Approximate Gradient Coding Using Convex Optimization

Sifat Munim and Aditya Ramamoorthy

***Abstract*—Gradient coding is a technique that helps mitigate the effect of slow or failed workers within large-scale distributed parameter learning. The basic idea is to introduce redundancy within the assignment of data points to the workers and use coding-theoretic ideas to allow the parameter server (PS) to recover an exact or approximate gradient even in the presence of failures. In this work, we present approximate gradient coding schemes by applying convex optimization methods to structured matrices such as circulants and transition matrices. For low complexity ($O(n)$) decoding by the PS, our techniques perform significantly better.**

***Index Terms*—Approximate gradient coding, second largest eigenvalue modulus (SLEM), convex optimization.**

I. Introduction

Large scale distributed learning is used routinely in most modern machine learning problems. In these problems, the huge amounts of data and corresponding computation needs, necessitate the usage of large clusters. In a typical parameter fitting scenario, there is a dataset and a corresponding loss function. The goal of the system is to minimize the loss function with respect to the parameter. For performing this in a distributed fashion, the original dataset is subdivided into smaller parts and different workers focus on computing the gradient on the part(s) assigned to them. A designated parameter server (PS) coordinates the training by aggregating the received gradients from the workers and generating a new parameter, and the process continues thereafter.

However, in many instances, workers within these clusters are often slower than their promised speed or even prone to failure. Such issues are often exaggerated in cloud based clusters, where the worker usage fluctuates based on the system load. To address these issues, the idea of gradient coding was first introduced in the work of [1]. The basic idea is to introduce redundancy within the assignment of data points to the workers. Once a given worker calculates the gradient on its assigned parts, 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 the full gradient even in the presence of node failures. For exact gradient recovery from *any* s failures, it can be shown that each part needs to be replicated at least $s+1$ times across the cluster.

There are scenarios in which the full gradient is not required or too costly to work with because of the high replication factor needed, e.g., many ML algorithms work with mini-batch stochastic gradient descent (SGD) where the gradients are only computed on a subset of the data points. Approximate gradient coding aims at designing schemes with guarantees on the gradient quality even in the presence of a large number of failures (greater than the replication factor of the parts). It was introduced in [2], which showed connections of this problem with the spectral properties of graphs and constructed schemes from expander graphs. The work of [3] (see also [4]) observed that Ramanujan graphs (expanders with the largest spectral gap) only exist in restrictive settings and considered the usage of sparse random graphs instead. The work of [5] also presented graph based schemes in this setting. Connections of approximate GC with block designs were considered in [6] and subsequently in [7].

In this work, we use convex optimization methods [8] along with structured matrices such as circulants and transition matrices to design approximate gradient coding schemes.

The authors are with the Department of Electrical and Computer Engineering, Iowa State University, Ames, IA 50010, U.S.A.

II. Background, Related Work and Contributions

A typical problem instance within distributed learning consists of a dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$ of m points (where the $(\mathbf{x}_i, y_i)$'s are feature & label pairs), and a loss function $L = \frac{1}{m}\sum_{i=1}^m l(\mathbf{x}_i, y_i, \mathbf{w})$ that needs to be minimized with respect to $\mathbf{w} \in \mathbb{R}^d$. Here, l is a function that measures the prediction error for each point. When the size of $\mathcal{D}$ is large, we can perform the task in a distributed manner.

The system consists of the PS and n worker nodes denoted $W_1, W_2, \ldots, W_n$. In each iteration, the dataset is partitioned into k disjoint subsets of equal size, $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_k$. Let $\mathbf{B} \in \mathbb{R}^{k\times n}$ be a matrix such that $\mathbf{B}_{i,j} \neq 0$ if and only if subset $\mathcal{D}_i$ is assigned to worker W_j. We call this the encoding matrix; it also specifies the assignment of subsets to workers. Let $nnz(v)$ denote the number of non-zero entries in a vector v. Let γ_i and δ_j denote $nnz(\mathbf{B}(i,:)$ and $nnz(\mathbf{B}(:,j))$ (using MATLAB notation). These correspond respectively to the number of times $\mathcal{D}_i$ appears in the cluster (replication factor) and the number of subsets assigned to W_j (computation load). In what follows, we will assume that $\gamma_i = \gamma$ for $i \in [k]$ and $\delta_j = \delta$ for $j \in [n]$.

Let $\mathbf{g}_i \triangleq \frac{k}{m}\sum_{(\mathbf{x}_j, y_j)\in D_i} \nabla l(\mathbf{x}_j, y_j, \mathbf{w})$. Define a matrix $\mathbf{G} \in \mathbb{R}^{d\times k}$ such that $\mathbf{G} = \frac{1}{k}[\mathbf{g}_1\ \mathbf{g}_2 \ldots \mathbf{g}_k]$. Worker W_j calculates $\mathbf{g}_i$ for $i \in \text{supp}(\mathbf{B}(:,j))$ (non-zero entries in the j^{th} column of $\mathbf{B}$) and sends the following linear combination to the PS: $\mathbf{v}_j = \frac{1}{k}\sum_{i=1}^k \mathbf{g}_i\mathbf{B}_{i,j} = \mathbf{G}\mathbf{b}_j$ ($\mathbf{b}_j$ denotes the j-th column of $\mathbf{B}$). The goal of the PS is to compute $\nabla L = \frac{1}{m}\sum_{i=1}^m \nabla l(\mathbf{x}_i, y_i, \mathbf{w}) = \frac{1}{k}\sum_{i=1}^k \mathbf{g}_i$ either exactly or approximately and iteratively update the parameter as $\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \frac{\mu_t}{m}\sum_{i=1}^m \nabla l(\mathbf{x}_i, y_i, \mathbf{w}^{(t)})$. Here, $\mu_t > 0$ is

the learning rate and $\mathbf{w}^{(t)}$ is the state of the parameter $\mathbf{w}$ at iteration t.

While decoding, the PS computes a linear combination of the encoded gradients that it receives from the non-straggling workers. Let s be the number of stragglers and $\mathcal{F} \subseteq [n]$ be a set of indices that correspond to the non-straggling workers. It computes $\mathbf{g}' = \mathbf{GBr}$, where $\mathbf{r} \in \mathbb{R}^n$ is a decoding vector such that $\mathbf{r}_i = 0$ if $i \in \mathcal{F}^c$.

Let $d_2(\mathbf{u}, \mathbf{v}) = ||\mathbf{u} - \mathbf{v}||_2$ (ℓ_2-norm) and $\mathbf{1}$ denote the all-ones vector. We have,

$$d_2(\mathbf{g}', \mathbf{G1}) = \|\mathbf{G}(\mathbf{Br} - \mathbf{1})\|_2 \leq \|\mathbf{G}\|_2 \|\mathbf{Br} - \mathbf{1}\|_2,$$

where $\|\mathbf{G}\|_2$ denotes the spectral norm of $\mathbf{G}$. Thus, if $\mathbf{Br} = \mathbf{1}$, then we recover the exact gradient, while if $||\mathbf{Br} - \mathbf{1}||_2 > 0$ then the gradient recovery is approximate.

The formulation of the gradient coding problem is predicated on letting the PS act essentially as a coordinator with low computational overhead. Thus, ideally, the decoding process should be a light-weight procedure. While the decoding vector $\mathbf{r}$ depends on $\mathcal{F}$, within "fixed decoding" we set $\mathbf{r}_i, i \in \mathcal{F}$ in a simple manner, typically such that all non-zero entries of $\mathbf{r}$ are the same; the decoding complexity is $O(n)$. On the other hand, within "optimal decoding", we choose $\mathbf{r}$ by solving a least-squares problem $\min_{\mathbf{r}} ||\mathbf{Br} - 1||_2$ subject to the constraint $\mathbf{r}_i = 0$ for $i \in \mathcal{F}^c$. The time-complexity of this is $O(n^3)$.

Definition 1*:* For a given encoding matrix $\mathbf{B}$ and a given set of non-stragglers corresponding to $\mathcal{F} \subseteq [n]$ of size $n - s$, the error $\text{Err}_{\mathcal{F}}(\mathbf{B})$ is defined as

$$\text{Err}_{\mathcal{F}}(\mathbf{B}) \triangleq \min_{\substack{\mathbf{r} \in \mathbb{R}^n \\ \text{supp}(\mathbf{r}) \subseteq \mathcal{F}}} \|\mathbf{Br} - \mathbf{1}\|_2 \tag{1}$$

The main goal within gradient coding is to design $\mathbf{B}$ such that $\text{Err}_{\mathcal{F}}(\mathbf{B})$ can be upper bounded over all $\mathcal{F}$ of size at least $(n - s)$ using either the fixed or optimal decoding techniques discussed above.

Owing to the high cost of optimal decoding in the PS, there are scenarios where the PS wants to operate only with fixed decoding. Thus, fixed decoding is the scenario we consider in this work.

Related Work: Prior work within approximate gradient coding demonstrates constructions (with $k = n$) from expander graphs [2] such as Ramanujan and Margulis graphs, sparse random graphs [3] and [5], block designs [6], [7] and the like. For fixed decoding with graph based schemes, [2] demonstrates an upper bound on $\text{Err}_{\mathcal{F}}(\mathbf{B})$ that depends on the second-largest eigenvalue modulus (SLEM) of $\mathbf{B}$. In particular, let $\mathbf{C} \in \mathbb{R}^{n \times n}$ be a matrix with eigenvalues $\lambda_i, i = 1, \ldots, n$ that satisfies the following properties:

- There is an orthogonal set of n eigenvectors of $\mathbf{C}$.
- $\mathbf{1}$ is an eigenvector of $\mathbf{C}$ and the corresponding eigenvalue λ_1 is real, and it satisfies the following: $\lambda_1 \geq \max_{j=2,\ldots,n} |\lambda_j|$.

Let us denote the SLEM of $\mathbf{C}$ by $\lambda \triangleq \max(|\lambda_2|, \ldots, |\lambda_n|)$. For a given $\mathcal{F} \subseteq [n]$ such that $|\mathcal{F}| = n - s$, define $\mathbf{u}_{\mathcal{F}} \in \mathbb{R}^n$ as

$$(\mathbf{u}_{\mathcal{F}})_i = \begin{cases} -1 & i \notin \mathcal{F} \\ \frac{s}{n-s} & i \in \mathcal{F} \end{cases} \tag{2}$$

Now, Let $\mathbf{B} = \frac{1}{\lambda_1}\mathbf{C}$ and $\mathbf{r}' = \mathbf{u}_{\mathcal{F}} + \mathbf{1}$ so that $\text{supp}(\mathbf{r}') = \mathcal{F}$.

Proposition 1*:* [2] For every nonempty set $\mathcal{F} \subseteq [n]$ of size $n - s$ such that $\mathbf{r}' = \mathbf{u}_{\mathcal{F}} + \mathbf{1}$, we have $Err_{\mathcal{F}}(\mathbf{B}) \leq \|\mathbf{Br}' - \mathbf{1}\|_2 \leq \frac{\lambda}{\lambda_1}\sqrt{\frac{ns}{n-s}}$.

We refer to $\frac{\lambda}{\lambda_1}$ as the improvement factor of the scheme. Evidently, $\mathbf{B}$ matrices with smaller λ/λ_1 have better performance under approximate gradient coding. However, constructing such matrices is not so straightforward since for a fixed computation load δ, each column of $\mathbf{B}$ needs to be sparse as it can have (at most) δ non-zero entries.

Main contributions: In this work, we propose a new approximate gradient coding scheme that uses fixed decoding. Our technique uses convex optimization to design circulant matrices [9] and probability transition matrices [10] (corresponding to Markov Chains on graphs) in order to minimize the SLEM λ of the corresponding matrices, and hence the upper bound of the fixed decoding error. These matrices are easy to construct and hence our scheme is feasible for any n and δ, in contrast to Ramanujan/Margulis graphs and block design based techniques that have restricted parameter sets. Our numerical experiments demonstrate that our schemes have significantly better performance under fixed decoding.

III. Optimizing SLEM of circulant matrices

Definition 2*:* A circulant matrix $\mathbf{C} \in \mathbb{R}^{n \times n}$ is a square matrix where all the rows contain the same entries and each row is shifted one entry to the right than the previous row. It has the following form.

$$\mathbf{C} = \begin{bmatrix} c_0 & c_{n-1} & \cdots & c_2 & c_1 \\ c_1 & c_0 & c_{n-1} & \cdots & c_2 \\ \vdots & c_1 & c_0 & \cdots & \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ c_{n-1} & \cdots & \cdots & c_1 & c_0 \end{bmatrix}.$$

The matrix is specified by one vector $\mathbf{c} = (c_0, c_{n-1}, \ldots, c_1)^T$ where $\mathbf{c}^T$ appears as the first row of $\mathbf{C}$.

Proposition 2*:* [9] The eigenvectors of a circulant matrix $\mathbf{C}$ are orthogonal and have the following form.

$$\mathbf{v}_j = \left(1, \omega_n^{j-1}, \omega_n^{2(j-1)} \ldots, \omega_n^{(n-1)(j-1)}\right)^T$$

for $j = 1, 2, \ldots, n$. Here, $\mathrm{i} = \sqrt{-1}$ and, $\omega_n = e^{\frac{2\pi \mathrm{i}}{n}}$ is the n^{th} root of unity. The corresponding eigenvalues are $\lambda_j = c_0 + c_{n-1}\omega_n^{j-1} + \cdots + c_1\omega_n^{(n-1)(j-1)} = \mathbf{c}^T\mathbf{v_j}$.

Suppose we choose a certain subset of entries in $\mathbf{c}$ to be strictly positive; the others are set to zero. This means that $\mathbf{C}$ is a non-negative matrix (all non-negative entries). It follows that the all-ones vector $\mathbf{1}$ is an eigenvector of $\mathbf{C}$ with eigenvalue $\mathbf{c}^T\mathbf{1}$. Then it follows (see Theorem 8.3.4 in [11]) that $\lambda_1 = \mathbf{c}^T\mathbf{1} = \rho(C)$ (spectral radius of $\mathbf{C}$).

Our main idea in this work is to choose $\mathbf{c}$ with exactly δ strictly positive entries, construct the circulant matrix $\mathbf{C}$ and set the encoding matrix $\mathbf{B}$ as $\frac{1}{\lambda_1}\mathbf{C}$. In our work, we choose the location of the δ non-zero entries of $\mathbf{c}$ at random and choose

their actual values using the following convex optimization procedure, which can be solved efficiently.

It follows from the previous discussion that the SLEM of $\mathbf{C}$ is $\lambda = \max_{j=2,\ldots,n} |\lambda_j| = \max_{j=2,\ldots,n} |\mathbf{c}^T \mathbf{v_j}|$. Thus, we can formulate an optimization problem with decision variable $\mathbf{c}$ as follows (the non-zero locations of $\mathbf{c}$ are in the subset $\mathcal{C}$).

$$\begin{aligned} \text{Minimize} \quad & t \\ \text{subject to} \quad & \mathbf{c}^T \mathbf{1} = 1, \\ & |\mathbf{c}^T \mathbf{v_j}| \leq t, \quad j = 2, \ldots, n, \\ & c_i = 0, \quad \forall c_i \notin \mathcal{C}, \\ & c_i \geq 0, \quad \forall i \in \{0, \ldots, n-1\}. \end{aligned} \tag{3}$$

We note here that (3) can be replaced by a constraint involving only real matrices, as follows.

Consider positive-definite and Hermitian, $\mathbf{V}_j = \mathbf{v}_j \mathbf{v}_j^\dagger$ (superscript † denotes conjugate-transpose) for $j = 2, \ldots, n$. Let $\mathbf{V}_j = \mathbf{V}_j^r + \mathrm{i}\mathbf{V}_j^c$, where $\mathbf{V}_j^r$ and $\mathbf{V}_j^c$ are matrices with real entries. Since $\mathbf{V}_j$ is hermitian, $\mathbf{V}_j^r$ is a symmetric matrix and $\mathbf{V}_j^c$ is a skew-symmetric matrix. So, $\mathbf{V}_j^r = (\mathbf{V}_j^r)^T$ and $\mathbf{V}_j^c = -(\mathbf{V}_j^c)^T$. For any $\mathbf{z} \in \mathbb{R}^n$, we have

$$\mathbf{z}^T \mathbf{V}_j^c \mathbf{z} = (\mathbf{z}^T \mathbf{V}_j^c \mathbf{z})^T = \mathbf{z}^T (\mathbf{V}_j^c)^T \mathbf{z} = -\mathbf{z}^T \mathbf{V}_j^c \mathbf{z},$$

so that $\mathbf{z}^T \mathbf{V}_j^c \mathbf{z} = 0$. Thus, $\mathbf{z}^T \mathbf{V}_j \mathbf{z} = \mathbf{z}^T \mathbf{V}_j^r \mathbf{z} + i\mathbf{z}^T \mathbf{V}_j^c \mathbf{z} = \mathbf{z}^T \mathbf{V}_j^r \mathbf{z}$ for any $\mathbf{z} \in \mathbb{R}^n$. So, $\mathbf{V}_j^r$ is also positive semi-definite since for any $\mathbf{z} \in \mathbb{R}^n$ such that $\mathbf{z} \neq 0$, $\mathbf{z}^T \mathbf{V}_j^r \mathbf{z} = \mathbf{z}^T \mathbf{V}_j \mathbf{z} \geq 0$. Consequently, $|\mathbf{c}^T \mathbf{v}_j|^2 = \mathbf{c}^T \mathbf{V}_j \mathbf{c} = \mathbf{c}^T \mathbf{V}_j^r \mathbf{c}$ is a convex function. So we can replace the inequality constraint $|\mathbf{c}^T \mathbf{v_j}| \leq t$ in (3) by $\mathbf{c}^T \mathbf{V}_j^r \mathbf{c} \leq t$. Since the inequality constraint functions are convex and equality constraint functions are affine, the optimization problem (3) is convex [8].

IV. Optimizing SLEM of Transition Matrices

Our second class of approximate gradient coding schemes are derived by optimizing the transition matrices of discrete-time Markov chains defined on graphs.

Definition 3*:* Consider a discrete-time Markov chain with n states. The corresponding transition matrix $\mathbf{P} \in \mathbb{R}^{n \times n}$ is a matrix whose entry $\mathbf{P}_{i,j}$ represents the transition probability of moving to state j given that the current state is i, i.e., for $i, j \in [n]$,

$$\mathbf{P}_{i,j} = \mathbf{Prob}(j|i) \geq 0. \tag{4}$$

Since the sum of the transition probabilities from a state i to all other states must be 1, we have $\mathbf{P1} = \mathbf{1}$.

If $\mathbf{P}$ is symmetric, then it has real eigenvalues and the eigenvectors are orthonormal. Let $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$ be the eigenvalues of $\mathbf{P}$. Moreover, since a transition matrix is non-negative and $\mathbf{1}$ is an eigenvector, by invoking Theorem 8.3.4 of [11] we get $1 = \lambda_1 \geq \max_{j=2,\ldots,n} |\lambda_j|$. So we can use encoding matrix $\mathbf{B}$ such that $\mathbf{B} = \frac{1}{\lambda_1}\mathbf{P}$.

To account for the computation load δ, we associate a δ-regular graph with a symmetric transition matrix $\mathbf{P}$. Let $G = (\mathcal{V}, \mathcal{E})$ be a δ-regular graph, $\mathcal{V} = \{1, \ldots, n\}$. We let $\mathbf{P}_{ij} = 0$ if and only if there is no edge between vertices i and j in graph G. As a consequence, $|\mathrm{supp}(\mathbf{p}_i)| = \delta$. Let $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n$ be the singular values of $\mathbf{P}$. Since $\mathbf{P}$ is symmetric, we have $\sigma_i = |\lambda_i|$ for all $i \in [n]$. The SLEM of transition matrix $\mathbf{P}$ is $\lambda = \max_{j=2,\ldots,n} |\lambda_j| = \max_{j=2,\ldots,n} \sigma_j = \|\mathbf{P} - \frac{1}{n}\mathbf{1}\mathbf{1}^T\|_2$. So, in this case, the objective function is $\|\mathbf{P} - \frac{1}{n}\mathbf{1}\mathbf{1}^T\|_2$. Finally, we have the following optimization problem [12].

$$\begin{aligned} \text{Minimize} \quad & \|\mathbf{P} - \frac{1}{n}\mathbf{1}\mathbf{1}^T\|_2 \\ \text{subject to} \quad & \mathbf{P} = \mathbf{P}^T, \\ & \mathbf{P1} = \mathbf{1}, \\ & \mathbf{P}_{i,j} = 0, \quad \text{if} \quad (i,j) \notin \mathcal{E} \\ & \mathbf{P}_{i,j} \geq 0, \quad \text{for all} \quad i, j \in [n] \end{aligned} \tag{5}$$

Here, the objective function is a norm and as a consequence, a convex function. If the underlying graph G is predetermined, then the equality constraint $\mathbf{P}_{ij} = 0$, if $ij \notin \mathcal{E}$ is an affine function. Since all the inequality constraint functions are convex and all the equality constraint functions are affine, the optimization problem (5) is convex [8].

V. Numerical Experiments

We now present the results of our numerical experiments that compare our results with prior approaches. The first set of experiments compares the improvement factor λ/λ_1 of our optimization-based approaches with prior approaches. Prior approaches based on Ramanujan and Margulis graph constructions typically only exist for large and restricted values of n. The second set of experiments considers fixed decoding and compares the fixed decoding error of certain *bad* straggler sets for the prior schemes with the improvement factor based upper bound of our schemes. For these experiments, we considered random regular graphs up to $n = 200$ as these correspond to the more practical regimes for the distributed learning setup.

The experiments were performed using CVXPY [13] and Gurobi Optimizer [14]. All software for recreating these results can be found at [15].

A. Improvement Factor Comparisons

Behavior with computation load: We computed the improvement factors for different values of computation load δ with $n = 200$ for random regular graphs and optimized circulant and transition matrices. In case of circulant matrices, δ non-zero entries of $\mathbf{c}^T$ were chosen randomly; 1500 random trials were conducted. The lowest improvement factor is shown in the plot in Fig. 1. In case of transition matrices, for each value of computation load δ, a random regular graph was generated, and it was used as the underlying graph of the optimization problem. From Fig. 1, it can be observed that the gap between the improvement factors of our schemes and random regular graphs increases as δ increases.

Comparisons with state-of-the-art instances: For Ramanujan graphs, the SLEM is $\lambda \leq 2\sqrt{\delta - 1}$, which is a tight bound [16]. For Margulis graphs, $\lambda \leq 5\sqrt{2}$. These graphs constitute instances of graphs with the best improvement factor.

We compared the improvement factors obtained by our schemes with these graphs for the same values of (n, δ). In

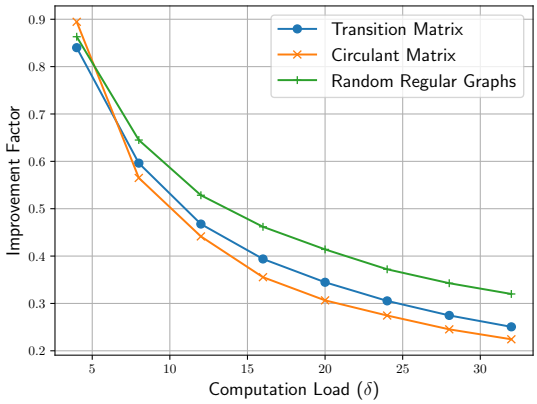


Fig. 1: Improvement factor vs computation load for $n = 200$.

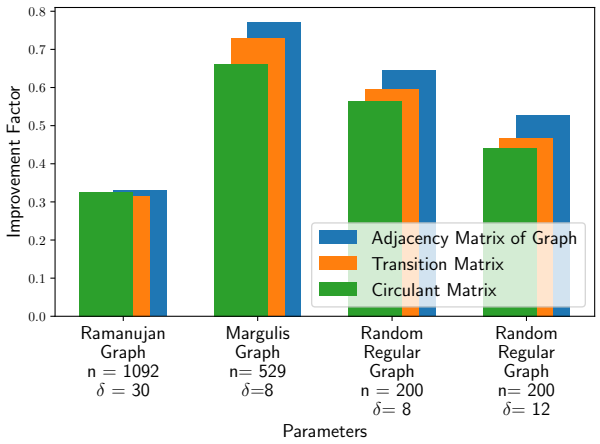


Fig. 2: Comparison of improvement factors for different constructions.

particular, we chose a Ramanujan graph ($(n, \delta) = (1092, 30)$), a Margulis graph ($(n, \delta) = (529, 8)$) and two random regular graphs ($(n, \delta) = (200, 8)$ and $(200, 16)$). For circulant matrices, 1000 optimization problems (randomly chosen non-zero entries) were run for each case and the best improvement factor was reported. For transition matrices, the optimization problems were run with the mentioned graphs as underlying graphs. As shown in Fig. 2 the improvement factors of circulant matrix and transition matrix are clearly better.

B. Fixed Decoding Error Comparisons

Our next set of comparisons are about the actual error of the different schemes under fixed decoding. For each value of s, for the random regular graph based scheme, we first identified a bad straggler set that causes the fixed decoding error to be relatively large. This is done by integer linear programming. The details can be found in the Appendix. We note here that finding the worst-case straggler set is in general a hard problem. Each such bad straggler set corresponds to how high the error can be for the graph-based scheme under fixed decoding. We compared this error with the corresponding upper bound on the fixed decoding error of our schemes, that leverages the improvement factor. We emphasize that the upper bound on the fixed decoding error of our schemes holds regardless of what straggler set is considered.

We chose regular graphs ($n = 100, 200$ and $\delta = 8, 16$) for this comparison, i.e., there were a total of four graphs. For $(n, \delta) = (100, 8)$, the cross-over points between the curves occur at about a straggler fraction of 0.05 and 0.29 for the circulant scheme and transition matrix scheme respectively. When $(n, \delta) = (100, 16)$, the cross-over points shift left to straggler fractions around 0.03 and 0.14 respectively.

For $(n, \delta) = (200, 8)$ the corresponding cross-over point shifts to straggler fraction around 0.24 for the circulant scheme; there is no cross-over for the transition matrix scheme until a straggler fraction of 0.4. When $(n, \delta) = (200, 16)$, these cross-over points shift left-ward to straggler fractions about 0.07 (circulant scheme) and 0.26 (transition matrix scheme).

We conclude that our optimization based schemes tend to have larger improvement in performance when the computation load is higher. This is expected, as a higher computation load, allows for a larger feasible optimization space.

VI. Conclusion

In this work, we proposed a convex optimization based schemes for approximate gradient coding under fixed decoding. Our work is applicable for any values of parameters n and δ, whereas techniques that are based on objects such as expander graphs and block designs are applicable for a specific set of parameters. Our numerical experiments demonstrate that our approach has lower fixed decoding error than graph-based schemes for a large range of straggler fractions.

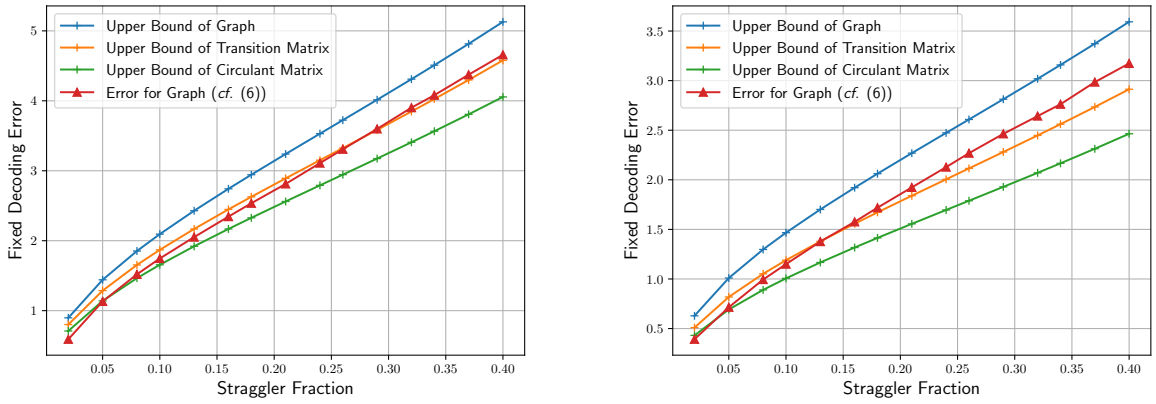
Appendix

Finding the worst case straggler set:
Let $\mathbf{r}'$ denote the decoding vector used under fixed decoding when the number of stragglers is s (*cf.* discussion after (2)). Thus, $\mathbf{r}'$ is such that all non-zero component values are the same. Let $\mathbf{B}$ be the adjacency matrix of a δ-regular graph, normalized by δ. We have,

$$\begin{aligned}\|\mathbf{B}\mathbf{r}' - \mathbf{1}\|_2^2 &= (\mathbf{B}\mathbf{r}' - \mathbf{1})^T(\mathbf{B}\mathbf{r}' - \mathbf{1}) \\ &= \mathbf{r}'^T\mathbf{B}^T\mathbf{B}\mathbf{r}' - 2\delta\mathbf{1}^T\mathbf{r}' + \mathbf{1}^T\mathbf{1} \\ &= \mathbf{r}'^T\mathbf{B}^T\mathbf{B}\mathbf{r}' - 2\delta n + n.\end{aligned}$$

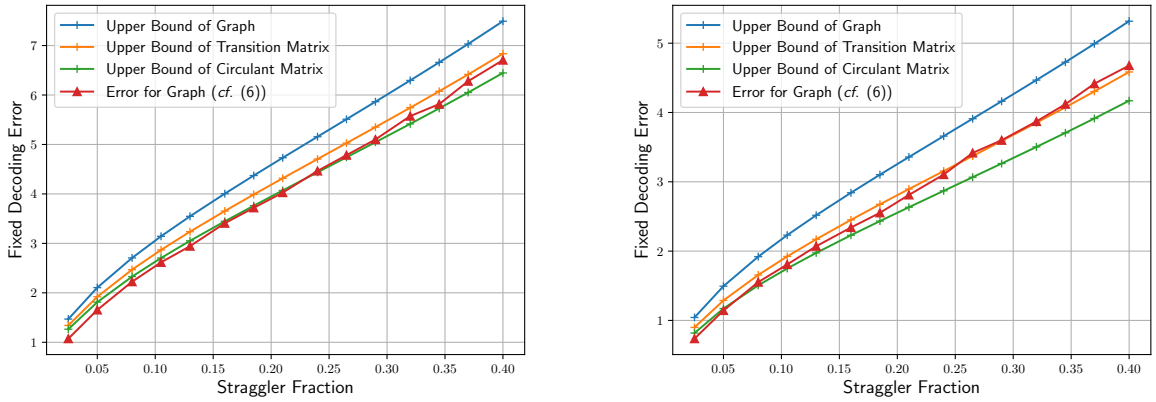
The second equality follows as the column-sums in $\mathbf{B}$ are δ and the third equality follows since $\mathbf{1}^T\mathbf{r}' = n$. Since the last two terms in the third equality are constants and all non-zero entries in $\mathbf{r}'$ take the same value, we can formulate the problem of maximizing the actual fixed decoding error for a given number of stragglers as the following integer linear programming problem.

$$\begin{aligned}\text{Minimize} \quad & -\mathbf{x}^T\mathbf{B}^T\mathbf{B}\mathbf{x} \\ \text{subject to} \quad & \mathbf{1}^T\mathbf{x} = n - s, \\ & \mathbf{x}_i \in \{0, 1\}, i = 1, \ldots, n\end{aligned} \tag{6}$$



(a) $n = 100$, $\delta = 8$.

(b) $n = 100$, $\delta = 16$.



(c) $n = 200$, $\delta = 8$.

(d) $n = 200$, $\delta = 16$.

Fig. 3: (a)-(d) Fixed decoding error for regular graph from solving (6), along with upper bounds of graph, circulant and transition matrix. The time limit to solve (6) was set to 300 seconds.

The equality constraint above ensures $\mathbf{x}$ has $n - s$ nonzero entries (as there are $n - s$ non-stragglers). In our experiments, we solved this problem using Gurobi Optimizer [14] under a limited time constraint.

REFERENCES

[1] R. Tandon, Q. Lei, A. G. Dimakis, and N. Karampatziakis, "Gradient coding: Avoiding stragglers in distributed learning," in *Intl. Conf. Mach. Learn. (ICML)*, August 2017, pp. 3368–3376.

[2] N. Raviv, R. Tandon, A. Dimakis, and I. Tamo, "Gradient coding from cyclic MDS codes and expander graphs," in *Intl. Conf. Mach. Learn. (ICML)*, July 2018, pp. 4302–4310.

[3] Z. Charles, D. Papailiopoulos, and J. Ellenberg, "Approximate gradient coding via sparse random graphs," 2017 [Online] arxiv:1711.06771.

[4] Z. Charles and D. Papailiopoulos, "Gradient coding via the stochastic block model," 2018 [Online] arxiv:1805.10378.

[5] M. Glasgow and M. Wootters, "Approximate gradient coding with optimal decoding," *IEEE J. Select. Areas Info. Th.*, vol. 2, no. 3, pp. 855–866, 2021.

[6] S. Kadhe, O. O. Koyluoglu, and K. Ramchandran, "Gradient coding based on block designs for mitigating adversarial stragglers," in *IEEE Intl. Symp. on Info. Th.*, 2019, pp. 2813–2817.

[7] A. Sakorikar and L. Wang, "Soft BIBD and Product Gradient Codes," *IEEE J. Select. Areas Info. Th.*, vol. 3, no. 2, pp. 229–240, 2022.

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

[9] R. M. Gray, "Toeplitz and circulant matrices: A review," *Foundations and Trends® in Communications and Information Theory*, vol. 2, no. 3, pp. 155–239, 2006.

[10] G. Grimmett and D. Stirzaker, *Probability and Random Processes*. Oxford University Press, 2020.

[11] R. A. Horn and C. R. Johnson, *Matrix analysis*. Cambridge University Press, 2012.

[12] S. Boyd, P. Diaconis, and L. Xiao, "Fastest mixing markov chain on a graph," *SIAM Review*, vol. 46, no. 4, pp. 667–689, 2004.

[13] S. Diamond and S. Boyd, "CVXPY: A Python-embedded modeling language for convex optimization," pp. 1–5, 2016.

[14] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2023. [Online]. Available: https://www.gurobi.com

[15] "Repository of Approximate Gradient Coding Using Convex Optimization." [Online]. Available: https://github.com/smunim-47/CVX_AGC

[16] S. Hoory, N. Linial, and A. Wigderson, "Expander graphs and their applications," *Bull. Amer. Math. Soc.*, vol. 43, no. 04, p. 439–562, Aug. 2006.