# On Arbitrary Ignorance of Stragglers with Gradient Coding

Xian Su*, Brian Sukhnandan†, Jun Li‡
*Graduate Center, City University of New York
†Arch
‡Queens College and Graduate Center, City University of New York

***Abstract*—Gradient methods, such as gradient descent, are widely deployed to train optimization-based models in machine learning. To train such models on a large dataset, the dataset is commonly split into multiple partitions which are trained on different workers. In order to tolerate stragglers, existing techniques either use gradient coding (GC) to recover the full gradients from a certain number of workers or recover gradients partially from an arbitrary number of workers. In this paper, we propose ignore-straggler gradient coding (IS-GC) that allows GC to tolerate an arbitrary number of stragglers. Compared to approximated gradient descent, IS-GC can recover more gradients when there is the same number of stragglers. We design a graph-based model to decode coded gradients from an arbitrary number of stragglers, and prove that it can maximize the recovery of gradients. We apply IS-GC on fractional repetition (FR) and cyclic repetition (CR), two representative dataset placement schemes of GC. We also propose hybrid repetition (HR) that generalizes over FR and CR and achieves a flexible trade-off between FR and CR. With extensive experiments, we demonstrate that IS-GC can flexibly tolerate an arbitrary number of stragglers and achieve a low completion time of training.**

## I. Introduction

Gradient methods, *e.g.*, stochastic gradient descent (SGD), include a large family of algorithms that solve optimization problems by iteratively updating the parameters from gradients and are widely used in training machine-learning models. Running gradient methods on a single server, however, can often be prohibitively slow with a large dataset, as the limited capacity of one server could become the bottleneck. Therefore, it has been common to train optimization-based models in a distributed infrastructure, where gradients on different partitions of the dataset are evaluated on different servers in parallel. Specifically, in distributed gradient descent, a dataset $D$ can be equally split into $n$ partitions, *i.e.*, $D_1, \ldots, D_n$. Then the full gradients $g$ can be computed as the sum of gradients evaluated on each partition, *i.e.*, $\sum_{i=1}^{n} g_i$. In practice, the $n$ partitions are stored on $n$ servers called *workers*, which also compute the gradients on corresponding dataset partitions. Another server called *master* receives $g_i$ from each worker and then computes $g$ to update parameters. Fig. 1(a) illustrates an example of distributed gradient descent where the dataset is split into 4 partitions, and the master expects to obtain $\sum_{i=1}^{4} g_i$ to proceed to the next step. This process will repeat until parameters converge. When workers compute the gradients from mini-batches, the algorithm is known as synchronous SGD.

This work is partially supported by the Google Cloud Research Credits program.

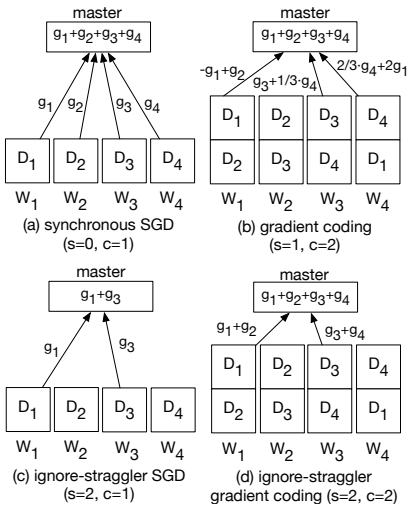



Fig. 1: Examples of techniques for tolerating stragglers in distributed SGD, with $n = 4$ workers.

However, it is well known that workers may become stragglers in a distributed infrastructure, resulting in a significant delay in the computation and/or communication of gradients [1], [2]. In order to get $g$, the master needs to wait for the gradients from all the $n$ workers. Therefore, even one straggler may significantly delay the progress of the training.

To tolerate $s$ stragglers in synchronous SGD, gradient coding (GC) was proposed by encoding gradients of $c$ dataset partitions on each worker [1]. As shown in Fig. 1(b), while we still split $D$ into 4 partitions, each of them is stored on two of the four workers. While each worker can now compute the gradients on two partitions, *i.e.*, $c = 2$, worker $W_i$ will not send the gradients on two partitions but their linear combinations. For instance, the worker $W_1$ sends $-g_1+g_2$ to the master. Thus, the master does not need to wait for the coded gradients from all the 4 workers but only 3 workers, making it possible to tolerate one straggler among the 4 workers, *i.e.*, $s = 1$. In this example, $(-g_1 + g_2) + (g_3 + \frac{1}{3}g_4) + (\frac{2}{3}g_4 + 2g_1) = g_1 + \cdots + g_4 = g$.

GC can tolerate at most $c - 1$ stragglers with the gradients fully recovered [1]. However, it also leads to restrictions in two ways. First, when there are more stragglers, the master cannot recover any gradients. Second, if the number of stragglers is less than $c - 1$, the additional coded gradients become useless

toward the recovery of gradients. Such two restrictions are due to the synchronous nature of GC, which can be easily resolved by asynchronous SGD.

Asynchronous SGD allows the master to update parameters immediately after receiving gradients from any single worker. However, asynchronous SGD has worse performance guarantees than synchronous SGD in terms of its error rate [3], [4] because of the high variance in the gradients leading to heavy fluctuations in the objective function. Therefore, a simple variant of SGD was proposed to achieve a trade-off between synchronous SGD and asynchronous SGD, which we name as *Ignore-Straggler SGD*, or simply IS-SGD [5]. With IS-SGD, the master can choose to wait for gradients from an arbitrary number of workers, ignoring the rest of the $s$ stragglers. Fig. 1(c) illustrates an example of IS-SGD with $s = 2$, where $g$ is partially recovered as $g_1 + g_3$. IS-SGD achieves a trade-off between the number of stragglers and the error in convergence [4], [6]–[9]. However, if some worker experiences severe or consistently lower performance, IS-SGD will still make the training biased toward the other dataset partitions on non-straggling workers.

From the discussion above, we can see there is no solution so far that can enjoy the advantages of GC and IS-SGD at the same time, *i.e.*, recovering gradients more completely and tolerating stragglers more flexibly. The solution we propose in this paper is *Ignore-Straggler Gradient Coding* (IS-GC), combining the benefits of IS-SGD and GC. With IS-GC, the master can flexibly determine the number of workers to wait for and recover more gradients than IS-SGD. As shown in Fig. 1(d), through deploying IS-GC, the master can now recover $g_1 + g_2 + g_3 + g_4$ with two available workers. Compared to the IS-SGD in Fig. 1(c), IS-GC fully recovers $g$. Meanwhile, compared to the GC, IS-GC releases the constraint of the number of tolerable stragglers, which is $c - 1 = 1$ in Fig. 1(b), for a full recovery of gradients. In other words, IS-GC may flexibly tolerate more stragglers than GC.

In this paper, we present the coding frameworks of IS-GC for two representative schemes of data placements, fractional repetition (FR) and cyclic repetition (CR). To maximize the recovery of gradients, we analyze the conflict property among workers in each scheme of data placement, since coded gradients from different workers may be conflicted. For example, in Fig. 1(d), $W_1$ is in conflict with $W_2$ since both of them provide $g_2$. Therefore, we propose a model of conflict graph that describes the conflicts among all workers. Based on the conflict graph, we deliberately design the decoding algorithms that are proven to maximize the recovery of gradients with an arbitrary number of available workers.

We demonstrate that FR recovers more gradients while CR allows more flexible choices of parameters. Hence, we further propose hybrid repetition (HR) by generalizing FR and CR and achieving the benefits of both. In other words, HR can recover a similar number of gradients as FR while allowing flexible choices of parameters as CR.

## II. Related Work

To mitigate stragglers in distributed computing based on gradient methods, Harlap *et al.* proposed FlexRR which detects stragglers and avoids stragglers by dynamic workload reassignment [10]. On the other hand, techniques based on replication have been proposed where the same task can be replicated on multiple workers [11]. As replication significantly increases the resource overhead, coding-based methods were proposed where the dataset is encoded to create coded tasks and the master decodes the results from a subset of coded tasks [2], [12]–[15]. However, such methods can only be applied on linear operations in the model.

To apply coding to tasks with general models, gradient coding (GC) was proposed where coding is not applied on the dataset but on the gradients. Random coding [1], cyclic MDS codes [16], and Reed-Solomon codes [17] have been applied to construct GC. Ye and Abbe [18] proposed communication-efficient GC where coding is applied not only among gradients but also within elements of gradient vectors, to reduce the amount of data transferred from workers to the master. While conventionally gradients on multiple dataset partitions are encoded into one single vector with GC, gradients can also be encoded into multiple vectors in order to utilize the resources on stragglers [19]–[21].

GC was originally designed for synchronous SGD, which recovers $\sum_{i=1}^{n} g_i$. Hence, GC can only mitigate a limited number of stragglers, and requires more dataset partitions on each worker to tolerate more stragglers. In practice, however, gradients often do not need to be recovered exactly, and hence IS-SGD (also known as $k$-sync SGD or fastest-$k$ SGD) was proposed where gradients on stragglers can be simply disregarded [4], [22], [23]. In order to recover more gradients with more stragglers $s$ or lower storage overhead $c$, approximate GC is designed to partially recover the gradients with gradient coding [5], [20], [24]–[26]. However, such designs trade off the computation load for a lower $l_2$ error, making it difficult to analyze and understand its convergence property and error rate. More importantly, they are all based on synchronous training, and thus lack the flexibility to tolerate an arbitrary number of stragglers. In a more closely related work [27], approximate gradient coding is constructed to recover the sum of gradients from at least $\delta n$ dataset partitions, $\delta \in (0, 1)$. However, its construction is constrained to specific combinations of parameters only, and the number of stragglers is still limited. IS-GC enjoys a very high level of flexibility, with a more complete recovery of gradients than IS-SGD. We also prove its convergence property in this paper.

## III. Background: Gradient Coding

Consider a dataset $D$ with samples $\{(x_i, y_i)\}_{i=1}^{d}$, $x_i \in \mathbb{R}^p$ and $y_i \in \mathbb{R}$. Assume that we aim to train a model by solving an optimization problem in the form as $\beta^* = \arg\min_{\beta \in \mathbb{R}^p} \sum_{i=1}^{d} f(\beta; x_i, y_i)$. To solve this optimization problem, gradient methods will iteratively update the model from step to step. For example, in the $t$-th step, $\beta^{(t+1)} = \beta^{(t)} - \eta g^{(t)} = \beta^{(t)} - \eta \sum_{i=1}^{n} g_i^{(t)}$ in synchronous SGD.

The original coding scheme of GC was designed for synchronous SGD, where the master will need to recover $\sum_{i=1}^{n} g_i$. To tolerate stragglers, the dataset $D$ is split into $n$ partitions $D_1, \ldots, D_n$ and placed on $n$ workers, and each worker stores $c$ dataset partitions. To place dataset partitions on the $n$ workers, there are two popular placement schemes: fractional repetition (FR) and cyclic repetition (CR).

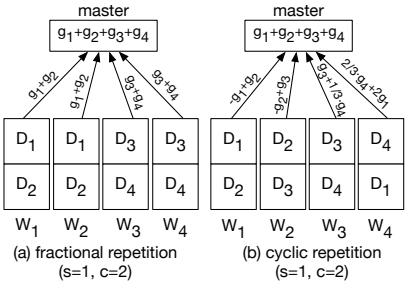



Fig. 2: Examples of FR and CR with $n = 4$.

FR requires $c|n$, so the $n$ workers can be split into $n/c$ groups. In the $i$-th group, the set of dataset partitions on each worker is $\{D_{(i-1)c+1}, D_{(i-1)c+2}, \ldots, D_{ic}\}$. Fig. 2(a) shows an example of FR with $n = 4$ and $c = 2$, where $W_1$ and $W_2$ have $D_1$ and $D_2$, and $W_3$ and $W_4$ have $D_3$ and $D_4$.

CR, however, does *not* require $c|n$. With CR, the dataset partitions are cyclically placed on each worker. On the $i$-th worker $W_i$, the set of its dataset partitions are $\{D_{((j-1) \bmod n)+1} | j = i, \ldots, i + c - 1\}$, $i = 1, \ldots, n$. For example, if $n = 4$ and $s = 1$, the placements of dataset partitions on the 4 workers are illustrated in Fig. 2(b).

With a placement scheme, we then need to consider how each worker should encode its gradients as a linear combination. As for FR, each worker can simply add up all gradients, as shown in Fig. 2(a). Hence, workers in the same group will send the same coded gradients to the master. Since there are $c$ workers in each group, we can recover the sum of gradients in all groups when there are no more than $s = c - 1$ stragglers.

As for CR, the construction of its coding scheme is more non-intuitive, so we only demonstrate one example in Fig. 2(b). In this example, among the coded gradients received from the four workers, the master can recover $g$ when there is no more than $c - 1 = 1$ straggler. Fig. 1(c) shows the same example with $W_2$ as a straggler. Due to the space limit, we omit the details of the code construction, which can be found in [1].

The above examples of GC are designed to recover $g$ exactly, with no more than $s$ stragglers. Although there exist schemes of approximate GC, they either have different convergence properties, or still require an upper bound of the number of stragglers. In the rest of this paper, we will present a framework that maximizes the recovery of $g$ with an arbitrary number of stragglers. In other words, the master can recover $g$ (partially) with any number of workers.

## IV. Coding Framework and Fractional Repetition

With IS-GC, gradients $g$ can be recovered as $\hat{g} = \sum_{i \in I} g_i$, where $I \subseteq \{1, \ldots, n\}$, with an arbitrary number of stragglers. In particular, we aim to maximize $|I|$. With the same number of stragglers, the value of $I$ will be larger than IS-SGD as $c$ can be greater than 1. Note that the value of $|I|$ may vary in different steps as the number of stragglers also varies. We aim to guarantee that for any partition, the chance of its gradients appearing in $\hat{g}$ equals that of any other partition, if the performance of each worker is homogeneous and independent.

We first present the general coding framework of IS-GC, which is independent of the placement scheme. The decoding algorithms which recover $\hat{g}$ in IS-GC are different with different placement schemes. Here, we use FR as an example to demonstrate the decoding algorithm, thanks to its simplicity. The decoding algorithms for CR and HR are much more challenging, which are presented in Sec. V and Sec. VI.

As mentioned in Sec. III, besides the placement, the coding scheme in GC also determines how workers encode their gradients. To maintain the flexibility to tolerate an arbitrary number of stragglers, we ask each worker to simply add up gradients computed on each dataset partition, regardless of the placement scheme. In this way, the coded gradients received from different workers can also be added up so that we can disregard gradients from any workers. Otherwise, if gradients are encoded with non-1 coefficients on each worker, they have to be decoded with coded gradients from some other worker(s), making it impossible to recover $\hat{g}$ with an arbitrary subset of workers. For example, in Fig. 2(b), there is no way to decode the coded gradients into $\hat{g}$ from less than 3 workers. We use $D_{i,j}$ to denote the $j$-th dataset partitions on Worker $W_i$, $j = 1, \ldots, c$, $i = 1, \ldots, n$. With FR, $D_{i,j} = D_{\lfloor (i-1)/c \rfloor c + j}$. We also use $W$ to represent the set of all workers, *i.e.*, $W = \{W_1, \ldots, W_n\}$.

In every step, each worker computes gradients from their dataset partitions and uploads the sum of gradients to the master. The master receives gradients from $n - s$ workers, denoted as $W'$, in each step. Hence, $W' \subseteq W$ and $|W'| = n - s$. The master needs to decode such gradients by running a Decode() function, returning $I$ so that the gradients can be recovered as $\hat{g} = \sum_{i \in I} g_i$. This function varies with different placement schemes. Note that the number of $s$ can be arbitrarily chosen in each step. For example, we can set a deadline in each step, and the master only accepts gradients received before the deadline. Hence, the number of stragglers may change with time. We may also choose to receive gradients from fewer workers at the beginning to save time, and then from more workers afterwards until convergence to make parameters converge more quickly.

In FR, as dataset partitions are duplicated in each of the $n/c$ groups, we can only have gradients from one worker in each group added up. Hence, we only need to count the number of groups that have at least one worker, and choose one worker randomly from each of such groups. We summarize this method in Alg. 1, whose complexity is $O(|W'|)$. Moreover, if the possibility of any worker's gradients being received by the master is equal to each other, we can see that $\forall i \in \{1, \ldots, n\}$, the chance of $i \in I$ equals to each other, too.

**Algorithm 1** The Decode() function for FR.

1: **for** $i = 0, \ldots, n/c - 1$ **do**
2: $I_i = W' \cap \{ic + j | j = 1, \ldots, c\}$
3: Let $v_i$ be one random element in $I_i$
4: **end for**
5: $I = \{v_i | i = 0, \ldots, n/c - 1\}$

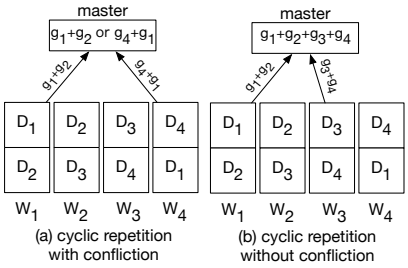



Fig. 3: Comparisons of decoding coded gradients with CR.

## V. Cyclic Repetition

### A. Conflict Graph

As the FR scheme requires that $c|n$, CR offers better flexibility. Compared to FR, the decoding algorithm for CR is not as straightforward. As shown in Fig. 3(a), if the master has received $g_1 + g_2$ from $W_1$, it cannot be added up with $g_4 + g_1$ on $W_4$, or $g_2 + g_3$ on $W_3$. However, if the master receives coded gradients from $W_2$ and $W_4$ after $W_1$, we will disregard $g_1 + g_2$ so that $g_2 + g_3$ and $g_4 + g_1$ can be added up to be $\sum_{i=1}^{4} g_i$. In other words, decoding coded gradients by their received sequence greedily is not optimal.

Given the placement of dataset partitions $\{D_{i,j} | i = 1, \ldots, n, j = 1, \ldots, c\}$, we construct a conflict graph $G = (W, E)$ reflecting if coded gradients on two workers conflict with each other, *i.e.*, if they can be added up. Hence, $W = \{W_i | i = 1, \ldots, n\}$, and $(W_i, W_j) \in E$ if and only if coded gradients on $W_i$ and $W_j$ cannot be added up. Fig. 4, we show two examples of the conflict graphs of FR and CR. We also prove in Theorem 1 that the conflict graph of CR is a circulant graph.

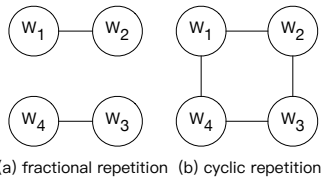



Fig. 4: Conflict graphs of the FR and CR schemes with $n = k = 4$ and $c = 2$.

**Theorem 1.** *The conflict graph of CR is a circulant graph* $C_n^{1,\ldots,c-1}$.

*Proof.* As $D_{i,j} = D_{((i+j-1) \bmod n)+1}$ in CR, $j = 1, \ldots, c$, we can obtain the conflict graph with general values of $n$ and $c$. Coded gradients on a worker $W_i$ can conflict with those on up to $2(c-1)$ other workers. We place all workers sequentially on a circle, and define $d(x, y)$ is the minimal distance between $W_x$ and $W_y$ on the circle, either clockwise or counterclockwise. Hence, $d(x, y) = \min(|x - y|, n - |x - y|) < c$. If $d(x, y) < c$, the gradients on $W_x$ and $W_y$ cannot be added up. The reason is that if $d(x, y) < c$, we can find $z_1$ and $z_2$ such that $D_{x,z_1} = D_{y,z_2}$, which is proved as follows.

We first assume that $0 < |x - y| < c$. Without loss of generality (since we can always switch the order of two workers), we can further assume $x < y$, *i.e.*, $0 < y - x < c$. Then we can let $z_1 = c$ and $z_2 = c - (y - x) \in (0, c)$. Thus $x + z_1 = y + z_2$ and $D_{x,z_1} = D_{y,z_2}$.

We then assume that $0 < n - |x - y| < c$, and further assume that $x > y$ without loss of generality. Hence, we have $n - c < x - y < n$. If we let $z_1 = c$ and $z_2 = c - (n - x + y) \in (0, c)$, then we also have $x + z_1 = y + z_2$.

On the other hand, if $d(x, y) \geq c$, *i.e.*, $c \leq |x - y| \leq n - c$. If we assume that there exist $z_1$ and $z_2$, such that $D_{x,z_1} = D_{y,z_2}$, then we either have $x + z_1 = y + z_2$, or $x + z_1 = (y + z_2 - 1) \bmod n + 1$. In the first case, we have $|z_2 - z_1| = |x - y| \geq c$, which is impossible since $1 \leq z_1, z_2 \leq c$. In the second case, since $(y + z_2 - 1) \bmod n + 1 = y + z_2 - 1 - n + 1 = y + z_2 - n$, we have $|z_2 - z_1 - n| = |x - y| \leq n - c$. This is also impossible since $1 - c - n \leq z_2 - z_1 - n \leq c - 1 - n$. □

Based on the graph model above, we can see that the conflict model directly reflects how we can add up received coded gradients, *i.e.*, when they are from workers that are not connected in the conflict graph. In other words, when we receive coded gradients from $W'$, we can then construct an induced subgraph of $G$, denoted as $G[W']$. In order to maximize $|I|$, we can see that it is equivalent to finding the maximum independent set of $G[W']$, as there are no two vertices in an independent set adjacent to each other. Since each worker has $c$ dataset partitions, we will get the maximum number of $|I|$ from the maximum independent set, which equals to $\alpha(G[W']) \cdot c$ where $\alpha(G[W'])$ denotes the independence number of $G[W']$. Although solving the maximum independent set problem in general is well known to be strongly NP-hard [28], there exists special graphs where their maximum independent set can be found with a polynomial-time complexity [29]–[32]. Below, we demonstrate that the maximum independent set in CR can be found with a linear time complexity.

### B. Decoding Algorithm

Based on the analysis above, we now present the DECODE() function for CR in Alg. 2. In general, we search for the maximal independent set greedily, starting from a random vertex of the conflict graph $G$ (Line 4-12). Theorem 2 proves that a maximal independent set can always be found. However, such a greedy search does not guarantee to find a maximum independent set. In Fig. 4(b), for example, assume that we receive gradients from workers $W_1$, $W_2$, and $W_3$. Starting from $W_1$ we can find a maximum independent set $\{W_1, W_3\}$, while we can only have a maximal independent set $\{W_2\}$ if the search starts from $W_2$. Therefore, we choose the starting vertex clockwise by at most $c$ vertices (Line 3), and at least one maximal independent set must be a maximum independent set (see Theorem 3).

**Theorem 2.** *The greedy search in Line 4-12 can always find a maximal independent set.*

*Proof.* Assuming that we can find a maximal independent set of size $m$ by starting the search from $W_i$, we prove that any independent set containing $W_i$ is not larger than $m$. We write the vertices in the maximal independent set as $\{W_{i_1}, \ldots, W_{i_m}\}$ where $i_1 = i$. Meanwhile, $d(W_{i_l}, W_{i_{l+1}}) \geq c, l = 1, \ldots, m-1$. We assume that there exists an independent set that includes more than $m$ vertices, *i.e.*, $\{W_{j_1} = W_i, \ldots, W_{j'_m}\}$, $m' > m$. Since $i_1 = j_1$, we have $d(i_1, i_2) \leq d(j_1, j_2)$, or $W_{i_2}$ will have no chance to be chosen in Line 10. Furthermore, if $d(j_2, j_3) \geq c$, then $d(i_2, j_3) \geq d(j_2, j_3) \geq c$. Hence, $d(i_1, i_3) \leq d(j_1, j_3)$, or $W_{i_3}$ will still not be chosen in Line 10. We keep applying the previous step, so $d(i_1, i_l) \leq d(j_1, j_l)$, $l = 2, \ldots, m$. Since $d(j_m, j_{m+1}) \geq c$, we also have $d(i_m, j_{m+1}) \geq d(j_m, j_{m+1}) \geq c$, so we can at least apply Line 10 once more, even if the maximal independent set has been found by Alg. 2. Therefore, any independent set that includes $W_i$ will have no more than $m$ vertices. □

**Theorem 3.** *One maximum independent set exists among at most $c$ maximal independent sets found by the greedy search in Alg. 2.*

*Proof.* We can see in Line 3, the greedy search will be performed from at most $c$ starting vertices. Hence, we prove that among vertices in $\hat{W} = W' \cap \{(u+v-1) \bmod n+1 | v = 0, \ldots, c-1\}$, there exists a maximum independent set covering at least one of such vertices.

If there is only one vertex in the maximum independent set, the proof is trivial as we can choose any vertex which will become a maximum independent set. □

**Algorithm 2** Decode() function for the CR scheme.

```
1:  I = ∅
2:  Let u be a random vertex in W'
3:  for i ∈ W' ∩ {(u + v − 1) mod n + 1 | v = 0, ..., c − 1}
    do
4:      I' = {i}
5:      i_0 = i
6:      while j = 1, ..., n − 1 do
7:          next = (i_0 + j − 1) mod +1
8:          if next ∈ W' and d(i_0, next) ≥ c and d(next, i) ≥ c
            then
9:              i_0 = next
10:             I' = I' ∪ {next}
11:         end if
12:     end while
13:     if |I'| > |I| then
14:         I = I'
15:     end if
16: end for
```

As Alg. 2 can find at least one maximum independent set by at most $c$ searches, and the complexity of the greedy search is $O(|W'|/c)$, the complexity of Alg. 2 is $O(|W'|)$. We can further save its complexity by starting the search from isolated vertices. Moreover, as vertices are isolated independently with an equal chance, gradients on each worker still have an equal chance to be added into $\hat{g}$ eventually.

### C. Tradeoff between FR and CR

Despite FR and CR sharing the same upper and lower bounds of $\alpha(G[W'])$, we also find that $\alpha(G[W'])$ in FR is expected to be higher than CR.

Considering an instance of the CR scheme with given $n$ and $c$, we denote its conflict graph as $G_{CR(n,c)} = (W, E_{CR(n,c)})$. Similarly, We use $G_{FR(n,c)} = (W, E_{FR(n,c)})$ to denote the conflict graph of the FR scheme with given $n$ and $c$, $c|n$. Although CR offers higher flexibility in terms of the value of $c$, we show that FR can recover more gradients by Theorem 4.

**Theorem 4.** $E_{FR(n,c)} \subset E_{CR(n,c)} \subset \cdots \subset E_{CR(n,n)}$.

*Proof.* If $(W_i, W_j)$ is an edge in $G_{CR(n,c)}$, we have $d(i,j) < c < c+1$, so it is also an edge in $G_{CR(n,c+1)}$, *i.e.*, $E_{CR(n,c)} \subset E_{CR(n,c+1)}$.

Given an edge $(W_i, W_j)$ in $G_{FR(n,c)}$, where we assume $i < j$ without loss of generality, there exists a positive integer $l$ such that $c(l-1)+1 \leq i < j \leq cl$. Hence, $d(i,j) \leq j - i \leq c$, and thus $(W_i, W_j)$ is also an edge in $G_{CR(n,c)}$, *i.e.*, $E_{FR(n,c)} \subset E_{CR(n,c)}$. Summarizing the two properties above, we have $E_{FR(n,c)} \subset E_{CR(n,c)} \subset \cdots \subset E_{CR(n,n)}$. □

When the master receives coded gradients from workers in $W'$ where $W' \subseteq W$. From the property above, we know that $G_{FR(n,c)}[W']$ is also a subgraph of $G_{CR(n,c)}[W']$, and thus also has a higher independence number, *i.e.*, $\alpha(G_{FR(n,c)}[W']) > \alpha(G_{CR(n,c)}[W'])$. In other words, with coded gradients from any workers, we can expect that FR can recover more gradients than CR.

## VI. Hybrid Repetition

From the analysis above, we can see that FR limits the flexible choices of $c$, especially when $n$ does not have many factors. Meanwhile, CR can support an arbitrary value of $c$, as long as $c \leq n$, but the recovered gradients are proved to be fewer than FR. Hence, in this section, we propose hybrid repetition (HR), a new placement scheme, that enjoys both CR's flexibility and FR's better recovery of gradients.

### A. Intuition and a Special Case

In order to achieve the desirable properties of both FR and CR, we need a placement scheme where the values of $c$ can be arbitrarily chosen, and meanwhile the tradeoff between FR and CR can be flexibly achieved. We first present an example of the HR scheme to demonstrate its intuition. In Fig. 5(a), we illustrate an example of the FR scheme with $n = 4$ and $c = 2$, where the $4$ workers are split into $2$ groups, and the $4$ dataset partitions are also split disjointedly into the $2$ groups. If we look at such two groups, we can see that the placement of the two dataset partitions can also be considered as a special case of the CR scheme with $n = c = 2$. As shown in Fig. 5(b), the placement of dataset partitions in the two groups follows the CR

scheme. As we only change the sequences of dataset partitions on all workers, the conflict graphs of the two placements are also the same.

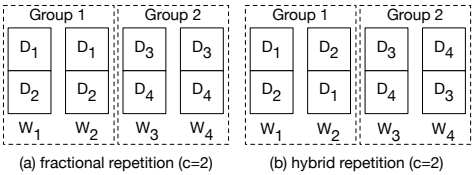



Fig. 5: Examples of hybrid repetition.

Following this intuition, we first present a special case of the HR scheme. Given $n$ and $c$, we assume that there exists an integer $g$ such that $g|n$, and the $n$ workers are equally split into $g$ groups. The $n$ dataset partitions are also equally split into the $g$ groups. In each group, we disjointly place $n_0 = n/g$ dataset partitions where each worker stores $c$ dataset partitions, following the CR scheme, as shown in Fig. 6. The example above corresponds to the case with $g = 2$. In this way, the conflicts among workers are limited within each group.

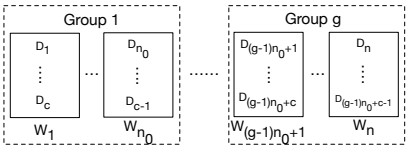



Fig. 6: The construction of the HR scheme.

Note that the value of $g$ does not necessarily equal $c$. In the HR scheme, we assume that $g \leq n/c$ (since the CR scheme requires that $c \leq n_0$). In general, as dataset partitions are disjoint in different groups, the HR scheme above can be decoded group by group, and the decoding algorithm for CR can be applied in each group.

As a special case, by Theorem 5 below, the decoding algorithm can be simplified to be the same as that of FR.

**Theorem 5.** *The conflict graph of HR is the same as* $G_{FR(n,n_0)}$ *when* $n_0 \leq 2c - 1$.

*Proof.* Since the dataset partitions are disjointedly placed in different groups, we only consider the conflict between workers in the same group, as no workers in different groups can have conflict. Since the dataset partitions in the same group are placed following the CR scheme, its conflict graph is also a circulant graph. Since each worker conflicts with $2(c-1)$ workers at most, when $2(c-1)+1 \geq n_0$, each pair of workers has a conflict. In other words, the conflict graph in each group is a complete graph, and the conflict graph of all workers becomes the same as that of an FR scheme where $c = n_0$. □

Moreover, when $g = 1$, we can also see from the construction above becomes the same as CR. In fact, FR and CR are just two extreme points in a spectrum of the placement of HR. Below we further generalize HR which eventually achieves the tradeoff between FR and CR.

## B. Generalization

To generalize the placement above, we still assume that the values of $n$ and $c$ are given, and there exists a positive integer $g$ such that $g|n$. The $n$ workers are equally split into such $g$ groups. We illustrate the general construction in Fig. 7. We denote the construction above as $\text{HR}(n, n_0, 0)$, as a special case of the following general construction $\text{HR}(n, c_1, c_2)$ where $c_1 = n_0$ and $c_2 = 0$.

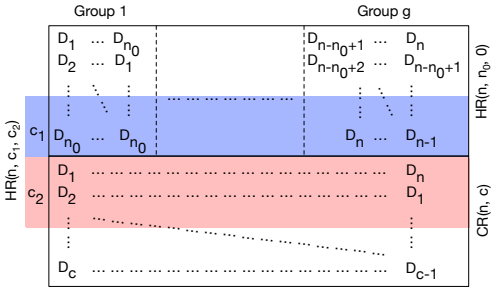



Fig. 7: An illustration of the general HR scheme.

As shown in Fig. 7, the placement in the general HR scheme has two parts. The upper part is a placement of $\text{HR}(n, n_0, 0)$, and the lower part is a placement of the CR scheme with the given $n$ and $c$. The actual placement of a $\text{HR}(n, c_1, c_2)$ placement is then a combination of such two parts, by choosing the bottom $c_1$ rows in $\text{HR}(n, n_0, 0)$ and the top $c_2$ rows in the CR scheme, where $c_1 + c_2 = c$. Since the placement becomes a CR scheme when $c_1 = 0$, we assume $c_1 > 0$ below.

In particular, we can also prove that when $n_0 = c$, $\text{HR}(n, c, 0)$ is equivalent to $\text{HR}(n, c-1, 1)$. From Fig. 7, we can see that the first rows in the upper part and the lower part are both from 1 to $n$. From $\text{HR}(n, c, 0)$ to $\text{HR}(n, c-1, 1)$, we remove the first row in the upper part and add the first row in the lower part, and thus it does not change the dataset partitions on each worker.

Theorem 6 below gives the range of $n_0$ in HR. With its help, we can further show in Theorem 7 that HR achieves a tradeoff between the special case above and CR.

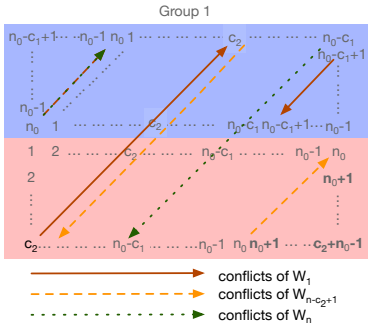



Fig. 8: An illustration of placement in a group with HR.

**Theorem 6.** *In* $\text{HR}(n, c_1, c_2)$, $c \leq n_0 \leq 2c - 1$.

*Proof.* In order to guarantee that all workers in the same group conflict with each other, the values of $c$, $c_1$, and $n_0$ should be

chosen such that $c + c_1 \geq n_0$. To prove this property, we only need to take a look at the placement of any group, thanks to the symmetry in the HR scheme. Without loss of generality, we consider the first group in Fig. 8. For convenience, we only show the subscripts of the dataset partitions, *e.g.*, $i$ in Fig. 8 denotes the dataset partition $D_i$. We can see that each entry may conflict with entries on the same anti-diagonal. Moreover, for entries between 1 and $n$, the entries that have a conflict can go up from the right end back to the left end. However, for other entries above $n$, they only appear in the lower part corresponding to the CR scheme. Therefore, from $W_1$ to $W_{n_0-c_2+1}$, there are $c_1 + c_2 - 1 = c - 1$ workers that have conflicts on its left, and the same number of workers on its right. However, other workers have fewer workers conflicting on its right. In the worst case, which is $W_{n_0}$, it only has $c_1$ workers conflicting on its right. Hence, in each group, each worker has at least $c+c_1-1$ conflicting workers, so $n_0 \leq c+c_1$.

Since $c > c_1$, we can further have $c > c_1 \geq n_0 - c$, and hence $n_0 < 2c$. Since $n_0 \leq 2c - 1$, each worker conflicts with all other workers in the HR scheme (when $c_1 > 0$). Also, as $c_1 \leq n_0$ in the upper part of the placement, we now get the valid range of $n_0$ as $c \leq n_0 \leq 2c - 1$. □

**Theorem 7.** $E_{HR(n,c,0)} \subseteq E_{HR(n,c-1,1)} \subseteq \cdots \subseteq E_{HR(n,n_0-c,2c-n_0)}$

*Proof.* As all workers in the same group are conflicting with each other, we only need to analyze the conflict between workers in different groups, which we name as outside neighbors. Since $g \geq c$, if two workers are not in the two neighboring groups, they are not conflicting, as any dataset partition may only have its replication at most $c - 1$ workers away. Without loss of generality, we take a look at the first two groups in Fig. 9.

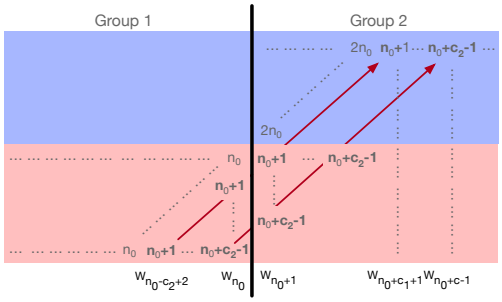



Fig. 9: An illustration of the conflict between two nearby groups.

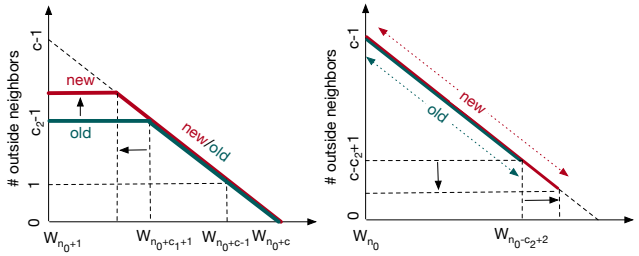



Fig. 10: The intuition of the function of outside neighbors.

First, we consider the outside neighbors on the right of a group and look at the outside neighbors of workers of Group 1 in Group 2. We can see that in Group 2, workers between $W_{n_0+1}$ and $W_{n_0+c_1+1}$ conflict with the $c_2 - 1$ workers on the right in Group 1. Furthermore, workers between $W_{n_0+c_1+1}$ and $W_{n_0+c-1}$ have their numbers of outside neighbors decreasing from $c_2 - 1$ to 1.

On the other hand, we consider the outside neighbors on the left of a group, and hence we look at workers in Group 1 as outside neighbors of workers in Group 2. Looking at the bottom dataset partition between $W_{n_0}$ and $W_{n_0-c_2+2}$, we can see their replicas can reach at most $c - 1$ workers on the right, and hence the number of outside neighbors decreases from $c - 1$ to $c - 1 - (c_2 - 1) = c - c_2$.

We show the number of outside neighbors as a function of workers in Fig. 10. With a fixed $c$, we can move the HR schemes between HR$(n, c, 0)$ and HR$(n, n_0-c, 2c-n_0)$ which becomes HR$(n, 0, c)$ when $n_0 = c$. When we decrease the value of $c_1$ (and increase the value of $c_2$ simultaneously), we can see how the two parts of the functions change in Fig. 10, and thus all edges in the conflict graph of HR$(n, c_1, c_2)$ are also edges in the conflict graph of HR$(n, c_1-1, c_2+1)$, *i.e.*, the conflict graph of HR$(n, c_1, c_2)$ is a subgraph of HR$(n, c_1-1, c_2+1)$. Therefore, $E_{HR(n,c,0)} \subseteq E_{HR(n,c-1,1)} \subseteq \cdots \subseteq E_{HR(n,n_0-c,2c-n_0)}$. □

When $n_0 = c$, we have $E_{FR(n,c)} = E_{HR(n,c,0)} \subseteq \cdots \subseteq E_{HR(n,0,c)} = E_{CR(n,c)}$. Hence, HR can be seen as a generalization of FR and CR and achieve a tradeoff between them.

### C. Decoding

When $c_2 = 0$, the conflict graph of the HR scheme is the same as the FR scheme with $c = g$. Hence, it can be decoded with the same algorithm. We now consider the decoding algorithm when $c_2 > 0$, which can be modified from Alg. 2.

**Algorithm 3** Decode() function for the HR scheme.

$I = \emptyset$
Let $i$ be a random integer, $i \in [0, g-1]$, such that $I_i = W' \cap \{in_0 + j | j = 1, \ldots, n_0\}$ is not an empty set
**for all** $i \in I_i$ **do**
  $I' = \{i\}$
  $i_0 = i$
  **while** $j = 1, \ldots, n-1$ **do**
    $next = (i_0 + j - 1) \bmod +1$
    **if** $next \in W'$ and (not CONFLICT$(i_0, next)$) and (not CONFLICT$(next, i)$) **then**
      $i_0 = next$
      $I = I \cup \{next\}$
    **end if**
  **end while**
  **if** $|I'| > |I|$ **then**
    $I = I'$
  **end if**
**end for**

**Algorithm 4** CONFLICT$(i_1, i_2)$ function for the HR scheme.

```
1: g_1 = ⌊i_1/n_0⌋, g_2 = ⌊i_2/n_0⌋
2: if g_1 = g_2 then
3:     return True
4: end if
5: if (g_2 − g_1) mod g = 1 then
6:     j_1 = i_1 mod n_0
7:     if (j_1 ≥ n_0 − c_2 + 1) and ((i_2 − i_1) mod n < c) then
8:         return True
9:     end if
10: end if
11: return False
```

Compared to Alg. 2, the decoding algorithm for HR in Alg. 3 mainly has two differences. First, instead of starting the greedy search from at most $c$ vertices, we can easily limit the starting vertices within any single group (Line 3 in Alg. 2 and Line 2-3 in Alg. 3). We show in Theorem 8 that a maximum independent set can be found that covers at least one such vertex in such a group.

**Theorem 8.** *$\forall u \in [1, n]$, defining $\hat{W}$ as the set of workers in $W'$ that are in the same group, there exists a maximum independent set covering at least one of such vertices in $\hat{W}$.*

*Proof.* We assume that there only exist maximum independent sets without covering any vertex in $\hat{W}$. We use the same technique used in the proof of Theorem 3. Given any one of such maximum independent sets, we can find the two nearest vertices on the two sides of vertices in $\hat{W}$, and we can remove one of such vertices from the maximum independent set and add the vertex that is nearest to it in $\hat{W}$. Hence, we build a maximum independent set that covers one vertex in $\hat{W}$. □

Second, in the greedy search, the criteria of conflict is not just depending on the distance (Line 8 in Alg. 2 and Alg. 3). The new criteria is defined in Alg. 4. Specifically, conflict exists if two workers are in the same group, Moreover, when they are in two nearby groups, there also exists conflict if a worker has the same gradients as those on the other group. For example, in Fig. 7, $D_1$ appears in both Group 1 and Group $n$. Theorem 9 demonstrates that Alg. 3 can always find a maximal independent set. Similar to Alg. 2, as long as $i$ in Line 3 is randomly permutated, gradients on each worker have an equal chance to be added into $\hat{g}$.

**Theorem 9.** *A maximal independent set can be found by a greedy algorithm in Alg. 3 and Alg. 4.*

*Proof.* We still prove that if a maximal independent set exists, we can use a greedy algorithm to find the same one or another maximal independent set of larger size starting from any vertex in the original one. The proof is based on an observation: given two vertices $u$ and $v$ in $W'$, if $u$ proceeds $v$ in the clockwise direction, there exists no vertex $w$ such that $u$ and $w$ are conflicting but $v$ and $w$ are not.

Assume if $w$ exists, we consider if $w$ is in the same group of $u$. If so, there exists no such a vertex $v$ as all vertices in the same group conflict with each other. Otherwise, we can see, from Fig. 8, that there are only the $c_2 - 1$ workers on the right that can conflict with workers in the next group, as $c_2 - 1 < c \leq g$, and thus $w$ must be in the next group in the clockwise direction. We can also see that the dataset partitions that may cause conflicts come from the bottom right corner in Fig. 8, which is a part of the CR scheme. Therefore, if $w$ and $u$ are conflicting, $w$ and $v$ must also be conflicting. Summarizing the two cases above, such a vertex $w$ does not exist. We then can extend Alg. 2 as the decoding algorithm for the HR scheme, by only making two changes. First, in Line 3, the greedy searches should start from all vertices in $W'$ that are in the same random group. Second, in Line 8, the conflict between $i_0$ and $next$ and the conflict between $next$ and $i$ should be identified differently, where the CONFLICT$(i_1, i_2)$ function returns true if $i_1$ and $i_2$ are conflicting. □

## VII. Theoretical Analysis

In this section, we theoretically analyze the performance of IS-GC. We first discuss the performance of decoding of IS-GC on FR and CR, in terms of the recovered gradients, and then analyze the convergence property of IS-GC.

### A. Recovered Gradients

To analyze the performance of decoding, we consider a question: what is the worst-case and best-case performance of decoding when there are $s$ stragglers? For convenience, we define $w = n - s$, *i.e.*, the number of available workers, or $|W'|$. Given $w$ workers in $W'$, we aim to find the upper and lower bounds of $\alpha(G[W'])$. It is interesting to find that FR, CR, and HR share the same bounds despite their different placement and decoding algorithms.

**Theorem 10.** $\alpha(G[W']) \geq min(\lceil \frac{w}{c} \rceil, \lfloor \frac{n}{c} \rfloor)$.

*Proof.* The value of $\alpha(G[W'])$ depends on the conflict among workers. The conflict in FR is simple. Two workers are in conflict if they are from the same group. With a general $w$, the worst case of $\alpha(G[W'])$ should be obtained when the $w$ workers come from as few groups as possible, which is $\lceil \frac{w}{c} \rceil$. As $c|n$ in the FR scheme, $\lceil \frac{w}{c} \rceil \leq \frac{n}{c} = \lfloor \frac{n}{c} \rfloor$.

As for CR, there are no such obvious "groups" due to the round-robin placement of dataset partitions. However, we can still see that the worst case is when all workers in $W'$ are consecutive. Given $w$ such consecutive workers, the first worker will rule out the next $c - 1$ workers, the $(c + 1)$-th worker will also rule out the next $c - 1$ ones, *etc*. Therefore, we can also have at least $\lceil \frac{n}{c} \rceil$ workers in $I$. Moreover, as CR does not require $c|n$, at most $\lfloor \frac{n}{c} \rfloor$ workers can have no conflict with each other.

Finally, as the conflict in HR is between FR and CR, its $\alpha(G[W'])$ has the same lower bound when FR and CR have the same lower bound. □

**Theorem 11.** $\alpha(G[W']) \leq min(w, \lfloor \frac{n}{c} \rfloor)$.

*Proof.* In FR and CR, the best case of $\alpha(G[W'])$ corresponds to the case when all workers have conflict as least as possible, or no conflict at all. Similar to the proof of Theorem 10, when $w \leq \lfloor \frac{n}{c} \rfloor$, the $w$ workers may have no conflict at all, either in different groups in FR, or being far away from each other in CR. All the $w$ workers can then be in $I$, *i.e.*, $\alpha(G[W']) = w$. When there are more workers, $\alpha(G[W'])$ is again limited to the maximum number of workers that can be in $I$, *i.e.*, $\alpha(G[W']) \leq \lfloor \frac{n}{c} \rfloor$. Again, as FR and CR have the same upper bound, the upper bound of HR is also the same. □

### B. Convergence Analysis

To obtain the convergence property in IS-GC, we make the following assumptions for its loss function $f(\beta; x, y)$.

**Assumption 1.** $\nabla f(\beta; x, y)$ is $L$-Lipschitz continuous such that with $L > 0$ we have $||\nabla f(\beta_1; x, y) - \nabla f(\beta_2; x, y)||_2 \leq L||\beta_1 - \beta_2||_2, \forall \beta_1, \beta_2 \in \mathbb{R}^p$.

As stragglers may differ from step to step, $\alpha(G[W'])$ may also vary in different steps. Hence, we define $D_d^{(t)}$ as the samples involved in the gradients after decoding in the $t$-th step, such that $|D_d^{(t)}| = \alpha(G[W']) \cdot \frac{cd}{n}$.

**Assumption 2.** The decoded gradients are an unbiased estimate of the true gradients: $\mathbb{E}_{D_d^{(t)}}[g(\beta^k, D_d^{(t)})] = \frac{1}{|D_d^{(t)}|} \sum_{i \in D_d^{(t)}} \nabla f(\beta^{(k)}; x_i, y_i) = \frac{1}{d} \sum_{i=1}^{d} \nabla f(\beta^{(k)}; x_i, y_i) = \nabla f(\beta^{(k)})$.

As FR and CR both guarantee that all data samples have the same probability in the decoded gradients. Therefore, IS-GC conforms to this assumption.

**Assumption 3.** We assume that the variance of the gradients joining the model update has an upper bound. Thus, $E_{D_d^{(t)}}[||g(\beta^{(t)}, D_d^{(t)})||^2] \leq \sigma^2$.

We now prove that the model will converge with IS-GC.

**Theorem 12.** *In IS-GC, we have* $E_{D_d^{(t)}}[f(\beta^{(t+1)})] \leq f(\beta^{(t)}) - \eta|D_d^{(t)}|||\nabla f(\beta^{(t)})||^2 + \frac{L\eta^2\sigma^2|D_d^{(t)}|^2}{2}$.

*Proof sketch.* By Assumption 1, $\frac{||\nabla f(\beta^{(t+1)}) - \nabla f(\beta^{(t)})||_2}{\beta^{(t+1)} - \beta^{(t)}||_2} \leq L$. In other words, $\nabla^2 f(\beta^{(t)}) \leq L$. By Taylor series, we can find that $f(\beta^{(t+1)}) - f(\beta^{(t)}) \leq f^{(1)}(w^k)||w^{k+1} - w^k|| + \frac{L}{2}||w^{k+1} - w^k||^2$. We then take the expectation for the formula above. After that, replace $w^{k+1} - w^k$ with the update rule in the expectation formula. After replacing some terms based on the corresponding assumptions, we can find that $E_{D_d^{(t)}}[f(\beta^{(t+1)})] \leq f(\beta^{(t)}) - \eta|D_d^{(t)}|||\nabla f(\beta^{(t)})||^2 + \frac{L\eta^2\sigma^2|D_d^{(t)}|^2}{2}$. □

We have already shown that $D_d^{(t)}$ is bounded in Sec. VII-A. Hence, Theorem 12 implies that when the learning rate $\eta$ is small enough, the perturbation from $L\eta^2\sigma^2|D_d^{(t)}|^2$ does not affect the convergence.

## VIII. Evaluation

### A. Implementation

We implement IS-GC using Ray, a distributed computing framework for AI applications [33]. On each worker, we maintain multiple copies of the same model for its different dataset partitions. In each step, a worker loads a mini-batch from each dataset partition and uses it to train the model. We carefully control all random seeds so that data in each batch are always the same in the same dataset partition. Afterwards, we can get gradients from the optimizer and encode gradients of all copies of the model with the corresponding coding scheme. All workers then send coded gradients to a master. The master receives coded gradients from $w$ fastest workers (using the ray.wait() function). After decoding, the master will broadcast decoded gradients to all workers, which will be used to update the parameters of all copies of the model.

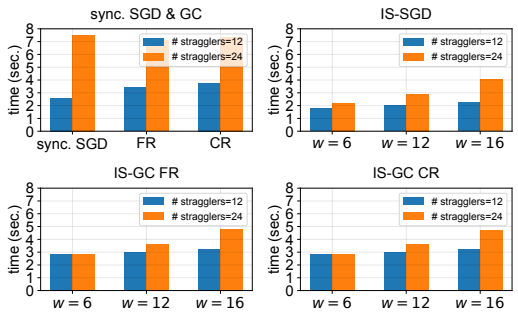
We also implement GC with FR and CR. With $c$ given, GC only tolerates $c - 1$ stragglers, *i.e.*, the master must receive gradients from $w = n - c + 1$ fastest workers. When $c = 1$, the three placement schemes become the same where the $n$ dataset partitions are simply placed on $n$ workers. GC, in this case, becomes synchronous SGD. Similarly, IS-GC becomes IS-SGD when $c = 1$. We also use the same random seeds in different schemes so that the same values of parameters are initialized in the model to make the comparisons fair.

### B. Simulation

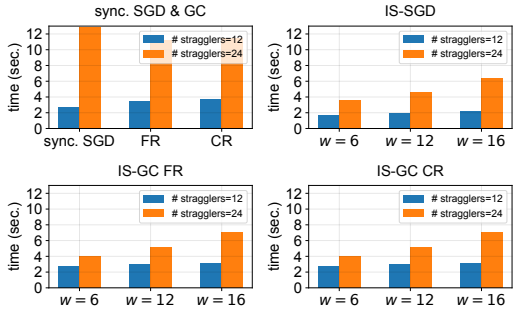
We first run a workload of training ResNet-18 on the ImageNet dataset on a local HPC with 24 workers, with 64 as the batch size. We focus on the performance of tolerating stragglers in the training. Hence, we simulate stragglers by adding an arbitrary delay before sending (coded) gradients to the master from 12 or 24 workers. The delay is generated randomly following an exponential distribution, based on the measurements from real cloud workloads [1], [2].

We present the average time per step in Fig. 11. We set $c = 2$ in GC and IS-GC, and thus GC can only tolerate one straggler. When the expected delay is 1.5 seconds, we can see in Fig. 11(a) that synchronous SGD and GC suffer significantly from such stragglers. Even worse, GC consumes much more time than synchronous SGD due to a higher $c$. On the other hand, IS-GC consumes significantly less time per step thanks to its flexibility in the values of $w$, by up to 74.9%.

It is not surprising to see that the average time per step with IS-GC is higher than IS-SGD, because of its higher value of $c$. However, when we increase the expected delay to 3 seconds in Fig. 11(b), we can see that the difference can be reduced to less than 10%, showing that the overhead is marginal when workers are more straggling. Although a higher $c$ does not help save the per-step time, it does help make the training faster with fewer steps before convergence, which will be shown in the experiments below. Similarly, FR and CR in IS-GC also have very similar per-step time as their workloads are very similar on each worker besides the different placement of dataset partitions. Hence, we omit HR and leave the comparison to Sec. VIII-C as well.




(a) delay $\sim exp(1/1.5)$ sec.




(b) delay $\sim exp(1/3)$ sec.

Fig. 11: Average time per step of training ResNet-18 on ImageNet.

### C. Experiments in the Cloud

We train a ResNet-18 using CIFAR-10 on Google Cloud. Before each experiment, we launch a Ray cluster on $n+1$ virtual machines of type n1-standard-4 with NVIDIA Tesla P100 GPUs, where $n$ is the number of workers. We choose 128 as the batch size, 0.006 as the learning rate, and $n = 4$. The optimizer is torch.optim.SGD. We train the model until the training loss reaches a given threshold. In Fig. 12, we present performance comparisons in training with different values of $w$. The data presented is the average result of 10 trials.

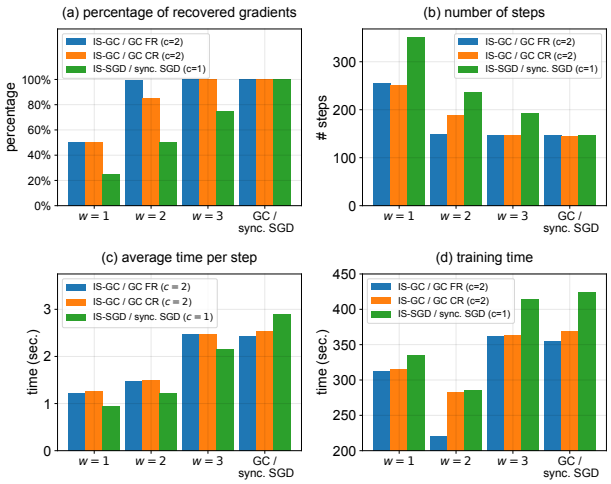



Fig. 12: Performance comparisons of training ResNet-18 on CIFAR-10.

**Percentage of gradients recovered.** Synchronous SGD and GC achieve fully recover of the gradients over all samples in the mini-batches, so we use the number of samples as the baseline and compare the percentage of samples in the recovered gradients in IS-GC and IS-SGD. In Fig. 12(a), we can see that with the increase of $w$, all schemes can recover more gradients with more samples, *i.e.*, more gradients are recovered. However, with a higher value of $c$, IS-GC can recover more samples in the gradients. When $w = 3$, IS-GC can fully recover gradients, the same as GC. When $w = 2$, GC cannot work but IS-GC can still recover gradients from up to $99.6\%$ of samples, which is even higher than expected thanks to an enduring straggler. We can also observe that FR can recover gradients better than CR when $w = 2$. When $w = 1$, FR is equivalent to CR as gradients are from one worker only. When $w = 3$, both FR and CR can fully recover the gradients.

**Number of steps.** Based on the results above, we can further observe in Fig. 12(b) that IS-GC can save the number of steps needed to train the models. When gradients are fully recovered, the minimum number of steps is achieved as 146. While the number of steps increases with the decrease of $w$, IS-GC can save the number of training steps by up to $37.1\%$ with each value of $w$. Meanwhile, FR also outperforms CR when $w = 2$, thanks to its better recovery of gradients.

**Average time per step.** It is not surprising to see in Fig. 12(c) that the average time per step in IS-GC is higher than IS-SGD with different values of $w$, because of different values of $c$. However, IS-GC does not increase the time by $c$ times. Therefore, we infer that most time is spent on uploading gradients to the master in this experiment, and hence stragglers are more likely to be caused by communication. Eventually, the saving of training steps compensates for the additional time incurred by IS-GC in each step, as illustrated in Fig. 12(d).

**Training time.** Finally, we observe in Fig. 12(d) that with the same values of $w$, IS-GC takes significantly less time than IS-SGD. The lowest training time is achieved when $w = 2$. With a higher $w$, the saving of time becomes less significant because fewer stragglers can be tolerated and IS-GC needs to spend more time at each step. When $w$ is lower, the higher number of steps needed outweighs the saved time in each step. Moreover, we can see that FR finishes training more quickly than CR when $w = 2$, as expected from the analysis above.

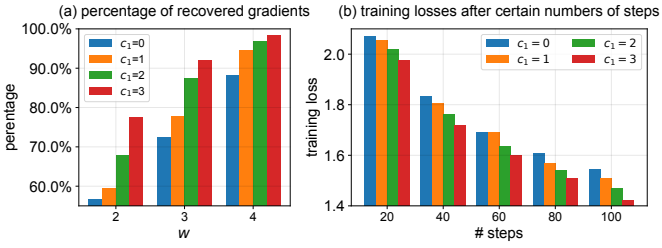



Fig. 13: Tradeoffs of gradients and losses achieved by HR.

**Tradeoffs in HR.** From the results above, we can see that FR recovers gradients better than CR when $w = 2$ in Fig. 12(a). Correspondingly, FR lowers the training time by $22.0\%$ when

$w = 2$ in Fig. 12(d). We now more carefully study how HR achieves a tradeoff between them. In Fig. 13, we choose $c = 4$ and $g = 2$, and construct HR$(8, c_1, 4 - c_1)$, where $0 \leq c_1 \leq 3$. When $c_1 = 0$, the HR scheme becomes a CR scheme. The placement when $c_1 = 3$ is the same as $c_1 = 4$, which is also the same as FR. We train the ResNet-18 on CIFAR-10 with $n = 8$ workers, by choosing the learning rate as $0.001$ and the batch size as $128$. We can see in Fig. 13 that in terms of both the number of recovered gradients and the accuracy, HR generalizes over FR and CR and achieves a tradeoff between them. Specifically, Fig. 13(a) shows a more clear trend that more gradients are recovered when $c_1$ increases. We further look at the corresponding training losses with $w = 2$ in Fig. 13(b). By comparing the training losses at different steps, we can further confirm that more gradients recovered help improve the progress of training.

## IX. Conclusion

In this paper, we present IS-GC for mitigating stragglers in distributed training with gradient methods. Compared to existing schemes including GC and IS-SGD, IS-GC achieves their advantages at the same time, by tolerating more stragglers flexibly and recovering more gradients. We apply IS-GC on two existing placement schemes, FR and CR. We then further design HR, a more generalized placement scheme that achieves a flexible tradeoff between FR and CR.

## References


[1] R. Tandon, Q. Lei, A. G. Dimakis, and N. Karampatziakis, "Gradient Coding: Avoiding Stragglers in Distributed Learning," in *Proceedings of International Conference on Machine Learning (ICML)*, vol. 70, 2017, pp. 3368–3376.

[2] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Speeding Up Distributed Machine Learning Using Codes," in *IEEE Transactions on Information Theory*, vol. 64, no. 3, 2018, pp. 1514–1529.

[3] F. Niu, B. Recht, C. Ré, and S. J. Wright, "HOGWILD!: A Lock-free Approach to Parallelizing Stochastic Gradient Descent," *Proceedings of the 24th International Conference on Neural Information Processing Systems (NIPS)*, 2011.

[4] S. Dutta, G. Joshi, S. Ghosh, P. Dube, and P. Nagpurkar, "Slow and Stale Gradients can Win the Race: Error-Runtime Trade-offs in Distributed SGD," in *Proceedings of International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2018, pp. 803–812.

[5] R. Bitar, M. Wootters, and S. E. Rouayheb, "Stochastic Gradient Coding for Straggler Mitigation in Distributed Learning," *IEEE Journal on Selected Areas in Information Theory*, vol. 1, no. 1, pp. 277–291, 2020.

[6] S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter, "PEGASOS: Primal Estimated Sub-gradient Solver for SVM," *Mathematical Programming*, vol. 127, no. 1, pp. 3–30, 2011.

[7] K. Gimpel, D. Das, and N. A. Smith, "Distributed Asynchronous Online Learning for Natural Language Processing," in *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, 2010, pp. 213–222.

[8] S. Shalev-Shwartz and A. Tewari, "Stochastic Methods for $L_1$-regularized Loss Minimization," *Journal of Machine Learning Research*, vol. 12, pp. 1865–1892, 2011.

[9] L. Bottou, "Online Learning and Stochastic Approximations," *On-line Learning in Neural Networks*, vol. 17, no. 9, p. 142, 1998.

[10] A. Harlap, H. Cui, W. Dai, J. Wei, G. R. Ganger, P. B. Gibbons, G. A. Gibson, and E. P. Xing, "Addressing the Straggler Problem for Iterative Convergent Parallel ML," *Proceedings of the 7th ACM Symposium on Cloud Computing (SoCC)*, pp. 98–111, oct 2016.

[11] D. Wang, G. Joshi, and G. Wornell, "Using Straggler Replication to Reduce Latency in Large-scale Parallel Computing," *ACM SIGMETRICS Performance Evaluation Review*, vol. 43, no. 3, pp. 7–11, nov 2015.

[12] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Polynomial Codes: an Optimal Design for High-Dimensional Coded Matrix Multiplication," *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS)*, 2017.

[13] ——, "Straggler Mitigation in Distributed Matrix Multiplication: Fundamental Limits and Optimal Coding," *IEEE Transactions on Information Theory*, vol. 66, no. 3, pp. 1920–1933, 2020.

[14] S. Dutta, V. Cadambe, and P. Grover, "'Short-Dot': Computing Large Linear Transforms Distributedly Using Coded Short Dot Products," *IEEE Transactions on Information Theory*, vol. 65, no. 10, pp. 6171–6193, 2019.

[15] P. Soto, J. Li, and X. Fan, "Dual Entangled Polynomial Code: Three-Dimensional Coding for Distributed Matrix Multiplication," in *Proceedings of the 36th International Conference on Machine Learning*, 2019, pp. 5937–5945.

[16] N. Raviv, I. Tamo, R. Tandon, and A. G. Dimakis, "Gradient Coding from Cyclic MDS Codes and Expander Graphs," *IEEE Transactions on Information Theory*, vol. 66, no. 12, pp. 7475–7489, 2020.

[17] W. Halbawi, N. Azizan, F. Salehi, and B. Hassibi, "Improving Distributed Gradient Descent Using Reed-Solomon Codes," in *Proceedings of IEEE International Symposium on Information Theory (ISIT)*. Institute of Electrical and Electronics Engineers Inc., 2018, pp. 2027–2031.

[18] M. Ye and E. Abbe, "Communication-computation Efficient Gradient Coding," in *Proceedings of 35th International Conference on Machine Learning (ICML)*, 2018, pp. 5610–5619.

[19] E. Ozfatura, S. Ulukus, and D. Gunduz, "Distributed Gradient Descent with Coded Partial Gradient Computations," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 2019-May. Institute of Electrical and Electronics Engineers Inc., 2018, pp. 3492–3496.

[20] ——, "Coded Distributed Computing with Partial Recovery," 2020. [Online]. Available: http://arxiv.org/abs/2007.02191

[21] H. Wang, S. Guo, B. Tang, R. Li, Y. Yang, Z. Qu, and Y. Wang, "Heterogeneity-aware Gradient Coding for Tolerating and Leveraging Stragglers," *IEEE Transactions on Computers*, 2021.

[22] J. Chen, X. Pan, R. Monga, S. Bengio, and R. Jozefowicz, "Revisiting Distributed Synchronous SGD," in *Proceedings of International Conference on Learning Representations Workshop Track*, 2016.

[23] S. K. Hanna, R. Bitar, S. E. Rouayheb, P. Parag, and Venkat Dasari, "Adaptive Distributed Stochastic Gradient Descent For Minimizing Delay In The Presence Of Stragglers," in *Proceedings of the 45th International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2020.

[24] S. Wang, J. Liu, and N. Shroff, "Fundamental Limits of Approximate Gradient Coding," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 3, no. 3, pp. 1–22, 2019.

[25] Z. Charles, D. Papailiopoulos, and J. Ellenberg, "Approximate Gradient Coding via Sparse Random Graphs," 2017. [Online]. Available: http://arxiv.org/abs/1711.06771

[26] H. Wang, Z. Charles, and D. Papailiopoulos, "ErasureHead: Distributed Gradient Descent without Delays Using Approximate Gradient Coding," 2019. [Online]. Available: https://arxiv.org/abs/1901.09671

[27] S. Sarmasarkar, V. Lalitha, and N. Karamchandani, "On Gradient Coding with Partial Recovery," 2021. [Online]. Available: https://arxiv.org/abs/2102.10163

[28] M. R. Garey and D. S. Johnson, "'Strong' NP-Completeness Results: Motivation, Examples, and Implications," *Journal of the ACM (JACM)*, vol. 25, no. 3, pp. 499–508, 1978.

[29] N. Sbihi, "Algorithme de Recherche d'un Stable de Cardinalite Maximum dans un Graphe sans Étoile," *Discrete Mathematics*, vol. 29, no. 1, pp. 53–76, 1980.

[30] G. J. Minty, "On Maximal Independent Sets of Vertices in Claw-free Graphs," *Journal of Combinatorial Theory, Series B*, vol. 28, no. 3, pp. 284–304, 1980.

[31] D. Nakamura and A. Tamura, "A Revision Of Minty's Algorithm for Finding a Maximum Weight Stable Set Of A Claw-free Graph," *Journal of the Operations Research Society of Japan*, vol. 44, no. 2, pp. 194–204, 2001.

[32] Y. Faenza, G. Oriolo, and G. Stauffer, "An Algorithmic Decomposition of Claw-free Graphs Leading to an $O(n^3)$-Algorithm for the Weighted Stable Set Problem," in *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2011, pp. 630–646.

[33] "Anyscale - Ray Distributed Computing - Anyscale." [Online]. Available: https://www.anyscale.com/ray-open-source