Leveraging partial stragglers within gradient coding

Aditya Ramamoorthy **Ruoyu Meng** **Vrinda S. Girimaji**
Department of Electrical and Computer Engineering
Iowa State University
Ames, IA 50010.
`{adityar, rmeng, vrindasg}@iastate.edu`

Abstract

Within distributed learning, workers typically compute gradients on their assigned dataset chunks and send them to the parameter server (PS), which aggregates them to compute either an exact or approximate version of ∇L (gradient of the loss function L). However, in large-scale clusters, many workers are slower than their promised speed or even failure-prone. A gradient coding solution introduces redundancy within the assignment of chunks to the workers and uses coding theoretic ideas to allow the PS to recover ∇L (exactly or approximately), even in the presence of stragglers. Unfortunately, most existing gradient coding protocols are inefficient from a computation perspective as they coarsely classify workers as operational or failed; the potentially valuable work performed by slow workers (partial stragglers) is ignored. In this work, we present novel gradient coding protocols that judiciously leverage the work performed by partial stragglers. Our protocols are efficient from a computation and communication perspective and numerically stable. For an important class of chunk assignments, we present efficient algorithms for optimizing the relative ordering of chunks within the workers; this ordering affects the overall execution time. For exact gradient reconstruction, our protocol is around $2\times$ faster than the original class of protocols and for approximate gradient reconstruction, the mean-squared-error of our reconstructed gradient is several orders of magnitude better.

1 Introduction

Large scale distributed learning is the workhorse of modern day machine learning (ML) algorithms. The sheer size of the data and the corresponding computation needs, necessitate the usage of huge clusters for the purpose of parameter fitting in most ML problems of practical interest: deep learning [1], low-rank matrix completion [2] etc. A typical scenario consists of a dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{\tilde{N}}$ of $\tilde{N}$ data points, where $\mathbf{x}_i$'s and y_i's are the features and labels respectively. We wish to minimize a loss function $L = \frac{1}{\tilde{N}} \sum_{i=1}^{\tilde{N}} l(\mathbf{x}_i, y_i, \mathbf{w})$ with respect to $\mathbf{w} \in \mathbb{R}^d$ ($\mathbf{w}$: parameter vector, l: prediction error). When $\mathcal{D}$ is large, we can perform the learning task in a distributed manner [3].

Background: We partition $\mathcal{D}$ into N equal-sized chunks denoted $\mathcal{D}_i, i \in [N]$ ($[n]$ denotes the set $\{1, \dots, n\}$), where a chunk is a subset of the data points and distinct chunks are disjoint. Within each chunk, the assignment of the data points to the workers is identical. Suppose that there are m workers $W_1, \dots, W_m$ and a parameter server (PS). We distribute the chunks to the different workers and let them compute the gradients on the data points assigned to them. The PS coordinates the training by aggregating the (partial) gradients from the workers and transmitting an updated parameter vector to the workers at each iteration. In the "baseline" scheme, $N = m$, W_j is assigned $\mathcal{D}_j$ and it computes $\sum_{i \in \mathcal{D}_j} \nabla l(\mathbf{x}_i, y_i, \mathbf{w}_t)$ (a vector of length-d) and sends it to the PS. Using these, the PS computes the desired gradient $\nabla L = \frac{1}{\tilde{N}} \sum_{i=1}^{\tilde{N}} \nabla l(\mathbf{x}_i, y_i, \mathbf{w}_t)$ and the updated parameter $\mathbf{w}_{t+1}$

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

thereafter. Unfortunately, in many large scale clusters, workers are often slower than their promised speed or even prone to failure. This is especially true in cloud platforms, where the load often fluctuates depending on system load and spot instance pricing [4] explicitly builds in the possibility of job preemption. To address these issues, gradient coding (GC) introduced in [5] incorporates redundancy within the assignment of chunks to the workers. Once a given worker calculates the gradient on all its assigned chunks, it computes a specified linear combination of these gradients and sends it to the PS. An exact gradient coding solution allows the PS to exactly recover ∇L even in the presence of limited node failures.

Let $A \in \mathbb{R}^{N \times m}$ be a matrix such that $A_{i,j} \neq 0$ if and only if $\mathcal{D}_i$ is assigned to W_j; henceforth, we call this the assignment matrix. Let $nnz(v)$ denote the number of non-zero entries in a vector v. Let γ_i and δ_j denote $nnz(A(i,:)$ and $nnz(A(:,j))$ (using MATLAB notation). These correspond respectively to the number of times $\mathcal{D}_i$ appears in the cluster (replication factor) and the number of chunks assigned to W_j (load factor). We assume that at most s workers out of m are stragglers. Let $g_{\mathcal{D}_j} = \sum_{i \in \mathcal{D}_j} \nabla l(\mathbf{x}_i, y_i, \mathbf{w}_t)$, so that $\nabla L = \sum_{j=1}^{N} g_{\mathcal{D}_j}$. At iteration t, worker W_j calculates $g_{\mathcal{D}_i}$ for all $i \in \text{supp}(A(:,j))$ (non-zero entries in $A(:,j)$) and linearly combines them to obtain $g_j = \sum_{i=1}^{N} A_{i,j} g_{\mathcal{D}_i}$. It subsequently transmits g_j to the PS. For decoding ∇L, the PS picks a decoding vector r which is such that $r_j = 0$ if W_j has not transmitted g_j (we say that W_j is straggling in this case). It subsequently calculates

$$\sum_{j=1}^{m} r_j g_j = \sum_{i=1}^{N} \left(\sum_{j=1}^{m} A_{i,j} r_j \right) g_{\mathcal{D}_i}. \tag{1}$$

Related Work: Under the original GC model [5], for *exact gradient coding*, we want that $Ar = \mathbb{1}$ (the all-ones vector) under any choice of at most s stragglers. This means that the PS can perform a full gradient update. For exact GC, we need $\gamma_i \geq s + 1$ for all $i \in [N]$. This setting has been studied extensively [6, 7, 8, 9, 10, 11]. *Approximate gradient coding* [6] considers the scenario where the full gradient is either not required (e.g., SGD [12] works with an inexact gradient) or too costly to work with because of the high replication factor needed. In this setting, we want to design A such that $||Ar - \mathbb{1}||_2$ (ℓ_2-norm) is small over all possibilities for the straggling workers. Prior work demonstrates constructions from expander graphs [6], sparse random graphs [13] and [14], block designs [15, 16] and the like. Within distributed training, a significant time cost is associated with the transmission of the computed gradients (vectors of length-d) by the workers to the PS [17, 18], e.g., deep learning usually operates in the highly over-parameterized regime [19] ($d \gg \tilde{N}$). For exact GC, if $\gamma_i \geq s + \ell$ for $i \in [N]$, then the dimension of the transmitted vectors from the workers can be lowered to d/ℓ [20, 21], thus saving on communication time. This is referred to as *communication-efficient GC* and allows us to trade-off communication for computation. However, both [20] and [21] use polynomial interpolation in their solution. This leads to significant numerical instability [22] to the extent that their solution is essentially unusable for systems with twenty or more workers. This point is also acknowledged within the papers: Section V of [21] and Section II of [20]. Some work considering these issues appears in [23] under restrictive parameter assumptions.

The usage of the partial work performed by stragglers has been considered [10, 24, 25, 26, 27, 28] only within exact GC (i.e., approximate GC has not been considered); some of these apply in the communication-efficient setting. However, these approaches use multiple messages from the workers to the PS and thus incur a higher communication cost per iteration.

Motivation: Consider Fig. 1a where the edge labels indicate the encoded gradients. Note that under ideal operating conditions when each worker operates at the same speed, the overall gradient can be computed as long as each W_i processes its first chunk $\mathcal{D}_i$ for $i = 1, \ldots, 3$. Thus, it is quite wasteful for the workers to continue processing their second assigned chunk as specified in the original GC scheme; it would double the computation time. In addition, the original GC formulation ignores partial work by slow but not failed workers; in the sequel, we refer to these as "partial stragglers". For instance, in Fig. 1a, we consider a scenario, where W_1 is slow and W_3 is failed. The state of the computation is such that there is enough information for the PS to obtain the full gradient. However, under the original GC model, W_1 will either wait until it processes $\mathcal{D}_2$ before generating the encoded gradient to be sent to the PS, or the PS will treat W_1 as failed and will have to settle for an approximate gradient.

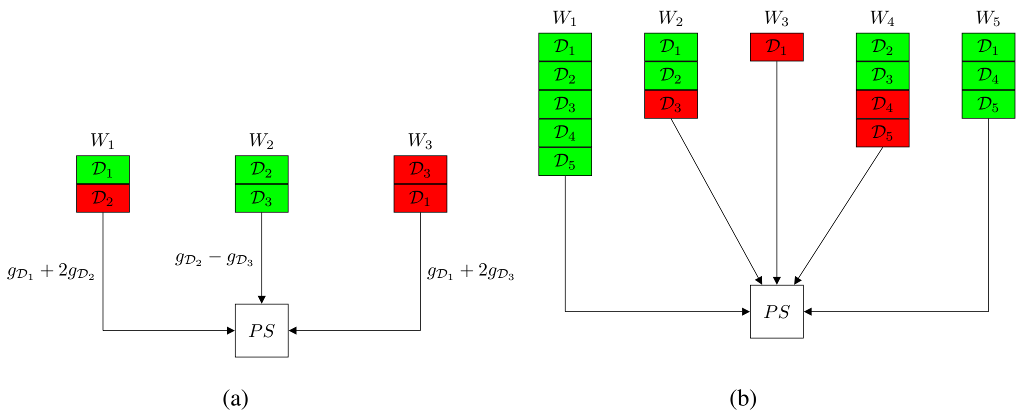


Figure 1: Green/red means that the worker did/didn't process a chunk. (a) System with $N = m = 3$. Each worker is assigned two chunks that they process in a top-to-bottom order. W_3 is failed and W_1 is slow. (b) An arbitrary assignment of chunks to the workers (example also appears in [21]).

Gradient coding can be viewed as an application of coding-theoretic techniques [29] to distributed learning; it allows the PS to be resilient to failures with very little coordination in the cluster. We note here that within classical coding theory [29] most constructions of erasure codes do not consider feedback from the receiver to the sender since such feedback may be expensive or noisy when the sender and receiver are at remote locations or not even necessary ([30], Chap. 7). However, feedback is quite easy to implement in the distributed learning setup.

Main Contributions: We present a new GC protocol that exploits a small amount of additional interactive communication to/from the PS and the workers. It greatly improves the computation-efficiency and communication-efficiency of GC while continuing to allow for efficient coordination within the cluster. Specifically, our protocol efficiently leverages the chunks processed by partial stragglers. Prior work that potentially applies in our setting suffers from the serious problem of numerical instability. In contrast, our protocol is provably numerically stable.

To our best knowledge, despite the importance of communication-efficiency, there are hardly any schemes for communication-efficient approximate GC. Our protocol provides an elegant way to address this problem, which in addition allows the PS to obtain an accurate estimate of the mean-square-error of the reconstructed gradient at any given point in the computation.

Prior work in the GC area ignores the relative ordering of the chunks within the workers. As our protocol leverages partial work performed by the workers, the relative ordering of the chunks within the workers is an important factor in the performance of the algorithm. For a large class of assignment matrices, we present an efficient polynomial-time algorithm that returns the optimal ordering of the chunks within workers.

2 Gradient Coding for partial stragglers

Our GC protocol operates under the following assumptions: (i) the workers know the assignment matrix and the ordering of the chunks within all the workers, (ii) at regular intervals the workers keep communicating to the PS, the number of chunks they have processed, and (iii) the PS wants the workers to communicate vectors of length d/ℓ for integer $\ell \geq 1$.

We now provide a top-level description of our protocol; a formal statement appears in Algorithm 1. At the beginning of the training process, the PS generates a random $\ell \times m$ matrix $\mathbf{R}$ with i.i.d. entries from a standard normal distribution, $N(0, 1)$ and shares it with all the workers. The workers keep reporting the number of chunks that they have processed in an iteration. The PS keeps monitoring these counts, and at a certain point, it broadcasts to all the workers an "encode-and-transmit" signal and an integer vector ψ of length-m that specifies the number of chunks that have been processed by each node. This shares the global system state amongst the workers. For exact GC, the PS sends the encode-and-transmit signal when at least ℓ copies of each $\mathcal{D}_i$ have been processed across the cluster. For approximate GC, the PS can send the signal even earlier.

Algorithm 1 Find-Encoding-Coeff

```
Input: $\ell \times m$ matrix $\mathbf{R}$, $m$-length state vector $\psi$, $\delta_i$ the number of chunks processed by $W_i$.
Output: Encoding coefficients for the $i$-th worker $\varepsilon_i$.
1: $W_i$ forms the indeterminate matrix $m \times N\ell$ matrix $\mathbf{B}$ based on $\psi$.
2: $W_i$ extracts the columns of $\mathbf{B}$ that correspond to its processed chunks. This matrix is called $\tilde{\mathbf{B}}_i$
   (cf. (4)).
3: Worker $W_i$ solves the following minimum-$\ell_2$-norm least-squares problem, where the variables
   are the indeterminates in $\tilde{\mathbf{B}}_i$.
       $\min ||\mathbb{1}_{\delta_i}^T \otimes I_\ell - \mathbf{R}\tilde{\mathbf{B}}_i||_2.$    (5)
4: Set $\varepsilon_i = \mathbf{B}(i,:)$.
```

Following this, the workers need to decide their own encoding coefficients for the gradients that they have computed. Each gradient $g_{\mathcal{D}_i}$ is block-partitioned into ℓ disjoint parts $g_{\mathcal{D}_i}[k], k = 0, \ldots, \ell-1$. Each worker forms a matrix $\mathbf{B}$ of dimension $m \times N\ell$ that consists of indeterminates that specify the encoding coefficients of all the workers (see example in the upcoming Section 2.2). Matrix $\mathbf{B}$ consists of N block-columns, where each block-column itself consists of ℓ columns of length-m, i.e., $\mathbf{B} = [\mathbf{B}^{(1)} \mid \mathbf{B}^{(2)} \mid \ldots \mid \mathbf{B}^{(N)}]$. The j-th block column, $\mathbf{B}^{(j)}$ is associated with $g_{\mathcal{D}_j}[k], k = 0, \ldots, \ell-1$. Based on the global state vector ψ, all workers know whether a chunk $\mathcal{D}_j, j \in [N]$ has been processed by worker $W_i, i \in [m]$. If $\mathcal{D}_j$ has been processed by W_i, then the i-th row of $\mathbf{B}^{(j)}$ is populated with indeterminates, otherwise these entries are set to zero. Once the indeterminates are found (see Algorithm 1 for a description), worker W_β encodes its gradients as

$$g_\beta = \sum_{i=1}^{N} \sum_{j=0}^{\ell-1} \mathbf{B}_{\beta,j}^{(i)} g_{\mathcal{D}_i}[j]. \tag{2}$$

Let $\vec{e}_i$ denote the i-th canonical basis vector of length-ℓ, i.e., it contains a 1 at location $i \in \{0, \ldots, \ell-1\}$ and zeros elsewhere. We denote

$$z_i = \mathbb{1}_N \otimes \vec{e}_i = \overbrace{[\vec{e}_i^T \; \vec{e}_i^T \; \ldots \; \vec{e}_i^T]^T}^{N \text{ copies of } \vec{e}_i}. \tag{3}$$

where $\otimes$ denotes the Kronecker product and $\mathbb{1}_N$ denotes the all-ones vector of length N. Recall that in the exact GC scenario, the PS wants to obtain $\sum_{i=1}^N g_{\mathcal{D}_i}[k]$ for $k = 0, \ldots, \ell-1$. This is equivalent to requiring that

$$z_i^T \in \text{row-span}(\mathbf{B}), \text{ for } i = 0, \ldots, \ell-1.$$

Our protocol is such that each W_i can independently calculate their encoding coefficients, such that collectively all the workers agree on the same matrix $\mathbf{B}$ with the same values assigned to all the indeterminates. Towards this end, let $\tilde{\mathbf{B}}_j$ denote the submatrix of $\mathbf{B}$ that is relevant to W_j, i.e., the block-columns in which the processed chunks of W_j participate. For instance, suppose that W_j has processed $\alpha_j \leq \delta_j$ chunks $\mathcal{D}_{i_1}, \ldots, \mathcal{D}_{i_{\alpha_j}}$ where $1 \leq i_1 < i_2 < \cdots < i_{\alpha_j} \leq N$. Then,

$$\tilde{\mathbf{B}}_j = [\mathbf{B}^{(i_1)} \, \mathbf{B}^{(i_2)} \, \ldots \, \mathbf{B}^{(i_{\alpha_j})}]. \tag{4}$$

W_j then solves a minimum-ℓ_2-norm least-squares solution to determine its encoding coefficients (see (5) in Algorithm 1). This ensures the solution is unique [31] even if the corresponding problem is under-determined. Thus, the workers automatically agree on the same solution.

Remark 1. *If $\ell = 1$, then it is possible to arrive at a protocol whereby the workers agree to transmit appropriately weighted partial sums of their gradients, so that the PS can exactly/approximately recover the sum. However, in the communication-efficient setting when $\ell > 1$, the encoding is no longer straightforward. Thus, $\ell > 1$ is the main scenario we consider in what follows.*

Remark 2. *We discussed that each worker can form the $m \times N\ell$ matrix of indeterminates $\mathbf{B}$ for the sake of ease of explanation. In fact, W_j only works with $\tilde{\mathbf{B}}_j$, which is of size at most $m \times \delta\ell$.*

Remark 3. *The additional communication assumed in our protocol is only $O(m)$ as against the parameter vector of length-d and $O(m) \ll d$. Thus, the additional communication cost of our algorithm is minimal.*

2.1 Analysis of Algorithm 1

It is evident from our description and Algorithm 1 that upon receiving the encode-and-transmit signal and the vector ψ, each worker can independently create the matrix of indeterminates $\mathbf{B}$.

Exact GC analysis: Suppose that each $\mathcal{D}_i, i \in [N]$ has been processed at least ℓ times across the cluster, and suppose that W_i has processed $\mathcal{D}_j$. This means that there is a block-column $\mathbf{B}^{(j)}$ such that each column within $\mathbf{B}^{(j)}$ has at least $\Delta \geq \ell$ indeterminates that need to be assigned values. For one column within $\mathbf{B}^{(j)}$, W_i will solve a system of equations that is specified by $\ell \times \Delta$ submatrix of $\mathbf{R}$ denoted X (an example appears in Section 2.2). Note that the columns of X will be linearly independent with high probability owing to the choice of $\mathbf{R}$ and since $\Delta \geq \ell$, a solution is guaranteed to exist. Let $\kappa_2(M)$ denote the condition number of matrix M. For a random $\ell \times \Delta$ matrix with i.i.d. $N(0,1)$ entries, it is known that $\mathbb{E}(\log \kappa_2(X)) \leq O(\log \Delta)$ [32]. Thus, each such system of equations is well-conditioned with very high probability. In the under-determined case when $\Delta > \ell$, each worker will still agree on the same values of the corresponding indeterminates since we enforce that we work with the (unique) minimum ℓ_2-norm solution; no additional communication between the workers is required.

Approximate GC analysis: It is possible that the PS sends the encode-and-transmit vector when some $\mathcal{D}_i$ has been processed $\Delta \leq \ell - 1$ times. In this case, the corresponding step for $\mathbf{B}^{(i)}$ in (5) will be an over-determined least-squares procedure, which implies that there will be a non-zero error associated with it. Let X be the relevant $\ell \times \Delta$ submatrix of $\mathbf{R}$ where we have $\Delta < \ell$ now, and recall that all entries of X are i.i.d. $N(0,1)$ random variables. The squared error corresponding to $\mathbf{B}^{(i)}$ can be expressed as $\sum_{i=0}^{\ell-1} ||XX^\dagger \vec{e}_i - \vec{e}_i||_2^2$ ($X^\dagger$ denotes the pseudo-inverse [31]). Let $X = USV^T$ denote the SVD of X, where U and V are orthogonal matrices of dimension $\ell \times \ell$ and $\Delta \times \Delta$ respectively and $S = [D \mid 0]^T$ where D is a $\Delta \times \Delta$ matrix with positive entries on the diagonal, and 0 represents a $\Delta \times (\ell - \Delta)$ matrix of zeros. Then, $X^\dagger = V[D^{-1} \mid 0]U^T$. It is well known ([33], Remark 5.2.8) that for a matrix with i.i.d. $N(0,1)$ entries, the singular vectors are uniformly distributed on the unit-sphere. Therefore, the expected squared error becomes

$$\mathbb{E}[||XX^\dagger \vec{e}_i - \vec{e}_i||_2^2] = \mathbb{E}[||\sum_{j=\Delta+1}^{\ell} u_j u_j^T \vec{e}_i||_2^2] = \sum_{j=\Delta+1}^{\ell} \mathbb{E}[(u_j^T \vec{e}_i)^2] = \frac{\ell - \Delta}{\ell}. \tag{6}$$

The last step above follows since each u_j is uniformly distributed over the sphere of dimension ℓ. Therefore, we have the $\mathbb{E}[u_{j,0}^2] = 1/\ell$ since $||u_j||_2^2 = 1$ and each $u_{j,k}, k = 0, \ldots, \ell - 1$ is identically distributed. We conclude that if $\mathcal{D}_i$ appears $\Delta_i \leq \ell - 1$ times, then its error contribution is $\ell - \Delta_i$ and the overall error is therefore $\sum_{i=1}^{N} \max(0, \ell - \Delta_i)$.

Complexity analysis: The time complexity of each least-squares problem is $O(\Delta^2 \ell)$ [34] and the i-th worker solves at most $\ell\delta_i$ of them independently and in parallel. The marginal cost of this calculation as against the calculation of the actual gradients will be very small in most practical settings.

2.2 Illustrative Example

Consider Fig. 1b (from [21]) where the dataset consists of chunks $\mathcal{D}_1, \ldots, \mathcal{D}_5$ and are assigned in a non-uniform fashion to workers $W_1, \ldots, W_5$. Suppose that the PS wants the exact gradient with $\ell = 2$. As two copies of each chunk have been processed, the PS broadcasts the encode-and-transmit signal and the vector $\psi = [5\ 2\ 0\ 2\ 3]$ to all the workers which indicates, e.g., that W_1 has processed all its chunks, W_3 is failed etc. The matrix $\mathbf{B}$ of indeterminates for this example and the corresponding $\mathbf{B}^{(i)}$'s, turn out to be

$$\mathbf{B} = [\mathbf{B}^{(1)} | \mathbf{B}^{(2)} | \mathbf{B}^{(3)} | \mathbf{B}^{(4)} | \mathbf{B}^{(5)}] \tag{7}$$

$$= \left[\begin{array}{cc|cc|cc|cc|cc} a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 & a_8 & a_9 & a_{10} \\ b_1 & b_2 & b_3 & b_4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & c_3 & c_4 & c_5 & c_6 & 0 & 0 & 0 & 0 \\ d_1 & d_2 & 0 & 0 & 0 & 0 & d_7 & d_8 & d_9 & d_{10} \end{array}\right]. \tag{8}$$

In this example, W_5 has processed $\mathcal{D}_1, \mathcal{D}_4, \mathcal{D}_5$ so that $\tilde{B}_5 = [\mathbf{B}^{(1)}\ \mathbf{B}^{(4)}\ \mathbf{B}^{(5)}]$. Note that the PS requires the vectors $[\vec{e}_0^T\ \vec{e}_0^T\ \vec{e}_0^T\ \vec{e}_0^T\ \vec{e}_0^T]$ and $[\vec{e}_1^T\ \vec{e}_1^T\ \vec{e}_1^T\ \vec{e}_1^T\ \vec{e}_1^T]$ to lie in the row-space of $\mathbf{B}$ so that it

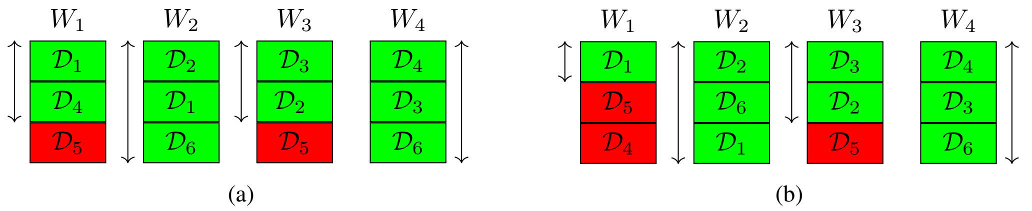


Figure 2: Two different relative orderings of the chunks within workers for the same assignment matrix. Individual figures show the calculation of Q_5. Similarly, other Q_i values can be computed. $Q_{\max} = Q_5$ for both assignments. Thus, (a) $Q_{\max} = 10$. (b) $Q_{\max} = 9$.

can recover $\sum_{i=1}^{5} g_{\mathcal{D}_i}[k], k = 0, 1$ from the encoded gradients. Towards this end, e.g., W_5 solves (5) in Algorithm 1 with the matrix $\tilde{B}_5$, i.e.,

$$\min ||[\mathbf{I}_2|\mathbf{I}_2|\mathbf{I}_2] - \mathbf{R}[\mathbf{B}^{(1)}\ \mathbf{B}^{(4)}\ \mathbf{B}^{(5)}]||_2 \tag{9}$$

where the decision variables are $a_1, a_2, a_7, a_8, a_9, a_{10}, b_1, b_2$ and $d_1, d_2, d_7, d_8, d_9, d_{10}$. For instance, corresponding to the first column of $\mathbf{B}^{(1)}$, W_5's problem becomes determining the minimum ℓ_2-norm solution of the under-determined least-squares problem

$$\left|\left| \begin{bmatrix} r_{01} & r_{02} & r_{05} \\ r_{11} & r_{12} & r_{15} \end{bmatrix} \begin{bmatrix} a_1 \\ b_1 \\ d_1 \end{bmatrix} - \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right|\right|_2 . \tag{10}$$

We emphasize that the same minimization will be independently performed at workers W_1 and W_2 as well. After each W_j determines its encoding coefficients and transmits g_j for $j = 1, \ldots, 5$, the PS can easily recover $\sum_{j=1}^{5} g_{\mathcal{D}_j}[k] = \sum_{i=1}^{5} r_{kj} g_j$ for $k = 0, 1$.

Algorithm 1 works as is, even in the case when the PS is interested in an approximate gradient. For instance, suppose that the PS sends the encode-and-transmit signal when the vector $\psi = [4\ 2\ 0\ 2\ 3]$, so that only one copy of $\mathcal{D}_5$ has been processed in the cluster. The corresponding encoding coefficients can still be computed using Algorithm 1. The only difference will be that the relevant indeterminate matrix becomes

$$\mathbf{B}' = \left[\begin{array}{cc|cc|cc|cc|cc} a'_1 & a'_2 & a'_3 & a'_4 & a'_5 & a'_6 & a'_7 & a'_8 & 0 & 0 \\ b'_1 & b'_2 & b'_3 & b'_4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & c'_3 & c'_4 & c'_5 & c'_6 & 0 & 0 & 0 & 0 \\ d'_1 & d'_2 & 0 & 0 & 0 & 0 & d'_7 & d'_8 & d'_9 & d'_{10} \end{array}\right]. \tag{11}$$

This means, e.g., when W_5 is trying to find d'_9, then it will be solving an over-determined least-squares problem: $\left|\left| \begin{bmatrix} r_{05} \\ r'_{05} \end{bmatrix} d'_9 - \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right|\right|_2$. It is evident, that all the workers can still agree on the same solution.

3 Chunk Ordering Algorithm

Note that the assignment matrix only specifies the assignment of chunks to workers but says nothing about the relative order of chunks within a worker. When taking into account partial stragglers, the relative ordering of the $\mathcal{D}_i$'s within a worker is crucial. Therefore, an important question when leveraging partial stragglers within GC is one of how to determine this chunk ordering within workers for a given assignment matrix. This will in general depend on models of worker speeds. However, it is a difficult task to obtain accurate models on the worker speeds within a cloud cluster as conditions can change dynamically.

We work instead with a combinatorial metric that depends on the total number of chunks that the cluster has to process in the worst case, such that at least a single copy of each $\mathcal{D}_i$ is processed; this was also used in [35, 36] in a coded matrix computation context. This metric minimizes the worst case number of chunks that the cluster needs to process in the case when $\ell = 1$. In particular, for a given $\mathcal{D}_i$, let Q_i denote the maximum number of chunks that can be processed by the cluster such that none of the copies of $\mathcal{D}_i$ are processed (see Figs. 2a and 2b) and let $Q_{\max} = \max_{i=1,\ldots,N} Q_i$. Thus, $1 + Q_{\max}$ is a metric that quantifies the worst-case number of chunks that need to be processed

Algorithm 2 Chunk-Ordering

```
Input: Assignment matrix $A$ such that $N = m$ and $\gamma_i = \delta_i = \delta$ for all $i \in [m]$
Output: An optimal ordering matrix $O$.
1: Let $\tilde{A}^{(1)} \in \{0,1\}^{m\times m}$ such that $\tilde{A}^{(1)}_{i,j} = \begin{cases} 1 & \text{if } A_{i,j} \neq 0 \\ 0 & \text{otherwise.} \end{cases}$ and $\delta^{(1)} = \delta$.
2: for $i$ ranges from 1 to $\delta$ do
3:     Apply Claim 1 with $\tilde{A}^{(i)}$ and $\delta^{(i)}$ and solve the max-bipartite matching problem in Claim 1.
       Let the permutation matrix be $P_i$.
4:     Let $\tilde{A}^{(i+1)} := \tilde{A}^{(i)} - P_i$ and $\delta^{(i+1)} = \delta^{(i)} - 1$.
5: end for
6: Set $O = \sum_{i\in[\delta]} iP_i$.
```

before at least a single copy of every $\mathcal{D}_i$ is guaranteed to be processed. Here, the worst-case is over the speeds of the different workers. We say that an assignment is optimal if it achieves the lowest possible $Q_{\max}$ for a given assignment matrix. Indeed, Figs. 2a and 2b show two different orderings for the same assignment matrix with different values of $Q_{\max}$.

From the point of view of leveraging partial stragglers, for exact GC it can be shown that the ordering imposed by the cyclic assignment scheme (in [5]) is optimal ((*cf.* Remark 1 in work [35])) for this $Q_{\max}$ metric. However, for approximate GC, when the number of stragglers can be much higher, other assignments, e.g., those based on regular graphs perform better [6, 37]. In particular, in these cases, the assignment matrix A is chosen as the adjacency matrix of the relevant graphs and is such that $N = m$ and $\gamma_i = \delta_i = \delta$ for $i \in [m]$. For such assignment matrices, Algorithm 2 presents an optimal algorithm for determining the ordering of the chunks within each worker in $Q_{\max}$-metric. Corresponding to the assignment matrix A, we can associate an ordering matrix O which is such that $O_{i,j} = 0$ if $A_{i,j} = 0$ and $O_{i,j} \in [\delta]$ otherwise. Thus, if $O_{i,j} = \alpha \neq 0$, it means that $\mathcal{D}_i$ is the α-th chunk in W_j, e.g., $\alpha = 1$ implies that the chunk is at the top and $\alpha = \delta$ means that it is at the bottom. Let $Q_i(O)$ denote the maximum number of chunks that can be processed across the cluster, such that no copy of $\mathcal{D}_i$ has been processed. Then, upon inspection, we can see that

$$Q_i(O) = \sum_{j\in[m]} \mathbf{1}_{\{O_{i,j}\neq 0\}}(O_{i,j} - 1) + (m-\delta)\delta = \sum_{j\in[m]} \mathbf{1}_{\{O_{i,j}\neq 0\}} O_{i,j} + (m-\delta-1)\delta,$$

where the notation $\mathbf{1}_Y$ denotes the indicator of Y. Next, let $Q_{\max}(O) = \max_{i\in[m]} Q_i(O)$. Thus, given an assignment matrix A, our goal is to find an ordering matrix, such that $Q_{\max}(O)$ is minimized. As $(m-\delta-1)\delta$ is a constant, this optimization is equivalent to the following min-max problem.

$$\underset{O}{\text{minimize}} \max_{i\in[m]} \sum_{j\in[m]} O_{i,j}. \tag{12}$$

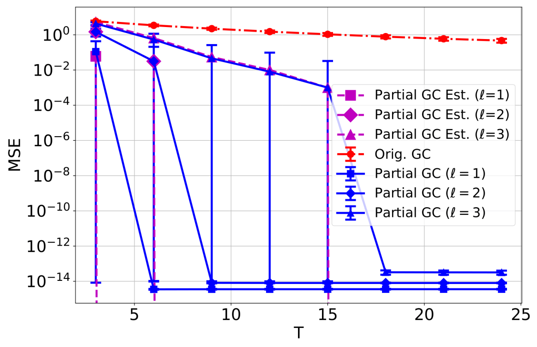
Each column of O has exactly δ non-zero entries, with each value in $[\delta]$ appearing exactly once. Counting the sum of the entries in O two different ways yields

$$m\frac{\delta(\delta+1)}{2} = \sum_{i\in[m],j\in[m]} O_{i,j} \leq mQ_{\max}(O), \text{ so that } Q_{\max}(O) \geq \frac{\delta(\delta+1)}{2}.$$

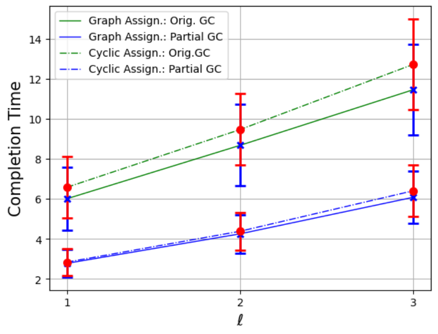
It turns out that this bound is in fact achievable. Let $\tilde{A} = \tilde{A}^{(1)}$ (defined in Algorithm 2). Define $\mathbb{G}(\tilde{A}) = (V, E)$ as a bipartite graph on vertex set $V = X \cup Y$ such that $|X| = |Y| = m, X \cap Y = \emptyset$ and $\deg(v) = \delta$ for all $v \in X \cup Y$. For $u \in X, v \in Y$, we have that $(u, v) \in E$ if and only if $\tilde{A}_{u,v} = 1$. Thus, $\tilde{A}$ is in one-to-one correspondence with $\mathbb{G}(\tilde{A})$. Algorithm 2 decomposes $\mathbb{G}(\tilde{A})$ into a collection of disjoint perfect matchings [38] which are then assigned values in $[\delta]$. This gives us the required ordering.

Claim 1. *Let $\tilde{A} \in \{0,1\}^{m\times m}$ be such that both i-th row sum and j-th column sum of $\tilde{A}$ are δ for all $i, j \in [m]$. Then there exists a permutation matrix $P \in \{0,1\}^{m\times m}$ such that $\tilde{A} - P \in \{0,1\}^{m\times m}$ and both i-th row sum and j-th column sum of $\tilde{A} - P$ are $\delta - 1$ for $i, j \in [m]$.*

Proof. We claim that $\mathbb{G}(\tilde{A})$ has a X-perfect matching. Let $S \subseteq X$ be arbitrary and $N(S) \subseteq Y$ denote its neighborhood. Let κ denote the average degree of the vertices in $N(S)$ in the subgraph



(a)



(b)

Figure 3: (a) Mean-squared error (MSE) vs. T for an approximate GC scenario. Blue curves: proposed protocol with $\ell = 1, 2, 3$, purple curves: corresponding MSE estimates, and red curve: original GC protocol with $\ell = 1$. Error bars correspond to one standard deviation. (b) Completion time vs. ℓ for exact GC scenario with two different assignment matrices. Blue curves: proposed protocol, green curves: original GC protocol. Error bars correspond to one standard deviation.

induced by $S \cup N(S)$. Then, we have

$$\delta|S| = \kappa|N(S)| \leq \delta|N(S)|, \text{ so that } |S| \leq |N(S)|.$$

The first inequality above follows because the degree of each node in $N(S)$ in $\mathbb{G}(\tilde{A})$ is δ. Thus, Hall's condition [39] holds and the claim follows. Since $|X| = |Y|$, this is actually a perfect matching M. Then, this perfect matching M gives the desired P: $P_{u,v} = 1$ if $(u, v) \in M$ and $P_{u,v} = 0$ if $(u, v) \notin M$. Removing the matching M from $\mathbb{G}(\tilde{A})$ results in a bi-regular bipartite graph with degree $\delta - 1$ which corresponds to $\tilde{A} - P$. □

Remark 4. *The proposed algorithm above finds the optimal ordering when considering the case of $\ell = 1$ with $N = m$ (number of chunks equal to number of workers); it only applies when $N = m$. While this algorithm is expected to have better performance than a randomly chosen ordering in the case of higher ℓ, we do not have an optimal construction in this case.*

Algorithm 2 Analysis: The algorithm gives δ permutation matrices $\{P_i\}_{i\in[\delta]}$ such that $\tilde{A} = \sum_{i=1}^{\delta} P_i$, i.e., the set of non-zero entries of the P_i's are disjoint. Since $O = \sum_{i\in[\delta]} i \cdot P_i$, this implies that each column has exactly δ non-zero entries such that these entries consists of elements of $[\delta]$. Therefore, O is an ordering matrix. In addition, each row has δ non-zero elements from $[\delta]$, so that all row sums and $Q_{\max}(O)$ equal $\frac{\delta(\delta+1)}{2}$. A maximum matching can be in time $O(m^2\delta)$ by converting it to a max-flow problem [38], so our overall complexity is $O(m^2\delta^2)$. The complexity can potentially be reduced further by using more efficient max-flow algorithms.

4 Numerical Experiments and Comparisons

Our proposed protocol in Section 2, utilizes additional communication between the PS and the workers. We note here that ideas in [21] that are based on Lagrange interpolation can potentially be adapted to arrive at an exact GC protocol within our setting. However, the numerical instability of Lagrange interpolation is a serious issue with this solution, since it can be shown that the degree of the polynomial to be interpolated is at least $m - \ell$. Even for values such as $\ell = 2$ and $m = 22, 27, 32$, the error in Lagrange interpolation is too high for the technique to be useful (see Appendix A).

In what follows, we present comparisons with the original GC protocol via a simulation of the distributed system for both the exact and approximate GC scenarios. All software code for recreating these results can be found at [40]. These simulations were performed within a MacBook Pro (M1 chip, 16 GB RAM). In both scenarios, we simulated node failures and slow-downs. We generated a random vector of length-m where α workers uniformly at random are chosen to be failed (α is chosen based on the scenario). For the other workers, the amount of time taken to process a chunk is chosen i.i.d. from an exponential distribution with parameter $\mu = 1$. The entries of the matrix $\mathbf{R}$ are chosen i.i.d. from the standard normal $N(0, 1)$.

Approximate GC simulation: We considered two different random regular graphs, $G_i, i = 1, 2$ with sizes $m = 200, 300$ and degree $\nu = 8$. These graphs have second-largest absolute value of eigenvalues: 5.14 and 5.23, i.e., less than $2\sqrt{\nu - 1}$ so they can be considered as Ramanujan graphs [41] (these were used in [6]). Their adjacency matrices were used as the assignment matrix. The number of failures α was set to $\nu - 1$.

For the original GC protocol with arbitrary assignment matrices, [21] provides a communication-efficient approximate GC method that relies on rational interpolation. However, even for $m = 100$, these will result in very complex basis functions. Furthermore, there are no numerical results in [21] (or posted software) and the approximation guarantees depend on the form of the function to be interpolated, i.e., the guarantees cannot be expressed in terms of the system parameters. Thus, in our comparison, we only consider the original GC protocol with $\ell = 1$.

For the original GC protocol, we compute the least squares solution $\hat{r}$ for minimizing $||Ar - \mathbb{1}||_2^2$. Here, $\hat{r}$ such that $\hat{r}_i = 0$ if W_i has not completed processing all its chunks at time T. For our proposed protocol, we leverage partial work completed by T, as described in Section 2. The overall error is computed as the sum of the errors for the least-squares solution for each $z_i^T, i = 0, \dots, \ell - 1$. Each data point on the curves was chosen by performing 1000 simulations of failures and slow-downs. We have also plotted the expected value of the error, derived in (6).

Fig. 3a shows the mean-squared-error (MSE) comparing the original GC approach with $\ell = 1$ and our partial GC approach for $\ell = 1, 2, 3$ for graph G_1 (see Appendix B for G_2 results). As can be observed, the MSE for our approach is several orders of magnitude lower with increasing T. Crucially, our estimate (6) closely tracks the behavior of the error of our method. Thus, it can easily be used by the PS as a way to decide when to send the encode-and-transmit message. We emphasize that our approach, even with $\ell \geq 2$ actually has a lower MSE than the original approach (that operates with $\ell = 1$). Note that with $\ell \geq 2$, we will enjoy a lower communication time in a real-world distributed cluster. However, as we are working with a simulation of the distributed system, at this time we do not have precise figures for the reduction in communication time on actual clusters.

In Fig. 6 in Appendix B, we compare the performance of our approach under the optimal ordering (*cf.* Section 3) and an appropriately chosen random ordering. The random ordering is picked as follows. We generated 100 independent random orderings and selected the one with the best (smallest) $Q_{\max}(O)$. Our optimal ordering, which can be found efficiently, clearly has a better performance.

Exact GC simulation: We considered (i) the cyclic assignment [5] with $N = m = 200$ and $\delta = 8$, and (ii) the assignment based on graph G_1 discussed above. The number of failures α in the simulations is set to $\delta - \ell$ so that exact gradient reconstruction is possible. For both approaches, we determined the time T such that each chunk is processed at least ℓ times across the cluster (for original GC we only consider workers that have processed all their chunks). These values were averaged over 1000 runs for each value of ℓ. In Fig. 3b we clearly observe that the exact gradient can be computed using our method is approximately half the time as compared to the original GC protocol. We note that the average completion time for the original GC protocol (G_1-based assignment) for $\ell = 1$ is about $T = 6$ time units. However, the MSE for the original GC protocol in the approximate scenario continues to be high at $T = 24$. The reason is that there are $\nu - 1$ failures that are introduced in the simulation. The approximate GC recovery operates by solving a least-squares problem for the fixed G_1-based assignment. Thus, the MSE does not necessarily drop to zero even if one copy of each chunk has been processed in the cluster. However, there are exact recovery algorithms that one can use in this case. We note here that when $\ell \geq 2$, the known techniques for exact recovery for the original GC protocol are based on Lagrange interpolation and are numerically unstable. Thus, the gradient recovered using the original approach will in general not be useful.

Fig. 7 in Appendix B compares the completion times of random vs. optimal ordering; the optimal ordering is clearly better.

5 Limitations of our work

Our chunk ordering algorithm (Section 3) is optimal only in the case when the assignment matrix has $N = m$ and $\gamma_i = \delta_i = \delta$ for $i \in [m]$. While several well known gradient coding schemes, especially those from regular graphs, satisfy this property, there are others that do not. Our results in Section 4 with $\ell \geq 2$ are simulations of the actual distributed cluster and indicate lower MSE than the original GC protocol. However, we do not have actual cloud platform statistics on the reduction in communication time within our method. We do however expect this reduction to be quite significant.

6 Conclusions

We presented a novel gradient coding protocol that leverages the work performed by partial stragglers. Our protocol is simultaneously computation-efficient and communication-efficient, and numerically stable. Furthermore, we present rigorous analyses of the protocol's correctness and performance guarantees. Our protocol is one of the first to provide a satisfactory solution to the problem of communication-efficient, approximate gradient coding for general assignment matrices.

Acknowledgments and Disclosure of Funding

We would like to the anonymous reviewers and area chair whose helpful comments and suggestions helped improve the paper. This work was supported in part by the National Science Foundation (NSF) under grant CCF-2115200.

References

[1] Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, Cambridge, MA, USA, 2016. `http://www.deeplearningbook.org`.

[2] Prateek Jain, Praneeth Netrapalli, and Sujay Sanghavi. Low-rank matrix completion using alternating minimization. In *Proceedings of the forty-fifth Annual ACM Symposium on Theory of Computing (STOC)*, pages 665–674, 2013.

[3] Mu Li, David G Andersen, Jun Woo Park, Alexander J Smola, Amr Ahmed, Vanja Josifovski, James Long, Eugene J Shekita, and Bor-Yiing Su. Scaling distributed machine learning with the parameter server. In *11th USENIX Symposium on operating systems design and implementation (OSDI 14)*, pages 583–598, 2014.

[4] Google Spot Virtual Machines.

[5] Rashish Tandon, Qi Lei, Alexandros G. Dimakis, and Nikos Karampatziakis. Gradient coding: Avoiding stragglers in distributed learning. In *Intl. Conf. Mach. Learn. (ICML)*, pages 3368–3376, August 2017.

[6] Netanel Raviv, Rashish Tandon, Alex Dimakis, and Itzhak Tamo. Gradient coding from cyclic MDS codes and expander graphs. In *Intl. Conf. Mach. Learn. (ICML)*, pages 4302–4310, July 2018.

[7] Wael Halbawi, Navid Azizan, Fariborz Salehi, and Babak Hassibi. Improving distributed gradient descent using reed-solomon codes. In *IEEE Intl. Symp. on Info. Th.*, pages 2027–2031, 2018.

[8] Tianyi Chen, Georgios Giannakis, Tao Sun, and Wotao Yin. Lag: Lazily aggregated gradient for communication-efficient distributed learning. *Adv. in Neural Inf. Process. Syst. (NeurIPS)*, 31, 2018.

[9] Shanuja Sasi, V. Lalitha, Vaneet Aggarwal, and B. Sundar Rajan. Straggler mitigation with tiered gradient codes. *IEEE Trans. on Comm.*, 68(8):4632–4647, 2020.

[10] Baturalp Buyukates, Emre Ozfatura, Sennur Ulukus, and Deniz Gündüz. Gradient coding with dynamic clustering for straggler-tolerant distributed learning. *IEEE Trans. on Comm.*, 71(6):3317–3332, 2023.

[11] Neophytos Charalambides, Hessam Mahdavifar, and Alfred O. Hero. Numerically stable binary gradient coding. In *IEEE Intl. Symp. on Info. Th.*, pages 2622–2627, 2020.

[12] Léon Bottou, Frank E. Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311, May 2018.

[13] Zachary Charles, Dimitris Papailiopoulos, and Jordan Ellenberg. Approximate gradient coding via sparse random graphs. 2017 [Online] arxiv:1711.06771.

[14] Margalit Glasgow and Mary Wootters. Approximate gradient coding with optimal decoding. *IEEE J. Select. Areas Info. Th.*, 2(3):855–866, 2021.

[15] Swanand Kadhe, O. Ozan Koyluoglu, and Kannan Ramchandran. Gradient coding based on block designs for mitigating adversarial stragglers. In *IEEE Intl. Symp. on Info. Th.*, pages 2813–2817, 2019.

[16] Animesh Sakorikar and Lele Wang. Soft bibd and product gradient codes. *IEEE J. Select. Areas Info. Th.*, 3(2):229–240, 2022.

[17] Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. Qsgd: Communication-efficient sgd via gradient quantization and encoding. *Adv. in Neural Inf. Process. Syst. (NeurIPS)*, 30, 2017.

[18] Jue Wang, Yucheng Lu, Binhang Yuan, Beidi Chen, Percy Liang, Christopher De Sa, Christopher Re, and Ce Zhang. Cocktailsgd: Fine-tuning foundation models over 500mbps networks. In *Intl. Conf. Mach. Learn. (ICML)*, pages 36058–36076, 2023.

[19] Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, July 2019.

[20] Min Ye and Emmanuel Abbe. Communication-computation efficient gradient coding. In *Intl. Conf. Mach. Learn. (ICML)*, pages 5610–5619, 2018.

[21] Tayyebeh Jahani-Nezhad and Mohammad Ali Maddah-Ali. Optimal communication-computation trade-off in heterogeneous gradient coding. *IEEE J. Select. Areas Info. Th.*, 2(3):1002–1011, 2021.

[22] V. Pan. How Bad Are Vandermonde Matrices? *SIAM Jour. on Mat. Analysis and Appl.*, 37(2):676–694, 2016.

[23] Swanand Kadhe, O. Ozan Koyluoglu, and Kannan Ramchandran. Communication-efficient gradient coding for straggler mitigation in distributed learning. In *IEEE Intl. Symp. on Info. Th.*, pages 2634–2639, 2020.

[24] Lev Tauz and Lara Dolecek. Multi-message gradient coding for utilizing non-persistent stragglers. In *2019 53rd Asilomar Conference on Signals, Systems, and Computers*, pages 2154–2159, 2019.

[25] Hankun Cao, Qifa Yan, Xiaohu Tang, and Guojun Han. Adaptive gradient coding. *IEEE/ACM Trans. on Networking*, 30(2):717–734, 2022.

[26] Haozhao Wang, Song Guo, Bin Tang, Ruixuan Li, Yutong Yang, Zhihao Qu, and Yi Wang. Heterogeneity-aware gradient coding for tolerating and leveraging stragglers. *IEEE Trans. on Comp.*, 71(4):779–794, 2022.

[27] Qi Wang, Ying Cui, Chenglin Li, Junni Zou, and Hongkai Xiong. Optimization-based block coordinate gradient coding. In *2021 IEEE Global Communications Conference (GLOBECOM)*, 2021.

[28] Nuwan Ferdinand, Benjamin Gharachorloo, and Stark C. Draper. Anytime exploitation of stragglers in synchronous stochastic gradient descent. In *IEEE Intl. Conf. on Mach. Learning and Appl. (ICMLA)*, pages 141–146, 2017.

[29] S. Lin and D. J. Costello. *Error Control Coding, 2nd Ed.* Prentice Hall, Upper Saddle River, 2004.

[30] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory 2nd Edition (Wiley Series in Telecommunications and Signal Processing).* Wiley-Interscience, 2006.

[31] Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, 2012.

[32] Zizhong Chen and Jack J. Dongarra. Condition numbers of gaussian random matrices. *SIAM Journal on Matrix Analysis and Applications*, 27(3):603–620, 2005.

[33] Roman Vershynin. *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge University Press, 2018.

[34] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[35] Anindya Bijoy Das, Li Tang, and Aditya Ramamoorthy. C^3LES : Codes for coded computation that leverage stragglers. In *IEEE Info. Th. Workshop*, pages 1–5, 2018.

[36] Anindya Bijoy Das and Aditya Ramamoorthy. Coded sparse matrix computation schemes that leverage partial stragglers. *IEEE Trans. on Info. Th.*, 68(6):4156–4181, 2022.

[37] Zachary Charles and Dimitris Papailiopoulos. Gradient coding via the stochastic block model, 2018 [Online] arxiv:1805.10378 [stat.ML].

[38] J. Kleinberg and E. Tardos. *Algorithm Design*. Pearson, 2005.

[39] J. H. Van Lint and R. M. Wilson. *A Course in Combinatorics*. Cambridge University Press, New York, 2001.

[40] Aditya Ramamoorthy, Ruoyu Meng, and Vrinda S. Girimaji. Leveraging partial stragglers within gradient coding - software repository. `https://github.com/flamethrower775/Leveraging-partial-stragglers-within-gradient-coding`, 2024.

[41] F. R. K. Chung. *Spectral Graph Theory*. American Mathematical Society, Rhode Island, 1997.

Appendix A Numerical Instability of Lagrange Interpolation.

Even for values such as $\ell = 2$ and $m = 22, 27, 32$, the error in Lagrange interpolation is too high for the technique of [21] to be useful. To see this consider Fig. 4 which shows the error (averaged over 100 random trials) in first interpolating and then evaluating the interpolated Lagrange polynomial (degrees 20, 25 and 30) at specific points (as is done in [21]); the x-axis is the precision. It can be observed that the error even with full-precision is too high for the technique to be useful.

Appendix B Additional numerical experiments

Fig. 5 shows the mean-squared-error (MSE) comparing the original GC approach with $\ell = 1$ and our partial GC approach for $\ell = 1, 2, 3$ for graph G_2 with $m = 300$ vertices. The results are similar in spirit to the results for the case of G_1 that has 200 vertices. Namely, our MSE is orders of magnitude lower than the original GC approach, even when we consider $\ell \geq 2$.

In Figs. 6a and 6b we study the impact of chunk ordering within the workers for the assignment matrices corresponding to graphs G_1 and G_2 respectively. Each data point on the curves corresponds to 1000 simulations (setup described in Section 4). In particular, in Fig. 6a corresponding to the case of $\ell = 1, 2, 3$ for G_1, we observe that the MSE of the optimal ordering consistently remains lower than the MSE of the random ordering and can in fact be multiple orders of magnitude lower when the

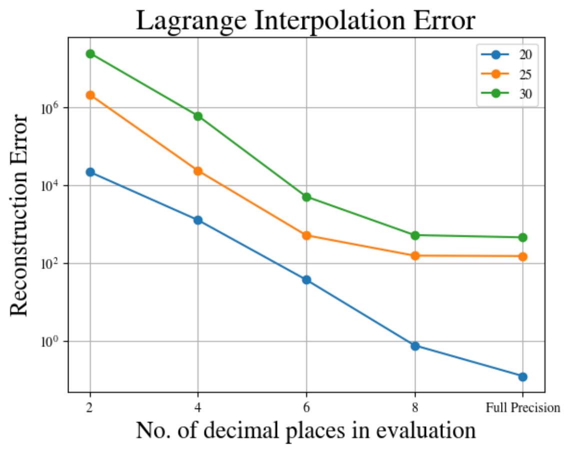


Figure 4: Error in Lagrange interpolation vs. the number of decimal places (precision) in the evaluation values. The three curves correspond to polynomials of degree 20, 25 and 30 (average of 100 trials).

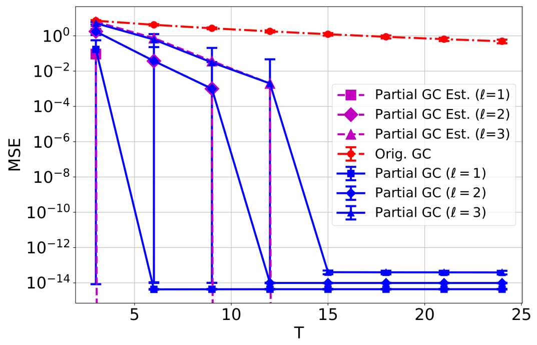
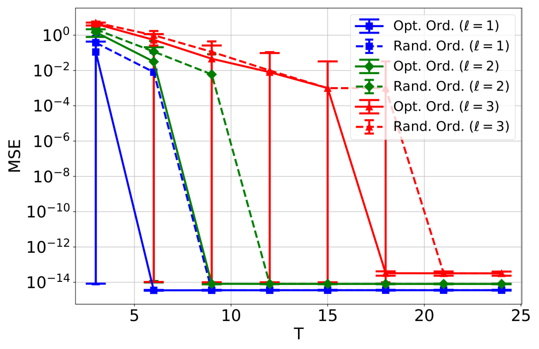
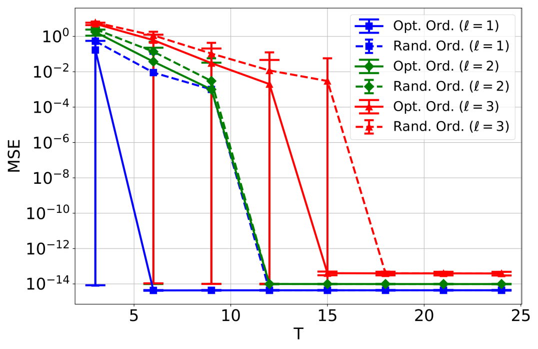


Figure 5: Mean-squared error (MSE) vs. T for an approximate GC scenario corresponding to an assignment matrix stemming from graph G_2. Error bars correspond to one standard deviation. Blue curves: proposed protocol with $\ell = 1, 2, 3$, purple curves: corresponding MSE estimates, and red curve: original GC protocol with $\ell = 1$.



(a)



(b)

Figure 6: Mean-squared error (MSE) vs. T for the approximate GC scenario when considering random chunk ordering and optimal chunk ordering within our protocol with $\ell = 1, 2, 3$. Error bars correspond to one standard deviation. (a) Assignment matrix corresponding to graph G_1. (a) Assignment matrix corresponding to graph G_2.

encode-and-transmit signal is sent at certain time intervals. A similar pattern can be observed in Fig. 6b, which shows the case of $\ell = 1, 2, 3$ and G_2.

Fig. 7 shows the results of a similar experiment comparing the random chunk ordering and our optimal chunk ordering in terms of completion time for exact GC. The assignment matrix corresponds to the graph G_1 discussed in Section 4. The random ordering is chosen by sampling 100 independent random orderings and selecting one with the smallest $Q_{max}(O)$. A data point is then generated by running 1000 simulations with the selected random ordering. The results indicate that in an average sense, the completion time of the optimal ordering is clearly lower.

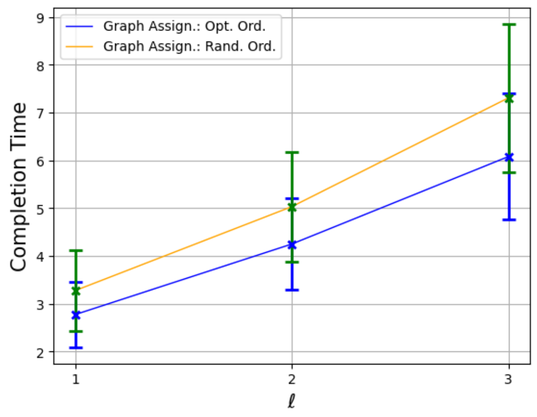


Figure 7: Completion time vs. ℓ for the exact GC scenario with random chunk ordering and optimal chunk ordering. Error bars correspond to one standard deviation.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We present a new gradient coding algorithm. We have discussed the assumptions underlying the algorithm and outlined the main contributions of our work. The contributions are substantiated within the body of the paper.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We have a section titled Limitations in the paper that discusses these issues. Computational complexity has been discussed for all our algorithms.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: The paper presents protocols and algorithms that are relevant to gradient coding. Within the main body of the paper, we have included analyses of these protocols and algorithms. The analyses include proof of correctness, performance analysis and computational complexity analysis.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We have provided complete descriptions of all our protocols and algorithms. In addition, we also have an example of the main protocol that guides the reader through the main steps.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in

some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We have uploaded commented code that recreates the main experimental results.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Our protocols and algorithms do not have any hyperparameters that need to be tuned. All experimental settings and details are available in the body of the paper itself.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We have included error bars for the relevant plots.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: This information is provided on the section on Numerical Experiments.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Yes. We have read the Code of Ethics and our submission conforms to it.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: There are no negative social impacts of our work. Our work allows for faster distributed training on cloud platforms. The positive societal impact is that it will save computational cycles on such platforms.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.

- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: We do not have such risks because the paper is about distributed training.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [NA]

    Justification: We do not use any existing assets.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: We are not releasing any new assets in this work.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: Our work is not related to crowdsourcing nor research with human subject

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: Our work does not involve human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.