigma_4)$, we have $A^{(1)}(\sigma_k) = A^{(\sigma - 1)} = D_{\sigma_k - 1}$. Using the decodability constraint in (4), worker $w_{\sigma_k}$ must decode $A^{(1)}(\sigma_k)$ from its cache content $Z_{\sigma_k}^t$ as well as the transmission $X(\pi_t, \pi_{t+1})$ which gives the following condition:

$$H(D_{\sigma_k - 1} | Z_{\sigma_k}^t, X(\pi_t, \pi_{t+1})) = 0, \quad \forall k \in [1: 4].$$

Consequently, we get the following bound using (10) and (11):

$$H(A | Z_{\sigma_2}^t, Z_{\sigma_3}^t, Z_{\sigma_4}^t, X(\pi_t, \pi_{t+1})) \leq H(D_{\sigma_4} | Z_{\sigma_4}^t, X(\pi_t, \pi_{t+1})) + \sum_{k=1}^4 H(D_{\sigma_k} | Z_{\sigma_k}^t) = 0,$$  

where (a) follows since $A = \{D_1, D_2, D_3, D_4\}$, and from the fact that $H(A, B) \leq H(A) + H(B)$ and that conditioning reduces entropy. Next, we obtain the following bound:

$$4d = H(A) \leq I(A ; Z_{\sigma_2}^t, Z_{\sigma_3}^t, Z_{\sigma_4}^t, X(\pi_t, \pi_{t+1}))$$
where \(R_{\text{worst-case}}\) follows from (12), (b) follows from (1), and (3), where \(X_{\pi, (\pi_{1}, \ldots, \pi_{n})}\) and \(Z_{k}\) for \(k \in [1 : 4]\) are deterministic functions of the data-set \(A\), (c) follows from Remark 1, (10), (11), and because conditioning reduces entropy, (d) follows from the storage content in (6), where \(D_{i}(j)\) is the part of \(D_{i}\) stored in the excess storage of worker \(w_{j}\). at time \(t\) follows due to the data points are independent and since out of the cache contents \(Z_{k}\), the data sub-point \(D_{k}(i)\) only depends on the sub-point \(D_{k}(j)\), for any \(i \neq j \neq k\), (f) follows from the chain rule of entropy where \(D_{k}(W)\) is the part of \(D_{k}\) stored in the excess storage of the workers whose indexes are in \(W\).

Summing up over all possible \(4! = 24\) permutations \(\sigma\) of the ordered set \((1, 2, 3, 4)\), we arrive at the following bound,

\[
R_{\text{worst-case}}^{*} d \geq 3d - \frac{1}{24} \sum_{\sigma \in [4]} |H(D_{\sigma_{1}}(\sigma_{2})) + H(D_{\sigma_{1}}(\sigma_{3})),
\]

where \([4]!\) is the set of all possible permutations of the set \((1, 2, 3, 4)\), and we changed the summation indexes due to the symmetry in the summation.

We assume a generic placement strategy for storing the data point \(D_{k}\), at time \(t\), in the excess storage of the workers \(w_{i}\), where \(i \neq k\), by defining \(D_{k,w_{i}}\) as the partition of \(D_{k}\) stored exclusively in the excess storage of the worker whose indexes are in the set \(W \subseteq [1 : 4]\). Defining \(|D_{k,w_{i}}| = H(D_{k,w_{i}})/d\), we can express the following enentries in terms of \(D_{k,w_{i}}\):

\[
H(D_{k}) = \sum_{W \subseteq [1 : 4]\backslash k} |D_{k,w_{i}}|d, \quad H(D_{k}(i)) = \sum_{W \subseteq [1 : 4]\backslash k} |D_{k,w_{i}}|d.
\]

The following two constraints are obtained in terms of \(D_{k,w_{i}}\):

- **Data size constraint**: The total size of the data is given by

\[
4 \geq \frac{1}{d} \sum_{k=1}^{4} \sum_{\ell=0}^{3} |H(D_{k})| = \frac{3}{4} \sum_{\ell=0}^{3} \sum_{W \subseteq [1 : 4]\backslash k} |D_{k,w_{i}}|,\]

where (a) follows from (15), and \(x_{\ell} \geq 0\) is defined as \(x_{\ell} = \sum_{k=1}^{4} \sum_{W \subseteq [1 : 4]\backslash k} |D_{k,w_{i}}|,\) for \(\ell \in [0 : 3]\).

- **Excess storage size constraint**: The size of the total excess storage of all the workers cannot exceed \(4(S - 1)\) d bits,

\[
4(S - 1) \geq \frac{1}{d} \sum_{k=1}^{4} \sum_{W \subseteq [1 : 4]\backslash k} |W||D_{k,w_{i}}| \geq \frac{3}{4} \sum_{\ell=0}^{3} x_{\ell},
\]

where (a) is true since a unique partition \(|D_{k,w_{i}}(i)|\) is counted \(|W|\) number of times, which is the number of workers storing this partition as excess storage.

In the summation term of (14), we obtain the term \(|D_{k,w_{i}}|\) only for \(|W| \in [1, 2, 3]\). Moreover, due to the symmetry, the coefficient of \(|D_{k,w_{i}}|\) for any \(k \in [1 : K]\) and \(W \in [1 : K]\) \(k\) is equal for every value of \(|W|\). Assume this constant coefficient is \(c_{\ell}\) for \(|W| = \ell\), where \(\ell \in \{1, 2, 3\}\). Therefore, we can simplify the bound in (14) as follows:

\[
R_{\text{worst-case}}^{*} \geq 3 - c_{1}x_{1} - c_{2}x_{2} - c_{3}x_{3},
\]

Obtaining the coefficient \(c_{\ell}\) is equivalent to obtaining the coefficient of \(|D_{k,w_{i}}(i)|\) for any \(k \in [1 : K]\) and \(|W| = \ell\). For example, we obtain \(c_{1}\) by finding the coefficient of \(|D_{1,w_{i}}(i)|\) in (14). We get \(|D_{1,w_{i}}(i)|\) in the first term of the summation in (14), i.e., \(H(D_{\sigma_{1}}(\sigma_{3}, \sigma_{4}))\), only if \(\sigma_{1} = 1\) which is satisfied in 6 out of the 24 permutations. In the second term, we obtain \(|D_{1,w_{i}}(i)|\) only if \(\sigma_{1} = 1\) and \(\sigma_{2} \neq 2\) in total number of 4 permutations. In the third term, we obtain \(|D_{1,w_{i}}(i)|\) only if \(\sigma_{1} = 1\) and \(\sigma_{2} = 2\) in total number of 2 permutations. Therefore, the coefficient of \(|D_{1,w_{i}}(i)|\) is \(c_{1} = \frac{6}{24} = \frac{1}{4}\).

Following the same steps, and as discussed in detail in [11, Example 2], we obtain \(c_{2} = \frac{2}{3}\) and \(c_{3} = \frac{3}{4}\). Therefore, we can write the bound in (18) as follows:

\[
R_{\text{worst-case}}^{*} \geq 3 - \frac{x_{1}}{2} - \frac{2x_{2}}{3} - \frac{3x_{3}}{4}.
\]

We get three bounds over \(R_{\text{worst-case}}^{*}\) by selected elimination of some of the variables \(x_{\ell}\), where \(\ell \in [0 : 3]\), from (19) using the constraints in (16) and (17), as follows:

\[
R_{\text{worst-case}}^{*} \geq 5 - 2S + \frac{x_{2}}{3} + \frac{3x_{3}}{4} \geq 5 - 2S,
\]

\[
R_{\text{worst-case}}^{*} \geq \frac{7}{3} - \frac{2S}{3} + \frac{x_{1}}{3} + \frac{x_{3}}{4} \geq \frac{7}{3} - \frac{2S}{3},
\]

\[
R_{\text{worst-case}}^{*} \geq \frac{4}{3} - \frac{S}{3} + \frac{5x_{0}}{6} + \frac{x_{3}}{4} \geq \frac{4}{3} - \frac{S}{3}.
\]

The intersection of the three bounds in (20), (21), and (22) is the lower convex hull of the 4 storage-rate pairs, \((S = m, R = \frac{4-m}{m})\) for \(m \in [1 : 4]\), which is the lower bound on \(R_{\text{worst-case}}^{*}\) shown in Figure 1, satisfying Theorem 2 for \(K = N = 4\). We can also observe that the storage-rate pairs achieved in Theorem 1, i.e., \(7/4, 3/2, 5/2, 2/3\), and \((13/4, 1/4)\), are optimal. The maximum gap between the bounds is at \(S = 1\), and is given by 4/3, satisfying the bound in Theorem 3.

The next Theorem provides an improved gap through a new scheme, which we call as “aligned coded shuffling”.

**Theorem 4**: For the data shuffling problem, the lower bound over \(R_{\text{worst-case}}^{*}\) in Theorem 2 is achievable for \(K < 5\) (hence leads to the optimal tradeoff), while for \(K \geq 5\) is achievable within a multiplicative gap which is bounded as

\[
\frac{R_{\text{worst-case, upper}}}{R_{\text{worst-case, lower}}} \leq \frac{K - 3}{K - 1} \leq \frac{7}{6}.
\]

In [11, Appendix D], we present the complete proof of Theorem 4 by closing the gap between the two bounds in Theorems 1 and 2 for the storage values \(S = m \frac{K}{2}\), and \(m \in \{1, K - 2, K - 1\}\). To illustrate the new ideas, we revisit Example 1 with \(K = 4\) workers and \(N = 4\) data points.

**Example 3**: From Figure 1, we notice that if we achieve the storage-rate pairs, \((1, 3), (2, 1),\) and \((3, 1/3)\), then we can fully characterize \(R_{\text{worst-case}}^{*}\) using memory sharing (see [11, 724], 2018 IEEE International Symposium on Information Theory (ISIT))
To this end, we present the aligned coded shuffling scheme for data delivery, which combines the ideas of coded shuffling and interference alignment. Let us consider the same shuffle as in Example 1, i.e., \( \pi_t = \{1, 2, 3, 4\} \), and \( \pi_{t+1} = \{2, 3, 4, 1\} \). Due to space limitation, we only describe here the achievability for the storage-rate pair \((2, 1)\), while the rest of the storage points can be found in [11, Example 3].

Storage Placement: At time \( t \), every data point \( D_i \) is partitioned into 3 sub-points, each labeled by the indexes in the set \([1 : 4]\) excluding the index of the worker being assigned to \( D_i \). For example, the data point \( D_1 \) (assigned to worker \( w_1 \)) is partitioned as \( D_1 = \{D_1(2), D_1(3), D_1(4)\} \). The storage placement at time \( t \) is shown in Fig. 3a. First, every worker stores the assigned data point. Then, every worker \( w_k \) stores the sub-points labeled with \( k \) from the remaining data points.

Aligned Coded Shuffling: According to the storage placement, every worker needs 2 sub-points of the new assigned point at time \( t + 1 \). From an interference perspective, each one of the needed sub-points is an interference to exactly one worker. For example, at time \( t + 1 \), \( D_3(4) \) is (i) needed by \( w_2 \), (ii) available at \( (w_3, w_4) \), and (iii) interference at \( w_1 \). Worker \( w_1 \) thus faces interference from exactly two sub-points, \( D_3(4) \), and \( D_4(2) \) needed by \( w_2 \) and \( w_3 \), respectively. Therefore, we first create the aligned coded symbol \( D_3(4) \otimes D_4(2) \) which is: (i) available at \( w_4 \), (ii) useful for \((w_2, w_3)\), and (iii) the only source of aligned interference for \( w_1 \). Similarly, we can produce 4 aligned coded symbols, summarized as follows:

\[
\begin{align*}
\text{Coded Symbol} & \quad \text{Interference at} & \quad \text{Available at} & \quad \text{Useful for} \\
D_3(4) \otimes D_4(2) & \quad w_1 & \quad w_4 & \quad w_2 \text{ & } w_3 \\
D_1(3) \otimes D_4(1) & \quad w_2 & \quad w_1 & \quad w_3 \text{ & } w_4 \\
D_1(2) \otimes D_2(4) & \quad w_3 & \quad w_2 & \quad w_1 \text{ & } w_4 \\
D_2(3) \otimes D_3(1) & \quad w_4 & \quad w_3 & \quad w_1 \text{ & } w_2
\end{align*}
\]

Therefore, these 4 coded symbols provide every worker with the 2 needed sub-points. Moreover, it suffices to send only three independent linear combinations of these 4 coded symbols, since every worker already has one of them available locally at its storage. The rate of this transmission is \( R = 3 \times 1/3 = 1 \), and the pair \((S = 2, R = 1)\) is achievable which closes the gap in Fig. 1 for \( S = 2 \).

Storage update and sub-points relabeling: The storage update at time \( t + 1 \) is needed to preserve the structure of the stored data points. This involves relabeling some of the data sub-points shown by red dashed frames in Fig. 3b. For example, the data point \( D_1 \) is processed by \( w_1 \) at time \( t \), and \( w_4 \) at time \( t + 1 \). Thus, to maintain the structural properties of storage at time \( t + 1 \), worker \( w_1 \) keeps from \( D_1 \), the sub-point \( D_1(4) \) and relabels it to \( D_1(1) \), and the same relabeling for \( D_1(4) \) also happens at worker \( w_4 \). Relabeling of other points is done similarly (see [11, Example 3]).

IV. CONCLUSION

In this paper, we characterized the approximately optimal worst-case communication vs storage tradeoff for the data shuffling problem, within a constant multiplicative gap of \(7/6\). We also characterized the optimal tradeoff for all \( K < 5 \) through a novel aligned coded data shuffling scheme. Future directions include closing the gap for \( K \geq 5 \), and adding a sub-divisibility constraint over the data points.

REFERENCES