Fifty-Fifth Annual Allerton Conference
Allerton House, UIUC, Illinois, USA
October 3-6, 2017

Matrix Sparsification for Coded Matrix Multiplication

Geewon Suh
School of EE, KAIST
gwsuh91@kaist.ac.kr

Kangwook Lee
School of EE, KAIST
kw1jjang@kaist.ac.kr

Changho Suh
School of EE, KAIST
chsuh@kaist.ac.kr

***Abstract*—Coded computation is a framework for providing redundancy in distributed computing systems to make them robust to slower nodes, or stragglers. In a recent work of Lee et al., the authors propose a coded computation scheme for distributedly computing $A \times x$ in the presence of stragglers. The proposed algorithm first encodes the data matrix A to obtain an encoded matrix F. It then computes $F \times x$ using distributed processors, waits for some subset of the processors to finish their computations, and decodes $A \times x$ from the partial computation results. In another recent work, Dutta et al. explore a new tradeoff between the sparsity of the encoded matrix F and the number of processors to wait to compute $A \times x$. They show that one can introduce a large number of zeros into F to reduce the computational overheads while maintaining the number of processors to wait relatively low. Hence, one can potentially further speed up the distributed computation. In this work, motivated by this observation, we study the sparsity of the encoded matrix for coded computation. Our goal is to characterize the fundamental limits on the sparsity level. We first show that the Short-Dot scheme is optimal if an Maximum Distance Separable (MDS) matrix is fixed. Further, by also designing this MDS matrix, we propose a new encoding scheme that can achieve a strictly larger sparsity than the existing schemes. We also provide an information-theoretic upper bound on the sparsity.**

I. Introduction

Efficient parallel and distributed algorithms play a key role in fueling large-scale applications in a wide variety of domains such as machine learning and big data analysis. One critical challenge in developing efficient distributed algorithms is *the straggler problem* since the classical approaches [1], [2] are extremely vulnerable to stragglers. This is because *all* the computation results have to be collected by the master node (or the fusion processor), and hence one needs to wait until the "slowest" node finishes its task.

Several approaches have been proposed to mitigate the straggler effect in a distributed computing system. One prominent approach is based on online detection of the stragglers using various kinds of probing mechanisms. When some stragglers are detected, one may redistribute the jobs assigned to the identified stragglers to the other nodes or new nodes, if available. This kind of reactive policies are widely adopted in many practical systems such as Hadoop [3]. While this conservative approach can effectively react to the stragglers at a low resource overhead, sometimes it is not feasible at all to implement such frequent probing mechanisms. Moreover, the size of distributed tasks can be so small that such reactive policies are not effective.

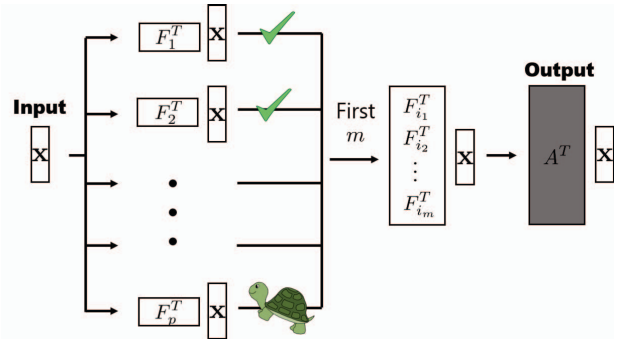

Fig. 1: The key idea of the MDS computation scheme [6]

Another interesting class of approaches is based on task replication [4], [5]. The key idea is very simple: one can allot the same task to multiple workers and collect the output of the fastest workers. These approaches can be easily deployed to the existing systems since they do require minimal modifications. Further, these proactive strategies can easily react to the stragglers at any time scale, though at relatively high resource overheads.

Recently, a new technique based on coding-theoretic ideas, called *coded computation*, has been proposed to speed up the distributed computation in the presence of stragglers [6]. Coded computation is a systematic framework of designing redundancy in distributed computation tasks using efficient codes. In a nutshell, a coded computation scheme *encodes* the computation tasks across distributed processors and *decodes* the desired computation result from the task results from some subset of the processors. The authors of [6] consider the linear computation task, i.e., a matrix-vector multiplication $A^\intercal x$, and show that a coded computation scheme based on Maximum Distance Separable (MDS) codes [7] can achieve an order-wise improvement in terms of expected computational time over the other existing schemes that do not employ coding-theoretic ideas.

The key idea of the basic coded computation scheme is illustrated in Fig. 1. Given the original data matrix A, the scheme first generates an encoded data matrix F, and then distributedly computes $F^\intercal \mathbf{x}$. Note that by applying an (p, m)-MDS code to the m columns of the matrix A, the scheme can generate p coded columns. It is clear that the fusion processor is able to decode $A^\intercal x$ as soon as m different computation results are collected.

978-1-5386-3266-6/17/$31.00 ©2017 IEEE

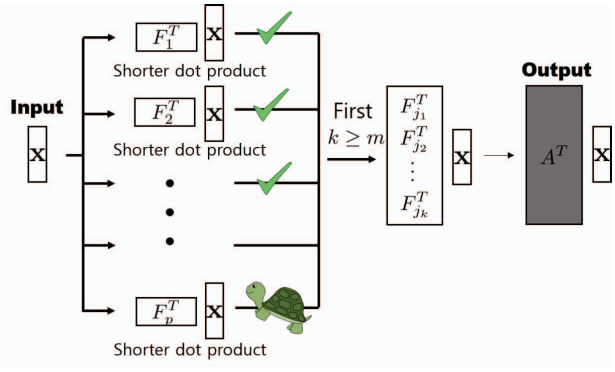

Fig. 2: The key idea of Short-Dot [8]

In [8], the authors show that there exists an interesting tradeoff between how sparse the coded matrix F can be and how flexible the computation scheme can be. Introducing a new design parameter k, $m \leq k \leq p$, which denotes the number of computation results required to decode the original computation task, the authors study the tradeoff between the number of nonzero elements in F and the flexibility parameter k. The authors propose a new coded computation scheme called Short-Dot, illustrated in Fig. 2. The key idea is to append $k - m$ dummy columns to A before applying the MDS code. By doing so, their scheme can reduce the size of distributed tasks at the cost of increased flexibility compared to the basic coded computation scheme. They also show that by appropriately choosing the size of tasks and the flexibility, the overall computational time can be reduced.

The goal of this work is to answer the following natural question: what is *the fundamental tradeoff* between the number of zeros in F, that we call *the sparsity*[1] of F, and the flexibility parameter k? In order to answer this question, we first introduce *the matrix sparsification problem*, which captures the essence of this tradeoff.

Our main findings on the matrix sparsification problem can be summarized as follows.

- (An MDS code is fixed, $k \geq m$) We first show that the tradeoff achieved by the Short-Dot scheme is optimal when the MDS code matrix is fixed. That is, the optimal sparsity is $n(k - m)$ where n is the number of rows in A.
- (An MDS code can be designed, $k = m$) We also characterize the fundamental tradeoff when k is fixed as m, and the only design parameter is the MDS code matrix. More specifically, we show that the optimal sparsity is $(k - 1)p$.
- (An MDS code can be designed, $k > m$) We propose a new matrix sparsification algorithm that can achieve a strictly better tradeoff under this general setup: our scheme can achieve the additional sparsity. Further, we provide an outer bound on the tradeoff.

[1]In this work, we define the sparsity of a matrix as the number of zeros in the matrix. Note the difference in the definitions of the sparsity between our work and [8].

A. Related Works on Coded Computation

In this section, we provide a brief overview of the related works on coded computation. Coded computation is a scheme that employs coding-theoretic ideas for *encoding* input data and/or output results and *running* distributed algorithms with those encoded input and output data [6]. The goal is to improve various performances of distributed algorithms such as average/tail computation latency in the presence of stragglers [6], [8], the probability of successful task completion under a deadline [9], approximation quality under a deadline [10], [11], communication overheads required for exchanging computation results [12], [13], security [14], optimization performance in the presence of data loss [15], etc.

Lee et al. propose a simple coded computation scheme based on MDS codes [6]. They also show via real-world experiments that the coded computation can significantly speed up certain computation tasks in practice. Dutta et al. show that by considering a more general encoding process, one may further speed up distributed computations.

The authors of [9] show that coded computation schemes can also significantly improve the probability of meeting computation deadline compared with uncoded computation schemes. In [10], [11], the authors show that the coded computation scheme can achieve a superior approximation quality at any time compared to the other schemes.

Another interesting metric is the communication cost between the distributed processors. Tandon et al. [12] propose a coded computation scheme for distributedly computing gradient descents. The authors of [13] propose a coded computation scheme that can achieve a significant speedup when a general function is computed with multicore setups.

B. Notations

Let $[n]$ denote the integer set $\{1, 2, \cdots, n\}$ and $[n_1 : n_2]$ denote $[n_2] \backslash [n_1 - 1]$. For a set A and $k \leq |A|$, we define $\binom{A}{k}$ to be all possible size-k subsets of A, i.e., $\{X \subseteq A : |X| = k\}$. For two matrices A_1 and A_2, $[A_1\ A_2]$ indicates the concatenation of A_1 and A_2.

We say that a matrix $B \in \mathbb{R}^{k \times p}$ satisfies the Maximum-Distance Separable (MDS) property, or B is an MDS code/matrix if any k columns of B are linearly independent.

II. The Matrix Sparsification Problem

We begin with providing a formal definition of *the matrix sparsification problem*.

A. Problem Formulation

A matrix sparsification algorithm encodes an input matrix $A \in \mathbb{R}^{n \times m}$ into a coded matrix $F \in \mathbb{R}^{n \times p}$ such that for some fixed k, $m \leq k \leq p$, *any* k columns of F span all the columns of A. Here, k denotes the resilience parameter to stragglers. More details are provided in Sec. II-B.

In this work, we focus on the following two-step encoding algorithm, inspired by the formulation of [8]. The first step of the two-step encoding algorithm appends $k - m$ dummy columns to A. Then, in the second step of the algorithm, it right-multiplies an MDS matrix B to $\tilde{A} = [A\ \bar{A}]$. More

		Achievable schemes	Upper bounds
Short-Dot [8]		$s = n(k-m)$ for P1	$s \leq n(k-m) + m^2\binom{p}{k-m+1}$ for P3
This work	P1 (Fixed B) (Sec. III-B)	N/A	$s \leq n(k-m)$
	P2 (No dummy columns) (Sec. III-C)	$s = (k-1)p$	$s \leq (k-1)p$
	P3 (Full flexibility) (Sec. III-D)	If $k \geq \frac{(n+p)^2}{(n+p)^2-n^2}m$, $s \approx n(k-m) + p\left(\sqrt{k} - \sqrt{k-m}\right)^2$. If $k < \frac{(n+p)^2}{(n+p)^2-n^2}m$, $s \approx \frac{nkp}{n+p}$.	$s \leq n(k-m) + kp$

TABLE I: Summary of our main results and the results in [8].

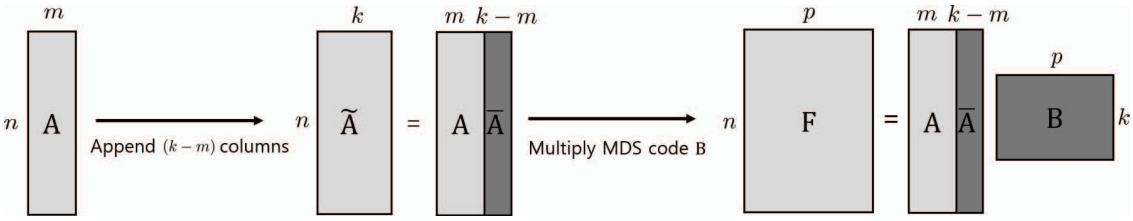

Fig. 3: The block diagram of the two-step encoding process. It first appends $(k-m)$ dummy columns to A, and right-multiplies the MDS matrix B to construct F.

formally, the two-step encoding algorithm of our interest can be described as

$$F = [\ A\ \bar{A}\ ]B \in \mathbb{R}^{n\times p},$$

where $\bar{A} \in \mathbb{R}^{n\times(k-m)}$ is a matrix consisting of $k-m$ dummy columns, and $B \in \mathbb{R}^{k\times p}$ is an MDS matrix. See Fig. 3 for the two-step encoding process.

The goal of the matrix sparsification problem is to maximize the *sparsity* of F by judiciously designing $\bar{A}$ and B, where the sparsity is defined as follows.

Definition 1. *Sparsity* $s(F)$ of a real matrix F is defined as the number of zero entries in F.

An encoding scheme can be fully characterized by specifying the pair of $\bar{A}(A)$ and $B(A)$ (or simply $\bar{A}$ and B). If an encoding scheme ϕ can achieve $s(F) \geq \theta$ for almost all matrices A, we say that the encoding scheme achieves the sparsity of θ, i.e., $s(\phi) = \theta$. And we say that $s^\star$ is an optimal sparsity if $s^\star = \max_\phi s(\phi)$, i.e., $s^\star$ is the maximum sparsity among all encoding schemes.

In this paper, we will simply use s instead of $s(F)$ if F is clear in the context. Further, let $A_i, \bar{A}_i, \tilde{A}_i$ denote the i-th row of $A, \bar{A}, \tilde{A}$, respectively. Let B_j denote the j-th column of B. For $J \in [p]$, B_J denotes the $k \times |J|$ sub-matrix of $B \in \mathbb{R}^{k\times p}$ by taking columns $\{B_j\}_{j\in J}$ of B.

Depending on the assumptions on $\bar{A}$ and B, one can consider the following three variations of the general matrix sparsification problem.

- (Problem 1: Fixed B [8]) What is the optimal design of dummy columns $\bar{A}$ when a fixed MDS code B is given?
- (Problem 2: No dummy columns [6]) What is the optimal choice of the MDS code B when $\bar{A} = \emptyset$?
- (Problem 3: Full flexibility) What is the optimal design of dummy columns $\bar{A}$ and MDS code B?

Note that Problem 1 where the MDS code B is assumed to be fixed, the general matrix sparsification problem reduces to the problem formulation in [8]. Further, Problem 2 is another extreme instance where no additional dummy columns are appended, and only B is used, i.e., $k = m$ and $\bar{A} = \emptyset$. The scheme proposed in [6] can be viewed as an achievable scheme for this regime. Finally, Problem 3 is the most general case where there are no restrictions on both $\bar{A}$ and B, and one has full flexibility in designing them.

B. Connection to Coded Matrix Multiplication

In this section, we briefly explain the connection between the matrix sparsification problem and coded matrix multiplication schemes. Consider the task of distributedly computing the product of a matrix $A^\intercal \in \mathbb{R}^{m\times n}$ and a vector $\mathbf{x} \in \mathbb{R}^{n\times 1}$ using p processors. For simplicity, assume that the number of processors p is no less than m, i.e., $p \geq m$.

Assume that there exists a matrix $F \in \mathbb{R}^{n\times p}$ of sparsity s such that any k columns of F can recover any columns of A, where $m \leq k \leq p$. Then, denoting the i-th column of F by F_i, it is clear that the task results $\{F_i^\intercal x\}$'s from any k processors suffice to recover $\{A_i^\intercal x\}$, and hence $A^\intercal x$.

Note that the amount of computation per processor decreases if the sparsity s increases. On the other hand, due to the fundamental tradeoff, one must wait for an increased number of processors' results compared to the original coded

matrix multiplication scheme, i.e., $k > m$. Hence, it is not clear how one should pick an operation point to optimize a certain metric of interest. The authors of [8] show that under some experimental circumstances, the Short-Dot scheme can further speed up the distributed matrix-vector multiplication by striking a critical balance between the sparsity and the flexibility of the code. In Sec. IV, we will revisit this connection and show how our results on the matrix sparsification problem can be applied to the coded computation problem.

C. Overview of the Existing Results

We now review the existing results on the matrix sparsification problem.

Lee et al. [6] propose the following algorithm for distributed matrix multiplication, called *coded matrix-vector multiplication*. Given a matrix A, they first construct the coded matrix $F = AB$ where B is an MDS matrix. Then, they assign the task of computing $F_i^\intercal \mathbf{x}$ to the i-th processor. Due to the property of MDS matrices, this algorithm can successfully compute the matrix-vector product $A^\intercal \mathbf{x}$ as soon as any m processors finish their computations.

We illustrate this approach via the following example. Consider the distributed computing system with $p = 3$ processors. Given a matrix $A \in \mathbb{R}^{n\times 2}$ and an MDS matrix

$$B = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix},$$

we first obtain the following encoded matrix:

$$\begin{aligned} F = AB &= \begin{pmatrix} A_1 & A_2 \end{pmatrix} \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} \\ &= \begin{pmatrix} A_1 & A_2 & A_1 + A_2 \end{pmatrix} \\ &= \begin{pmatrix} F_1 & F_2 & F_3 \end{pmatrix}. \end{aligned}$$

It is clear that one can always recover A from any $k = 2$ of F_i's. For instance, we can recover A from F_2 and F_3 since $A = \begin{pmatrix} (F_3 - F_2) & F_2 \end{pmatrix}$. Thus, one can successfully compute $A^\intercal \mathbf{x}$ from the outputs of the two fast processors, ignoring the slowest one. In general, this algorithm is able to compute $A^\intercal \mathbf{x}$ from the $k = m$ fastest processors, ignoring the $p-k = p-m$ slowest processors.

While the goal of [6] is not maximizing the sparsity of F, one can still view their scheme as an achievable scheme for the matrix sparsification problem. More precisely, their scheme can be interpreted as an achievable scheme for Problem 2 with $s = 0$ since they do not append any additional columns to A.

Improving upon this basic coded computation strategy, Dutta et al. [8] propose a new technique called Short-Dot. The Short-Dot scheme first fixes B as an arbitrary MDS matrix, and then carefully designs the dummy columns $\bar{A}$. The authors show that the dummy columns $\bar{A}$ can be chosen such that the sparsity of each column F_i is at least $\frac{n(k-m)}{p}$. That is, the Short-Dot scheme achieves $s = n(k - m)$.

III. Main Results

A. Our Approach

We now describe our approach to the matrix sparsification problem. The key idea is to view the matrix sparsification process as an inference problem as follows. We first view the input matrix A as a message and F as an encoded symbol. Recall that the matrix F has to satisfy the condition that *any* k columns of it has to span all the columns of A. This requirement can be viewed as the successful decoding condition. More precisely, consider an erasure channel that erases $p - k$ columns of F and outputs the remaining k columns as $\hat{F} \in \mathbb{R}^{n\times k}$. Upon receiving $\hat{F}$, the decoder recovers $\hat{A} \in \mathbb{R}^{n\times k}$. We say that the message matrix A is successfully recovered if $\hat{A} = A$. In Fig. 4, we provide a block diagram of the inference problem, which is equivalent to the matrix sparsification.

Leveraging this interpretation, we obtain achievable schemes and lower bounds for different instances of the matrix sparsification problem. Our findings can be summarized as follows.

- (Problem 1: Fixed B [8]) We show that the Short-Dot scheme indeed achieves the optimal sparsity when the MDS matrix B is fixed. That is, $s^\star = n(k - m)$.
- (Problem 2: No dummy columns [6]) We show that even without adding any dummy columns to A, it is possible to achieve $s = (k-1)p$. Further, we show that the optimality of our scheme, i.e., $s^\star = (k - 1)p$.
- (Problem 3: Full flexibility) We first propose an achievable scheme that jointly designs both $\bar{A}$ and B to achieve a strictly larger sparsity than those of the existing schemes. Moreover, we provide an upper bound on the optimal sparsity.

We also summarize our findings in Table I.

B. Problem 1: Fixed B

In this subsection, we characterize the optimal sparsity of F when we design $\bar{A}$ with fixed MDS matrix B. That is, we only consider the first step of the two-step encoding process. The following theorem characterizes the fundamental limit on sparsity of F.

Theorem 1. *Fix an MDS matrix B. For any coding scheme $\bar{A}$, $\min_A s(F) \le n(k - m)$.*

Proof. As mentioned in Sec. III-A, we view the computation process as an inference problem, considering A as a message. For simplicity, assume that the entries of A are in the finite field of size q.[2] We can easily check that $\tilde{A}$–F–$\hat{F}$–$\hat{\tilde{A}}$ forms a Markov chain. Then,

$$\begin{aligned} nm \log_2 q = H(A) &\le H(\tilde{A}) \\ &= H(\tilde{A}|\hat{\tilde{A}}) + I(\tilde{A}; \hat{\tilde{A}}) \\ &\overset{(1)}{\le} H(\tilde{A}|\hat{\tilde{A}}) + I(F; \hat{F}) \\ &= H(\tilde{A}|\hat{\tilde{A}}) + H(F) - H(F|\hat{F}) \\ &\overset{(2)}{\le} H(F) \\ &\overset{(3)}{\le} (nk - \min_A s(F)) \log_2 q, \end{aligned}$$

[2]The proof can be immediately generalized to the case of real numbers using either differential entropy or linear algebra.

Fig. 4: The matrix sparsification problem viewed as an inference problem. By viewing A as a message and stragglers as erasures, we can view the matrix sparsification problem as an inference problem (a channel coding problem).

which gives the upper bound on $\min_A s(F)$. Here, (1) holds by the data processing inequality, (2) holds from $H(\tilde{A}|\hat{\tilde{A}}) = 0$, i.e., Fano's inequality with successful recovery. Note that every row of F is a linear combination of k rows of B. Thus, the entropy of each row of F is at most $k \log_2 q$. Considering the minimum sparsity of F, we have $H(F) \leq (nk - \min_A s(F)) \log_2 q$, hence inequality (3) holds. □

Remark that Short-Dot [8] achieves $s = n(k-m)$. Thm. 1 gives the converse proof and thus implies that the Short-Dot scheme is optimal in the matrix sparsification problem when MDS matrix is fixed.

C. Problem 2: No Dummy Columns

In this subsection, we characterize the optimal sparsity of matrix sparsification problem when there are no dummy columns, i.e., $k = m$.

Definition 2. A matrix $A \in \mathbb{R}^{n\times k}$ satisfies Condition 1 if there does not exist a nonzero vector $v \in \mathbb{R}^k$ and $I \in \binom{[p]}{k}$ such that $v \in V_i = \text{span}\{A_i, \cdots, A_{i+k-2}\}$ for all $i \in I$.

Clearly, Condition 1 is not satisfied only within a measure 0 set, i.e., almost all A satisfies Condition 1. We now present our main theorem for Problem 2.

Theorem 2. *Suppose $n \geq p$. For all matrices A satisfying Condition 1, we can design an MDS matrix B such that $s \geq (k-1)p$. If every set of k rows of A are linearly independent, this sparsity is optimal.*

Proof. We first show the achievability. For $i \in [p]$, Let B_i be a vector orthogonal to $A_i, \cdots, A_{i+k-2}$ ($A_l = A_{l-n}$ if $l > n$). Since B_i is orthogonal to at least $k-1$ rows of A, $s = (k-1)p$ is achievable.

Now we show that B satisfies the MDS property for almost all A. Note that B_i is unique up to constant multiplication if $A_i, \cdots, A_{i+k-2}$ are linearly independent. Let $V_i = \text{span}\{A_i, \cdots, A_{i+k-2}\}$. Then, $B_i \in V_i^{\perp}$, the nullspace of V_i.

Suppose that there exist linearly dependent k columns $\{B_i\}_{i\in I}$ of B, where $I \in \binom{[p]}{k}$. This implies that $\dim\left(\sum_{i\in I} V_i^{\perp}\right) < k$. Here, the sum of linear vector spaces $W_1 + W_2$ is defined to be $\{w : w = a_1 w_1 + a_2 w_2 \text{ for some } a_1, a_2 \in \mathbb{R}, w_1 \in W_1, w_2 \in W_2\}$.

By an analog of De Morgan's law for linear spaces, we have $\dim\left(\bigcap_{i\in I} V_i\right)^{\perp} < k$, i.e., $\dim\left(\bigcap_{i\in I} V_i\right) \geq 1$. This implies that there exists a nonzero vector $v \in \mathbb{R}^k$ such that $v \in V_i = \text{span}\{A_i, \cdots, A_{i+k-2}\}$ for all $i \in I$. Therefore, by contrapositive proposition, we can design an MDS matrix B with $s \geq (k-1)p$ since A satisfies Condition 1.

We now prove the converse statement. If every set of k rows of A are linearly independent, each column of B is orthogonal to at most $k-1$ rows of A. Thus, any column of F cannot have more than $k-1$ zeros, meaning that $s = (k-1)p$ is the optimal sparsity. □

That is, the optimal sparsity for the matrix sparsification problem with no dummy columns equals to $(k-1)p$.

D. Problem 3: Full Flexibility

We now propose a new scheme that uses both coding stages, i.e., designs both $\bar{A}$ and B. Our scheme, explained in Algorithm 1, randomly designs some part of $\bar{A}$, designs MDS code B as a function of A and the designed part of $\bar{A}$, and then designs the rest of $\bar{A}$ as in Short-Dot [8]. Our scheme is illustrated in Fig. 5, and the sparsity pattern is illustrated in Fig. 6.

Let $t \leq m$ be a constant, which will be fixed later. Let $p' := \left\lfloor \min\left(\frac{p}{k-t}, \frac{n}{t}\right)\right\rfloor$ denote the maximum number of $t \times (k-t)$ blocks we can choose in F diagonally.

Definition 3. Fix an integer t, $0 < t \leq m$. Let $V_i := \text{span}(A_{(i-1)t+1}, \cdots, A_{it})$. A matrix $A \in \mathbb{R}^{n\times m}$ satisfies Condition 2 if it satisfies one of the following conditions for any $I \in \binom{[p']}{\lceil \frac{k}{k-t} \rceil}$:

1) $\text{rank}(A_{[(i-1)t+1,it]}) > t - (k-m-1)$ for all $i \in I$, or
2) $\text{rank}(A_{[(i-1)t+1,it]}) > t - (k-m)$ for all $i \in I$ and there is no nonzero $v \in \mathbb{R}^{k-m}$ such that $v \in \bigcap_{i\in I} V_i$.

It is straightforward to show that Condition 2 is satisfied if $\{A_i\}_{i\in[(j-1)t+1,jt]}$ is independent for all $j \in [p']$, which is not satisfied only within a measure 0 set. Hence, Condition 2 holds for almost all matrices A. We now characterize the achievable sparsity of our scheme.

Theorem 3. *Fix an integer t, $0 < t \leq m$. For all matrices A satisfying Condition 2, we can design $\bar{A}$ and B such that*

$$s \geq n(k-m) + t(m-t)\left\lfloor \min\left(\frac{p}{k-t}, \frac{n}{t}\right)\right\rfloor.$$

Proof. First, we randomly choose $\bar{A}_1, \cdots, \bar{A}_{tp'}$ with i.i.d. Gaussian entries. Next, for any $1 \leq j \leq p'$, we choose $\{B_{(j-1)(k-t)+1}, \cdots, B_{j(k-t)}\}$ to be a random basis of $\text{null}(\tilde{A}_{(j-i)t+1}, \cdots, \tilde{A}_{jt})$. By the argument in Appendix A, we can design B satisfying the MDS property.

With B designed above, $F_i = \tilde{A}_i B$, the i-th row of F, contains at least $(k-t)$ zeros for $1 \leq i \leq tp'$ since $\tilde{A}_i$ is orthogonal to $\{B_i\}_{[i'(k-t)+1,i'(k-t)]}$ where $i' = \left\lfloor \frac{i}{t} \right\rfloor$. For the remaining entries of $\tilde{A}$, we follow Short-Dot [8], which guarantees at least $k-m$ zeros for each row of F.

As a result, the first tp' rows of F have at least $k-t$ zeros and the remaining rows have at least $k-m$ zeros, which means that $s \geq n(k-m) + t(m-t)p'$. □

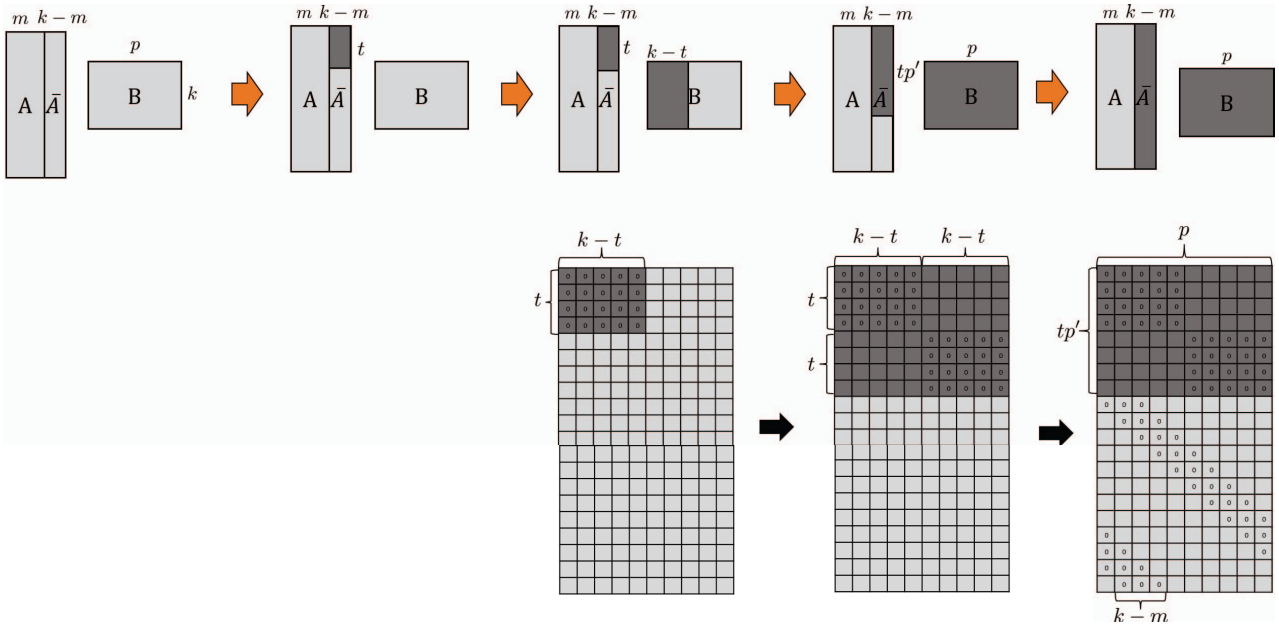

Fig. 5: Illustration of our achievable scheme in Sec. III-D. The first row illustrates our achievable scheme. It first chooses t rows of $\bar{A}$ at random and design first $k-t$ columns of B to be null vectors of $\tilde{A}_{[t]}$. We repeat this procedure until we choose all $\bar{A}$ or all B. Remaining $\bar{A}$ is designed by the Short-Dot scheme.

Algorithm 1 Our Scheme

Given: A, k, $\bar{A}_1, \cdots, \bar{A}_{tp'}$

for $j = 1$ *to* p' **do**
  $B_{(j-1)(k-t)+1}, \cdots, B_{j(k-t)} \leftarrow$
  null$(\tilde{A}_{(j-1)t+1}, \cdots, \tilde{A}_{jt})$
end for

for $j = tp' + 1$ *to* n **do**
  $U \leftarrow \{(j-1), \cdots, (j+k-m-1)\}$ mod p
  $B_U \leftarrow$ Columns of B indexed by U
  $\bar{A}_j \leftarrow -A_j B_U^{1:m} (B_U^{m+1:k})^{-1}$
end for

Now it remains to find the maximum value of the function $f(t) = t(m-t)\left\lfloor \min\left(\frac{p}{k-t}, \frac{n}{t}\right)\right\rfloor$. Let $g(t) := \frac{pt(m-t)}{k-t}$ and $h(t) := n(m-t)$. Since $g(t) \leq h(t)$ if and only if $pt \leq n(k-t)$, i.e., $t \leq t_1 := \frac{nk}{n+p}$, we get $f(t) \approx \min(g(t), h(t)) = g(t)\mathbf{1}\{t \leq t_1\} + h(t)\mathbf{1}\{t > t_1\}$.

One can show that $g(t)$ attains its maximum when $t = t_2 := k - \sqrt{k^2 - km} < m$, and $h(t)$ is a decreasing function with respect to t. So if $t_1 \geq t_2$, $f(t)$ has a maximum value $g(t_2)$ when $t = t_2$. If $t_1 \leq t_2$, $f(t)$ has a maximum value $h(t_1)$ when $t = t_1$.

From the previous observations, f attains its maximum value when $t^* \approx \min(t_1, t_2) = \min(k - \frac{pk}{n+p}, k - \sqrt{k^2 - km})$. Since $t_1 \geq t_2$ if and only if

$$\sqrt{k^2 - km} \geq \frac{pk}{n+p} \iff k \geq \frac{(n+p)^2}{(n+p)^2 - n^2} m,$$

we get the following theorem:

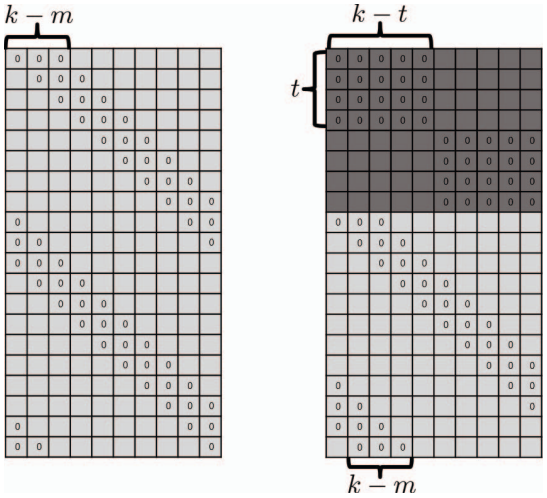

Fig. 6: Comparison of the sparsity patterns of F between Short-Dot [8] and our new scheme, proposed in Sec. III-D. Observe the additional zeros in the first tp' rows.

Theorem 4. *For all matrices A satisfying Condition 2, we can design $\tilde{A}, B$ such that*

$$s \approx \begin{cases} n(k-m) + p\left(\sqrt{k} - \sqrt{k-m}\right)^2, & \text{if } k \leq \frac{(n+p)^2}{(n+p)^2 - n^2} m, \\ \frac{nkp}{n+p}, & \text{otherwise.} \end{cases}$$

In the argument above, we use the approximation symbol '$\approx$' since we do not take the integer effect from the floor function into account, which is negligible unless the parameters are very small integers.

Comparing with Short-Dot [8], our scheme achives extra $p(\sqrt{k} - \sqrt{k-m})^2$ sparsity in F when $k \leq \frac{(n+p)^2}{(n+p)^2 - n^2} m$, and

$\frac{n}{n+p}(nm+mp-nk)$ otherwise. Note that $\frac{n}{n+p}(nm+mp-nk) = h(t_1) > h(t_2) > h(m) = 0$ when $t_1 < t_2$ since h is a decreasing function.

While our new achievable scheme strictly improves the sparsity achieved by Short-Dot scheme, we acknowledge that the improvement is marginal when $n \gg p$. This is because the extra sparsity obtained in Thm. 4 compared to Short-Dot does not scale with n.

We next give an upper bound on the optimal sparsity.

Theorem 5. *For any coding scheme ($\bar{A}$, B), $\min_A s(F) \leq n(k-m)+kp$.*

Proof. For simplicity, assume that the entries of A are in the finite field of size q.[3]

For given B, we can easily check that $\tilde{A}$–F–$\hat{F}$–$\hat{\tilde{A}}$ forms a Markov chain. Then,

$$\begin{aligned} nm\log_2 q = H(A) &= H(A,\bar{A},B) - H(\bar{A},B|A) \\ &= H(\tilde{A},B) - H(\bar{A},B|A) \\ &\leq H(B) + H(\tilde{A}|B) \\ &= H(B) + H(\tilde{A}|\hat{\tilde{A}},B) + I(\tilde{A};\hat{\tilde{A}}|B) \\ &\overset{(1)}{\leq} H(B) + H(\tilde{A}|\hat{\tilde{A}},B) + I(F;\hat{F}|B) \\ &= H(B) + H(\tilde{A}|\hat{\tilde{A}},B) + H(F|B) \\ &\quad - H(F|\hat{F},B) \\ &\overset{(2)}{\leq} kp\log_2 q + H(F|B) \\ &\overset{(3)}{\leq} (kp+nk-\min_A s(F))\log_2 q, \end{aligned}$$

which gives the upper bound on $\min_A s(F)$. Here, (1) holds by the data processing inequality, (2) holds from inequalities that $H(B) \leq |B|\log_2 q = kp\log_2 q$, $H(\tilde{A}|\hat{\tilde{A}},B) = 0$ by Fano's inequality with successful recovery. When B is given, every row of F is a linear combination of k rows of B. Thus, the entropy of each row of F is at most $k\log_2 q$. Considering the minimum sparsity of F, we have $H(F|B) \leq (nk - \min_A s(F))\log_2 q$ which implies (3). □

This bound exactly coincides with the number of free variables, i.e., $|\bar{A}| + |B|$. The authors in [8] give an upper bound of sparsity by showing that there exists a matrix A with $s \leq n(k-m)+m^2\binom{p}{k-m+1}$ under general construction of encoded matrix F. Thm. 5 gives a new tighter upper bound.

We remark that our upper bound is close to the achievable sparsity of our scheme when m is close to p. However, the gap between the upper bound and the achievable sparsity increases as m becomes smaller. See Fig. 7 for a numerical comparison.

IV. Latency Analysis

In this section, we compare the expected computation time of our algorithm to those of the existing algorithms. Based on the model proposed in [6], we assume that the time T_n for computing a dot product of length n is given by the cumulative distribution function

$$P(T_n \leq t) = (1 - e^{-\mu(\frac{t}{n}-1)})\mathbf{1}\{t \geq n\}.$$

Here, μ is the "straggling parameter", which determines the size of tail in the model. Intuitively, the computation time is proportional to n, and it takes at least n unit times to finish the task. The distribution of unpredictable latency is described as an exponential tail with parameter μ.

With sparsity s, the average length of the dot product each worker computes is equal to $n - \frac{s}{p}$. Then, the overall expected computation time is given by the k-th order statistic of $T_{n-\frac{s}{p}}$. The authors of [6] prove that the expectation of the k-th order statistic of T_n is given by $n\left(1 + \frac{\log(\frac{p}{p-k})}{\mu}\right)$. We optimize the expected computation time by choosing an appropriate resilience parameter k, which is equal to:

$$E(T) = \min_{m \leq k < p} \left(n - \frac{s}{p}\right)\left(1 + \frac{\log(\frac{p}{p-k})}{\mu}\right),$$

where s in our scheme is obtained by Thm. 4.

Based on this model, we compare the expected computational times of our algorithm with those of the existing algorithms. Fig. 7 shows that as m gets closer to p, the expected computational time approaches the fundamental limit, and the gap compared with previous works increases.

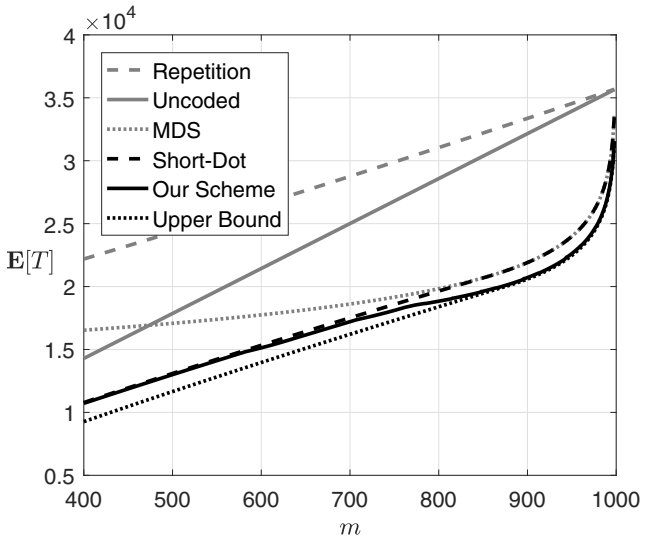

Fig. 7: Comparison of expected computational times. In the first figure, we compare the expected computational times of the existing computation schemes and our new scheme. In the second figure, we compare the computational time obtained by the upper bound of optimal sparsity with our scheme. ($\mu = 5, p = 1000, n = 15000$)

V. Discussion: Decoding Overhead

In the previous approaches [6], [8], the authors make use of the MDS codes for which efficient decoding algorithms are known. For instance, by applying an efficient algorithm for polynomial interpolation or Reed-Solomon decoding, one can

[3]The proof can be immediately generalized to the case of real numbers using either differential entropy or linear algebra.

show that the decoding time complexity of their algorithms is $\mathcal{O}(nk \log^2 k \log\log k)$.

On the other hand, our achievable scheme designs the MDS matrix as a function of the given matrix A, and hence it is not guaranteed to have an efficient decoding algorithm. That is, one has to invert the MDS matrix B and decode the computation results via a matrix-matrix multiplication, i.e., the decoding complexity is $\mathcal{O}(nk^2 + k^3)$. Therefore, when the decoding overhead is considered, it is not clear which scheme will be the fastest one in practice. The study of the fundamental tradeoff between the sparsity of F, the flexibility of the code k, and the decoding overhead is an interesting open direction. For instance, a new latency model proposed in a recent work [16], which can capture the decoding complexity, might be useful to study this tradeoff.

VI. Conclusion

In this work, we study the matrix sparsification problem with three variations: fixed MDS code, no dummy columns, and with full flexibility. When the MDS code is fixed, we prove that Short-Dot [8] is an optimal scheme by showing the converse proof. When there are no dummy columns, we find an achievable scheme and show that it achieves the optimal sparsity. With full flexibility, we propose a new scheme that achieves a larger sparsity by jointly designing both $\bar{A}$ and B. Also, we provide a new upper bound of the optimal sparsity.

Acknowledgement

This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIT) (2017-0-00694, Coding for High-Speed Distributed Networks).

References

[1] V. Kumar, A. Grama, G. Anshul, and G. Karypis, "Introduction to parallel computing: Design and analysis of algorithms," *The Benjamin/Cummings Publishing Company, Inc., Redwood City*, pp. 81–84, 1994.

[2] G. Fox, S. Otto, and A. Hey, "Matrix algorithms on a hypercube I: Matrix multiplication," *Parallel Computing*, vol. 4, no. 1, pp. 17 – 31, 1987.

[3] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The Hadoop distributed file system," in *2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, May 2010.

[4] G. Ananthanarayanan, A. Ghodsi, S. Shenker, and I. Stoica, "Effective straggler mitigation: Attack of the clones," in *Presented as part of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*, 2013.

[5] K. Lee, R. Pedarsani, and K. Ramchandran, "On scheduling redundant requests with cancellation overheads," in *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, Sept 2015.

[6] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Speeding up distributed machine learning using codes," *IEEE Transactions on Information Theory*, vol. PP, no. 99, pp. 1–1, 2017.

[7] R. Singleton, "Maximum distance q-nary codes," *IEEE Transactions on Information Theory*, vol. 10, no. 2, pp. 116–118, April 1964.

[8] S. Dutta, V. Cadambe, and P. Grover, "Short-Dot: Computing large linear transforms distributedly using coded short dot products," in *Advances In Neural Information Processing Systems*, 2016.

[9] S. Dutta, V. R. Cadambe, and P. Grover, "Coded convolution for parallel and distributed computing within a deadline," *CoRR*, vol. abs/1705.03875, 2017.

[10] N. S. Ferdinand and S. C. Draper, "Anytime coding for distributed computation," in *2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, Sept 2016.

[11] Y. Yang, P. Grover, and S. Kar, "Coding Method for Parallel Iterative Linear Solver," *arXiv preprint arXiv:1706.00163*, June 2017.

[12] R. Tandon, Q. Lei, A. G. Dimakis, and N. Karampatziakis, "Gradient coding: Avoiding stragglers in distributed learning," in *Proceedings of the 34th International Conference on Machine Learning*, vol. 70, 06–11 Aug 2017.

[13] K. Lee, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Coded computation for multicore setups," in *2017 IEEE International Symposium on Information Theory (ISIT)*, June 2017.

[14] R. Bitar, P. Parag, and S. E. Rouayheb, "Minimizing latency for secure distributed computing," in *2017 IEEE International Symposium on Information Theory (ISIT)*, June 2017.

[15] C. Karakus, Y. Sun, and S. Diggavi, "Encoded distributed optimization," in *2017 IEEE International Symposium on Information Theory (ISIT)*, June 2017.

[16] W. Halbawi, N. Azizan-Ruhi, F. Salehi, and B. Hassibi, "Improving distributed gradient descent using Reed-Solomon codes," *arXiv preprint arXiv:1706.05436*, June 2017.

Appendix

A. Independency Condition in Thm. 3

Given $A_1, \cdots, A_{tp'} \in \mathbb{R}^m$, we construct $\tilde{A}_1, \cdots, \tilde{A}_{tp'} \in \mathbb{R}^k$ by appending $(k-m)$ iid Gaussian entries for each A_i's. Let $\tilde{V}_j := \text{span}(\tilde{A}_{(j-1)t+1}, \cdots, \tilde{A}_{jt})$, $(t < m)$ and let $B_{(j-1)(k-t)+1}, \cdots, B_{j(k-t)} \in \mathbb{R}^k$ be a random basis of $\text{null}(\tilde{V}_j) = \tilde{V}_j^{\perp}$. We can choose random basis by multiplying random matrix to a fixed basis of nullspace.

Suppose that for any random choice of B, we can find k linearly dependent column vectors of B. This implies that there exists I such that $(k-t)|I| \geq k$ and $\dim\left(\sum_{i\in I} \tilde{V}_i^{\perp}\right) < k$. By an analog of De Morgan's law for linear spaces, $\sum_{i\in I} \tilde{V}_i^{\perp} = \left(\bigcap_{i\in I} \tilde{V}_i\right)^{\perp}$. From this equality, we have $\dim\left(\bigcap_{i\in I} \tilde{V}_i\right) \geq 1$.

Let $V_j := \text{span}(A_{(j-1)t+1}, \cdots, A_{jt})$. For randomly chosen $\bar{A}$, the inequality always holds only if for any $i \in I$, $\tilde{V}_i$ contains $\{[v\ v_2] \in \mathbb{R}^k : v_2 \in \mathbb{R}^{k-m}\}$ for some $v \in \mathbb{R}^m$.

Therefore, if $\dim\left(\bigcap_{i\in I} \tilde{V}_i\right) \geq 1$, $\{V_i\}$ satisfies either

1) $t - \dim V_i \geq k - m - 1$ and $v \in V_i$ for all $i \in I$, for some nonzero $v \in \mathbb{R}^m$,
2) $t - \dim V_i \geq k - m$. (case when $v = 0$) for all $i \in I$.

Here, $t - \dim V_i$ is equal to the number of rows of the form $[\mathbf{0}_m\ v_2]$ in the reduced row echelon form of $\tilde{A}_{[(i-1)p'+1, ip']}$. If V_i's have a common vector v, we only need $k-m-1$ of the form $[\mathbf{0}_m v_2]$ to span $\{[v\ v_2] \in \mathbb{R}^k : v_2 \in \mathbb{R}^{k-m}\}$. Else, we need $k-m$ rows of such form.

Note that $\dim V_i = \text{rank}(A_{[(i-1)t+1, it]})$. Hence, by contrapositive proposition, B satisfies the MDS property if A satisfies Condition 2. For example, if we can find p' subsets of size t in the rows of A such that each subset has rank at least $t-k+m+2$, we can find an MDS code B with sparsity in Thm. 4.