On the Capacity of Privacy-Preserving and Straggler-Robust Distributed Coded Computing

Qicheng Zeng
Dept. Electronic Engineering, Tsinghua University
Email: zengqc19@mails.tsinghua.edu.cn

Sheng Zhou
Dept. Electronic Engineering, Tsinghua University
Email: sheng.zhou@tsinghua.edu.cn

2021 IEEE/CIC International Conference on Communications in China (ICCC) | 978-1-6654-4385-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICCC52777.2021.9580206

***Abstract*—Distributed computing can well exploit the computation resources in edge and cloud for many applications of large-scale machine learning, which also raises concerns on data privacy and straggling effect. A promising method to address these issues is using codes. In our work, we design a general computation framework that incorporates multi-stage computing tasks with multiple inputs and can be expressed as a *multi-variable arbitrary-degree* polynomial function f, with N distributed servers as workers, over a batch of data D that consists of data from different sources. We propose a privacy-preserving and straggler-robust coding scheme based on Lagrange polynomials, which can address up to S straggling workers *and* up to L colluding workers. We prove the optimality of the proposed scheme in terms of downlink communication efficiency, defined as the amount of bits of desired results versus that of the downloading results, and obtain an explicit expression of the capacity: $C = \frac{N-S-d(L-1)-1}{d(N-S)}$, which is the supremum of downlink communication efficiency over all feasible encoding schemes, and d is the degree of function f.**

I. Introduction

The ever-growing amount of data on various devices makes local processing difficult, especially in large-scale machine learning applications. A promising method is to disperse the data to nearby servers acting as workers and perform computation in parallel [1]. However, imperfect computing and networking conditions bring time uncertainties in the result feedback from workers, called *straggling effect*. To solve this issue, researchers have proposed to use codes [2] to speed up the process via adding redundancy of computation resources among workers. Meanwhile, how to guarantee the privacy and security of distributed computing is also a major concern, i.e., keeping the original data private against malicious servers.

Matrix multiplication is widely considered as one of the major applications in coded computing, especially for machine learning. Recently, Yu et al. [3] have proposed a coding strategy named *entangled polynomial code*, where original matrices are divided into submatrices of equal size to encode based on polynomials. Similarly, Aliasgari et al. [4] have proposed a secure generalized *PolyDot code* for matrix multiplication with additional security and privacy constraints.

In coded computing, the utilization efficiency of computation, storage and communication are key metrics. Chang et al. [5] consider maximizing downlink communication efficiency, the supremum of which is called *capacity*, by exploiting a naive polynomial coding strategy. Also, the privacy constraint on one part of the involving data matrices, i.e., A of matrix multiplication AB, is discussed in [5]. Similarly, Kakar et al. [6] focus on the maximum downlink communication rate and have proposed a coding strategy based on polynomials with privacy constraints on both sides of data matrices, i.e., A and B. Besides, a variety of recent related work has studied other aspects of distributed coded computing, such as heterogeneous servers [7], [8], algebraic structures of computation tasks [9], exploiting stragglers [10], combination with federated learning [11], [12], and computation resources allocation [13].

Coding schemes for matrix multiplication based on polynomials have been shown to have satisfying performance in terms of communication load and computation latency [3]–[6]. However, when computing tasks considered are more complicated or multi-stage, e.g., neural network training and inference, the above coding schemes [3]–[6] are not suitable. A more general computing framework are proposed in [14], which encapsulates matrix multiplication and gradient computation in machine learning. Nevertheless, the computing framework in [14] only considers the privacy and security constraints on only one part of the input data when reduced into matrix multiplication, and is not applicable to multi-variable computing functions.

To justify the motivation of considering multi-variable functions, we take neural network inference tasks as an illustrating example. The inputs include the sample data and model data that come from users and the data center respectively, and all needs to be private and secure. If we regard the computation of each layer in the neural network as one stage of an inference task, the entire task is a multi-stage computing task with multiple inputs, which can be expressed as a *multi-variable arbitrary degree* polynomial function.

In this paper, we design a general computation framework and propose a privacy-preserving and straggler-robust coding scheme based on Lagrange polynomials for the framework. In particular, our contributions include:

- We have found from the example of neural network inference that if matrix multiplication in single layer can be regarded as distributed computing tasks, we need perform many rounds of communication and computation to obtain final results. Therefore, we design a general computation framework to represent the end-to-end computing tasks as a *multi-variable arbitrary-degree* polynomial function and reduce the communication load

This work is sponsored in part by the National Key R&D Program of China No. 2018YFB1800804, by the Nature Science Foundation of China (No. 62022049, No. 61871254, No. 61861136003), by Open Research Fund Program of Beijing National Research Center for Information Science and Technology, and Hitachi Ltd.

of interactions between consecutive rounds.

- We prove that the privacy-preserving and straggler-robust coding scheme based on Lagrange polynomials can preserve the privacy of *all* input data instead of partial data. Additionally, we prove that the coding scheme is optimal in terms of downlink communication efficiency, defined as the amount of bits of desired results versus that of the downloading results, and obtain an explicit expression of the capacity which is the supremum of downlink communication efficiency over all feasible encoding schemes.

II. System Model and Problem Formulation

Compute a J-variable polynomial function $f := \mathbb{U} \to \mathbb{V}$ over a batch of data $D = (D_1, D_2, \ldots, D_k, \ldots, D_K)$, where $\mathbb{U}$ and $\mathbb{V}$ are multidimensional spaces over a sufficiently large field $\mathbb{F}$, and $D_k \in \mathbb{U}, k \geq 1$ are equal-size parts of the input data D divided in advance for distributed computing. Each D_k consists of J input variables that can come from J sources, i.e., $D_k = (D_{k,1}, \ldots, D_{k,j}, \ldots, D_{k,J}), D_{k,j} \in \mathbb{F}^{m_j \times m_j^{'}}$, which also means that the J variables do not need to be matrices of the same size. Obviously, the dimension of $\mathbb{U}$, denoted by U, satisfies that $U = \sum_{j=1}^{J} m_j m_j^{'}$.

For each round of distributed computing, the goal is to compute K results, i.e., $\{f(D_k)\}_{k=1}^{K}$. The degree of the polynomial function f is defined as the maximum order of its monomials, denoted by d. Here is a simple *6-variable 3-degree* polynomial function for example:

$$f(D_k) = AD_{k,4}D_{k,3}D_{k,1} + BD_{k,6}D_{k,5}D_{k,2} \tag{1}$$

where A and B are coefficient matrices. Without considering the existence of activation functions, the above polynomial function can be regarded as a comprehensive result of two neural network inference tasks, where $D_{k,1}, D_{k,2}$ serves as sample data from users and $D_{k,3}$ to $D_{k,5}$ serve as model data. Furthermore, after using polynomial functions to approximate nonlinear activation functions, the inference tasks can still be expressed as such multi-variable polynomial functions.

As Fig.1 shows, we assume that the computation is distributed to N workers due to limited energy and computation resources on users' devices. Considering the unreliable channels and the uncertain computation capabilities of workers, some of them may fail to send results back to the user in time, which are referred to as *stragglers*. Without loss of generality, we assume that the maximum number of stragglers that can be tolerated is S, and use the index set $\mathcal{S} = \{n_1, n_2, \ldots, n_{N-S}\} \subseteq [1:N], |\mathcal{S}| = N - S$ to denote the first $N - S$ workers that successfully return results.

We also assume that workers are honest but curious, i.e., they will send back the right computing results but there are at most L colluding workers which can communicate with each other and attempt to learn something about the data D. We call them *colluders*. The number of colluders is $L', L' \leq L$, and the user does not know which L' workers collude. We use the index set $\mathcal{L} = \{l_1, l_2, \ldots, l_{L'}\} \subseteq [1:N], |\mathcal{L}| = L'$ to denote the colluding workers.

In order to preserve the privacy of computation as well as combating stragglers, the data sent to workers needs to be encoded. The actual input data that worker n receive is

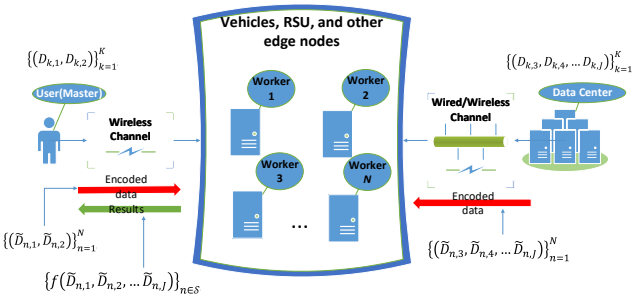

Fig. 1. Distributed Computing System Model.

denoted by $\widetilde{D}_n = g(D_1, D_2, \ldots, D_K, x_n), n \in [1:N]$, where g is the encoding function and x_n is a coding parameter used for worker n, and $\widetilde{D}_n$ has the same dimension as D_k. Consequently, workers should compute N results, i.e., $\{f(\widetilde{D}_n)\}_{n=1}^{N}$ instead of K original desired results, and return them to the user.

For the convenience of analysis, we denote the corresponding encoded data sent to workers in the subset $\mathcal{S}$ and $\mathcal{L}$ by $\widetilde{D}_{\mathcal{S}} := (\widetilde{D}_{n_1}, \widetilde{D}_{n_2}, \ldots, \widetilde{D}_{n_{N-S}})$ and $\widetilde{D}_{\mathcal{L}} := (\widetilde{D}_{l_1}, \widetilde{D}_{l_2}, \ldots, \widetilde{D}_{l_{L'}})$, respectively. Hence, the information-theoretic privacy-preserving constraint can be written as

$$I(D_1, D_2, \ldots, D_K; \widetilde{D}_{\mathcal{L}}) = 0, \tag{2}$$

where $I(\cdot)$ is the mutual information function.

To evaluate the performance of coding schemes in terms of using as few communication resources as possible, we define a key capacity metric called *downlink efficiency* [5]. It is the ratio of the bits of the desired result $f(D_k)$, to the communication load of downloading results from workers $f(\widetilde{D}_n)$:

$$R = \frac{\sum_{k=1}^{K} H(f(D_k))}{\sum_{n \in \mathcal{S}} H(f(\widetilde{D}_n))}. \tag{3}$$

Accordingly, capacity C is defined as the supremum of the downlink efficiency R over all feasible coding schemes that can address up to L colluders *and* up to S stragglers.

III. Main Result and Proof

In this section, we first provide a lower bound of the capacity and then obtain the converse through the *recovery threshold*, which is defined as the minimum number of workers required to recover desired computing results.

A. The lower bound of the capacity

Theorem 1. *For an (N, S, L, f) distributed computing problem, where N is the number of workers, S, L is the maximum number of stragglers and colluding workers that can be tolerated, respectively, and f is a multi-variable arbitrary-degree polynomial computing function, we have the following lower bound on the capacity C:*

$$C \geq \frac{N - S - d(L-1) - 1}{d(N-S)}. \tag{4}$$

where d is the degree of f.

We prove Theorem 1 by describing an achievable scheme that can guarantee the decodability and information-theoretic

privacy. By showing the downlink efficiency of the scheme $R = \frac{N-S-d(L-1)-1}{d(N-S)}$, we obtain a lower bound of the capacity.

The coding scheme is inspired by the Lagrange coding proposed in [14]. Note that target computing tasks are $\{f(D_k)\}_{k=1}^K$ and each D_k consists of J input variables, i.e., $D_k = (D_{k,1}, \ldots, D_{k,j}, \ldots, D_{k,J}), D_{k,j} \in \mathbb{F}^{m_j \times m_j^{'}}$. Input data are preprocessed using the following encoding function based on Lagrange polynomials:

$$g(\{D_{k,j}\}_{k=1}^K, x) := \sum_{k=1}^{K} D_{k,j} \cdot \prod_{i=1, i\neq k}^{K+L} \frac{x - b_i}{b_k - b_i} + \sum_{k=K+1}^{K+L} Z_{k,j} \cdot \prod_{i=1, i\neq k}^{K+L} \frac{x - b_i}{b_k - b_i}, \quad (5)$$

where $\{b_i\}_{i=1}^{K+L}$ are $(K+L)$ distinct elements chosen from $\mathbb{F}$, and $\{Z_{k,j}\}_{k=K+1}^{K+L}$ are uniformly random matrices chosen from $\mathbb{F}^{m_j \times m_j^{'}}$. Note that $D_{i,j} = g(\{D_{k,j}\}_{k=1}^K, b_i), i \in [1 : K]$ and

$$\begin{aligned} f(D_k) &= f(D_{k,1}, \ldots, D_{k,j}, \ldots, D_{k,J}) \\ &= f(g(\{D_{k,1}\}_{k=1}^K, b_k), \ldots, g(\{D_{k,J}\}_{k=1}^K, b_k)), \end{aligned} \quad (6)$$

due to the property of Lagrange polynomials.

We then select N distinct elements $\{a_n\}_{n=1}^N$ from $\mathbb{F}$ and ensure that $\{a_n\}_{n=1}^N \cap \{b_i\}_{i=1}^{K+L} = \varnothing$, and the input data sent to worker n is encoded as $\widetilde{D}_n = (\widetilde{D}_{n,1}, \ldots, \widetilde{D}_{n,j}, \ldots, \widetilde{D}_{n,J})$ and $\widetilde{D}_{n,j} = g(\{D_{k,j}\}_{k=1}^K, a_n)$. Hence the result returned by worker n is:

$$\begin{aligned} f(\widetilde{D}_n) &= f(\widetilde{D}_{n,1}, \ldots, \widetilde{D}_{n,j}, \ldots, \widetilde{D}_{n,J}) \\ &= f(g(\{D_{k,1}\}_{k=1}^K, a_n), \ldots, g(\{D_{k,J}\}_{k=1}^K, a_n)). \end{aligned} \quad (7)$$

The results received by the user are $\{f(\widetilde{D}_n)\}_{n\in\mathcal{S}}, |\mathcal{S}| = N - S$.

For the sake of illustration, we use $f(g(x))$ to denote $f(g(\{D_{k,1}\}_{k=1}^K, x), \ldots, g(\{D_{k,J}\}_{k=1}^K, x))$. It is easy to see that $f(g(x))$ is still a polynomial function about x because both f and g are polynomial functions, and the degree satisfies $\deg f(g(x)) \leq d \cdot (K + L - 1)$. There are at least $N - S$ workers that successfully return results, which also correspond to evaluation points of the polynomial function $f(g(x))$ that the user obtains. Consequently, if we choose a suitable size of input data batch D, i.e., $K = \frac{N-S-d(L-1)-1}{d}$, such that the following inequality can be satisfied:

$$N - S \geq \deg f(g(x)) + 1, \quad (8)$$

which means enough evaluation points of the polynomial function $f(g(x))$ are obtained by the user to recover polynomial function parameters. As a result, the desired result $f(D_k) = f(g(b_k)), k \in [1 : K]$ can be determined easily and the decodability is satisfied.

As for the downlink efficiency and capacity, since we can get the K desired items $\{f(D_k)\}_{k=1}^K$ out of the $N - S$ returned items, and the desired result $f(D_k)$ has the same data size (in bits) as the downloaded result $f(\widetilde{D}_n)$, the achievable downlink efficiency for this scheme can be written as

$$R = \frac{K}{N-S} = \frac{N - S - d(L-1) - 1}{d(N-S)}, \quad (9)$$

which serves as a lower bound of the capacity C.

Besides, to show the proposed scheme can preserve the privacy, we have to prove

$$I(D_1, D_2, \ldots, D_k; \widetilde{D}_{\mathcal{L}}) = 0.$$

To show this, we have:

$$\begin{aligned} &I(D_1, D_2, \ldots, D_K; \widetilde{D}_{\mathcal{L}}) = H(\widetilde{D}_{l_1}, \ldots, \widetilde{D}_{l_{L'}}) \\ &- H(\widetilde{D}_{l_1}, \ldots, \widetilde{D}_{l_{L'}} | D_1, D_2, \ldots, D_K), \\ &\overset{(a)}{=} H(\widetilde{D}_{l_1}, \ldots, \widetilde{D}_{l_{L'}}) - \sum_{j=1}^{J} H(Z_{K+1,j}, \ldots, Z_{K+L,j}), \\ &\overset{(b)}{=} H(\widetilde{D}_{l_1}, \ldots, \widetilde{D}_{l_{L'}}) - \sum_{j=1}^{J} L \cdot m_j m_j^{'} \cdot \log|\mathbb{F}|, \\ &\leq H(\widetilde{D}_{l_1}) + \cdots + H(\widetilde{D}_{l_{L'}}) - \sum_{j=1}^{J} L \cdot m_j m_j^{'} \cdot \log|\mathbb{F}|, \\ &\overset{(c)}{\leq} \sum_{j=1}^{J} L' \cdot m_j m_j^{'} \cdot \log|\mathbb{F}| - \sum_{j=1}^{J} L \cdot m_j m_j^{'} \cdot \log|\mathbb{F}| \\ &\leq 0 = 0. \end{aligned} \quad (10)$$

where (a) is due to the fact that all random matrices $\{Z_{k,j}\}_{k=K+1}^{K+L}$ are independent of the input data $\{D_k\}_{k=1}^K$, (b) is because that the entropy of each element in random matrices equals $\log|\mathbb{F}|$, and (c) follows that the upper bound of the entropy of each elements in $\widetilde{D}_{l_{(\cdot)}}$ equals to $\log|\mathbb{F}|$.

B. Converse

In this part, the upper bound of the capacity is derived from the following theorem.

Theorem 2. *For an (N, S, L, f) distributed computing problem, where N is the number of workers, S, L is the maximum number of stragglers and colluding workers that can be tolerated, respectively, and f is a multi-variable arbitrary-degree polynomial computing function, the size K of the input data batch $(D_1, , D_2, \ldots, D_k, \ldots, D_K)$ satisfies:*

$$K \leq \frac{N - S - d(L-1) - 1}{d}. \quad (11)$$

where d is the degree of f, and the encoding scheme is linear.

Consequently, we have an upper bound on the downlink efficiency of f:

$$R = \frac{K}{N-S} \leq \frac{N - S - d(L-1) - 1}{d(N-S)}. \quad (12)$$

And the capacity C has the same upper bound because it is the supremum of R.

Hence, with the lower bound in Theorem 1, we conclude that the capacity for an (N, S, L, f) distributed computing problem is:

$$C = \frac{N - S - d(L-1) - 1}{d(N-S)}. \quad (13)$$

The proof of Theorem 2 is presented in Section IV. As for the linearity of the encoding scheme, it means that the encoded data is a linear combination of original input data, i.e., $\widetilde{D}_n = \sum_{k=1}^K G_{n,k} D_k + \widetilde{Z}_n, n \in [1 : N]$, where $G \in \mathbb{F}^{N \times k}$ is defined as the encoding matrix corresponding to the aforementioned encoding function $g(x)$ and $\widetilde{Z}_n$ is the additive randomness.

IV. Proof of the upper bound of the capacity

In Section III-B, we found the key to the upper bound of the capacity is the maximum size K of the input data batch. Meanwhile, we note that maximizing the size of input is equivalent to minimizing the number of required workers, which means that we can obtain the maximum number K from the minimum number $(N-S)$ of workers that send back results successfully. This term is also known as the *recovery threshold*. Similar work has been done in [14], but in this paper we provide an alternative but more rigorous proof and extend to a general computation framework that incorporates multi-stage computing tasks with multiple inputs. First, The definition of recovery threshold is as follows:

Definition IV.1. *In the framework of distributed computing, the recovery threshold is the minimum number of workers that successfully return computed results, required to guarantee decodability.*

To simplify the problem further, we first add some constraints on the function f, and derive a weakened result under the condition of multilinearity. After that, in order to extend to the case of a general polynomial function, we give a construction of multilinear functions based on polynomial functions. The above two steps are explained in Section IV-A and IV-B, respectively. They enable the converse, and the proof is completed in Section IV-C.

A. Conclusion with multilinearity constraints

With the assumption of the multilinearity of f, we derive the *recovery threshold* in the following lemma.

Lemma 1. *For an (N, S, L, f) distributed computing problem, where N is the number of workers, S, L is the maximum number of stragglers and colluding workers that can be tolerated, respectively, where $f(X_1, X_2, \ldots, X_J)$ is a multivariable multilinear polynomial function. The degree of f is d, the size of the input data batch is K, and the recovery threshold is denoted by Rec. If the number of input variables J also equals to d, we have:*

$$Rec \geq d(K + L - 1) + 1, \tag{14}$$

when the encoding scheme is linear.

To prove Lemma 1, we first give the definition of a multilinear polynomial function f.

Definition IV.2. *For a multilinear polynomial function $f(X_1, \ldots, X_j, \ldots X_J)$, $X_1, \ldots, X_j, \ldots X_J$ are its J input variables, and f is linear with respect to each variables.*

Obviously, due to the multilinearity, the maximum order for each input variable X_j cannot be greater than 1, so the degree of f denoted by d satisfies $d \leq J$. The property also helps the construction in Section IV-B and the proof of Lemma 2.

With the above preparation, we consider such a multilinear function f discussed in Lemma 1. The task is to compute $\{f(D_k)\}_{k=1}^{K}$ over the input data batch $D = (D_1, D_2, \ldots, D_k, \ldots, D_K)$, and $D_k = (D_{k,1}, D_{k,2}, \ldots, D_{k,d})$ denotes the d input variables of f, meaning that the number of input variables equals to d.

Suppose $L = 0$, which means that there are no colluding workers, then what we need to prove is transformed into $Rec \geq d(K-1)+1$. In the end of the proof, it is found that the size of input data K can be replaced by $(K+L)$ without loss of generality. Here we use mathematical induction to prove $Rec \geq d(K-1)+1$.

1) First step: If $d = 1$, we need to prove $Rec \geq K$. As we mentioned previously, the encoding scheme can be expressed:

$$\widetilde{D}_n = \sum_{k=1}^{K} G_{n,k} D_k, n \in [1:N], \tag{15}$$

where N is the number of workers. The additive random matrices $\widetilde{Z}_n$ is ignored temporarily because we do not consider colluding workers at present. And $\{\widetilde{D}_{n_1}, \ldots, \widetilde{D}_{n_r}, \ldots, \widetilde{D}_{n_{Rec}}\}$ are the encoded data corresponding to the first Rec workers that are able to return results, which are formulated as follows:

$$\widetilde{D}_{n_r} = \sum_{k=1}^{K} G_{n_r,k} D_k, r \in [1:Rec]. \tag{16}$$

We prove $Rec \geq K$ by contradiction. Assuming $Rec < k$, the new encoding matrix corresponding to the Rec results $G' = \{G_{n_r,k}\}^{Rec \times k}$ cannot be column full rank due to $Rec < k$. As a result, there are non-zero elements in the null space of G', which means workers can receive the same encoded data $\{\widetilde{D}_{n_r}\}_{r=1}^{Rec}$ and the user can receive the same results $\{f(\widetilde{D}_{n_r})\}_{r=1}^{Rec}$ for different $\{D_k\}_{k=1}^{K}$, and this contradicts the decodability. Hence, we have proved the case for $d = 1$.

2) Second step: We assume the correctness of $Rec \geq d(K-1)+1$ for $d = d_0$, and consider a multilinear function $f(D_{k,1}, D_{k,2}, \ldots, D_{k,d_0}, D_{k,d_0+1}), k \in [1:K]$ with degree (d_0+1). In order to connect with the case for $d = d_0$, we construct a multilinear function f' with degree d_0, which satisfies the following mapping:

$$\begin{aligned} &f'(D_{k,1}, D_{k,2}, \ldots, D_{k,d_0}) \\ =&f(D_{k,1}, D_{k,2}, \ldots, D_{k,d_0}, \bar{X}_k), k \in [1:K]. \end{aligned} \tag{17}$$

Besides, we use Rec_f and $Rec_{f'}$ to denote the *recovery threshold* of f and f', which means that there exists a scheme that can guarantee the decodability of $f(D_{k,1}, D_{k,2}, \ldots, D_{k,d_0}, D_{k,d_0+1}), k \in [1;K]$ with the returned results of any Rec_f workers, and the same for f'.

We have $Rec_{f'} \geq d_0(K-1)+1$ because the degree of f' is d_0. If we can prove the decodability of $f'(D_{k,1}, D_{k,2}, \ldots, D_{k,d_0}), k \in [1:K]$ with the returned results of $(Rec_f - K)$ workers, we can deduce that $Rec_f - (K-1) \geq Rec_{f'}$ and complete the proof of the Second step in induction.

Recalling the linearity of the encoding scheme, we expand it into matrix form $\widetilde{D} = GD$ as follows for clearer illustration, and take the multiple input variables into account.

$$\begin{bmatrix} \widetilde{D}_{1,1} & \cdots & \widetilde{D}_{1,d_0+1} \\ \vdots & \ddots & \vdots \\ \widetilde{D}_{N,1} & \cdots & \widetilde{D}_{N,d_0+1} \end{bmatrix} = \begin{bmatrix} G_{1,1} & \cdots & G_{1,K} \\ \vdots & \ddots & \vdots \\ G_{N,1} & \cdots & G_{N,K} \end{bmatrix} \cdot \begin{bmatrix} D_{1,1} & \cdots & D_{1,d_0+1} \\ \vdots & \ddots & \vdots \\ D_{K,1} & \cdots & D_{K,d_0+1} \end{bmatrix}. \tag{18}$$

From the perspective of the matrix multiplication, each row of G corresponds to the encoded data that one worker receives. Due to the fact that G must be column full rank, we can choose K rows of G to construct a new matrix G_B, of which rows are a set of basis of $\mathbb{F}^K$. The corresponding workers of the rows are denoted by the set $\mathcal{K}, |\mathcal{K}| = K$.

Owing to the relationship between f and f' shown in (17), the decodability of $f'(D_{n,1}, D_{k,2}, \ldots, D_{k,d_0}), k \in [1 : K]$ is equivalent to the decodability of $f(D_{k,1}, D_{k,2}, \ldots, D_{k,d_0}, \bar{X}_k), k \in [1 : K]$. Consequently, the user can choose $\{\bar{X}_k\}_{k=1}^K$ according to the following equation:

$$\begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}^{K\times 1} = G_B^{K\times K} \cdot \begin{bmatrix} \bar{X}_1 \\ \vdots \\ \bar{X}_K \end{bmatrix}. \tag{19}$$

Since rows of G_B form a set of basis of $\mathbb{F}^K$, G_B is non-singular and $\{\bar{X}_k\}_{k=1}^K$ must be a zero vector. In fact, if there exist random matrices $\widetilde{Z}$ to combat colluding workers, $\{\bar{X}_k\}_{k=1}^K$ still exist and are unique due to the property of G_B. Accordingly, the (d_0+1)th variable of encoded data that workers in $\mathcal{K}$ receive is 0, and these workers will return constant 0 due to the multilinearity of f.

Hence, $f(D_{k,1}, D_{k,2}, \ldots, D_{k,d_0}, \bar{X}_k)$ is decodable with results from only $(Rec_f - k)$ workers, which also guarantees the decodability of $f'(D_{k,1}, D_{k,2}, \ldots, D_{k,d_0})$. The proof of the Second step is completed.

We have proved $Rec \geq d(K-1)+1$. Note that the purpose of adding random matrices $\widetilde{Z}$ is to combat up to L colluding workers. $\widetilde{Z}$ can be decomposed into a linear combination of random vectors of uniform size as follows:

$$\begin{aligned}\widetilde{D}_n &= \sum_{k=1}^{K} G_{n,k} D_k + \widetilde{Z}_n \\ &= \sum_{k=1}^{K} G_{n,k} D_k + \sum_{k=K+1}^{K+M} G_{n,k}^{(*)} Z_k, n \in [1:N]. \end{aligned} \tag{20}$$

Accordingly, we can take the M random matrices $\{Z_k\}_{k=K+1}^{K+M}$ as additional input data by adding new columns $\{G_{n,k}^{(*)}\}_{k=K+1}^{K+M}$ into G. Thus the encoding scheme can still be expressed as the form like $\widetilde{D} = G^* D^*$, where G^* and D^* are the new encoding matrices and the new input data, respectively. The size of D^* equals $(K+M)$. Therefore, replacing K with $(K+M)$ does not change the correctness of the original inequality.

Meanwhile, the M added random matrices should guarantee the privacy constraint, that is:

$$\begin{aligned} &I(D_1, D_2, \ldots, D_K; \widetilde{D}_{\mathcal{L}}) = H(\widetilde{D}_{l_1}, \ldots, \widetilde{D}_{l_{L'}}) \\ &- H(\widetilde{D}_{l_1}, \ldots, \widetilde{D}_{l_{L'}} | D_1, D_2, \ldots, D_K), \\ &\leq \sum_{j=1}^{J} L' \cdot m_j m_j^{'} \cdot \log|\mathbb{F}| - \sum_{j=1}^{J} M \cdot m_j m_j^{'} \cdot \log|\mathbb{F}| \\ &\leq 0 = 0. \end{aligned} \tag{21}$$

where the derivation is similar to that of (10). Since we have $L' \leq L$, the privacy constraint is satisfied if and only if $M \geq L$. Consequently, we conclude that $Rec \geq d(K+M-1)+1 \geq d(K+L-1)+1$ and complete the proof of Lemma 1.

B. The construction of multilinear functions

In Section IV-A, we have derived a weakened result with multilinearity constraints. In order to generalize to the case of polynomial functions, we give a construction method of multilinear functions as the following lemma:

Lemma 2. *For a general polynomial function f with degree d, f' is a d-variable multilinear polynomial function constructed based on f and satisfies:*

$$f'(D_1, D_2, \ldots, D_d) = \sum_{\mathcal{T}\subseteq[1:d]} [(-1)^{|\mathcal{T}|} f(\sum_{k\in\mathcal{T}} D_k)], \tag{22}$$

where f' is linear with respect to each input variable, $\mathcal{T}$ is a subset of the set $[1:d]$ and the degree of f' also equals to d.

In order to prove Lemma 2, we have to prove the order of each variable in f' is at most 1 due to the multilinearity of f'. Therefore, if the coefficients of higher-order terms in the multilinear polynomial function f' equals 0, the proof is completed.

For any $j \in [1:d]$, we use $h(D_j)$ to denote a general higher-order term in f'. In $h(D_j)$, the order of D_j is greater than 1. And $h(D_j)$ consists of $\{D_j, D_{j_1}, D_{j_2}, \ldots, D_{j_m}\}$ through multiplication.

Obviously, the number of subset $\mathcal{T}$ that includes $\{j, j_1, j_2, \ldots, j_m\}$ is $2^{(d-m-1)}$, and the constant coefficients of $h(D_j)$ are the same for these different $\mathcal{T}$ in the calculation result of $f(\sum_{k\in\mathcal{T}} D_k)$. We only need to consider the impact of $(-1)^{|\mathcal{T}|}$.

Note that for $i \in [0 : d-m-1]$, there are $\begin{pmatrix} i \\ d-m-1 \end{pmatrix}$ subsets $\mathcal{T}$ that meet above conditions and include extra i variables except for $\{j, j_1, j_2, \ldots, j_m\}$. Consequently, any coefficient of $h(D_j)$ denoted by Coeff_j can be obtained as follows:

$$\begin{aligned} \mathrm{Coeff}_j &= \sum_{\{j,j_1,j_2,\ldots,j_m\}\subseteq\mathcal{T}, \mathcal{T}\subseteq[1:d]} (-1)^{|\mathcal{T}|}, \\ &= \sum_{i=0}^{d-m-1} \begin{pmatrix} i \\ d-m-1 \end{pmatrix} (-1)^{m+1+i}, \\ &= (-1)^{m+1} \cdot \sum_{i=0}^{d-m-1} \left[\begin{pmatrix} i \\ d-m-1 \end{pmatrix} 1^{d-m-1-i} (-1)^i \right], \\ &= (-1)^{m+1} \cdot (1-1)^{d-m-1} = 0, \end{aligned} \tag{23}$$

which completes the proof of Lemma 2.

C. Proof of Theorem 2

To prove Theorem 2 based on above two lemmas, we consider a general polynomial function f with degree d. The minimum number of workers required to recover the K desired results is denoted by $Rec(f, K, L)$, where L is the maximum number of colluding workers that can be tolerated.

Then, we construct a multilinear function f' according to Lemma 2 as follows:

$$f'(D_1, D_2, \ldots, D_d) = \sum_{\mathcal{T}\subseteq[1:d]} [(-1)^{|\mathcal{T}|} f(\sum_{k\in\mathcal{T}} D_k)], \tag{24}$$

and similarly, $Rec(f', K, L)$ denotes the recovery threshold for f'.

Note that f' is a linear combination of functions $f(\sum_{n\in\mathcal{S}} D_n)$ and has the same degree as f. Therefore, given the linearity of the encoding scheme, any scheme for f can be directly applied to f'. In fact, the only constraint for f is that f must be a polynomial function, which also holds for f'. Hence, the k desired results of f' can be recovered with $Rec(f, K, L)$ workers by using the same computation scheme with f, which means $Rec(f, K, L) \geq Rec(f', K, L)$.

According to Lemma 1, we have $Rec(f', K, L) \geq d(K + L-1)+1$ because f' is a multilinear function with degree d. Besides, the actual number of results returned by workers is $(N - S)$, which must be greater than the recovery threshold. Consequently, we have

$$N - S \geq Rec(f, K, L) \geq Rec(f', K, L) \geq d(K + L - 1) + 1.$$

As a result, $K \leq \frac{N-S-d(L-1)-1}{d}$ is derived, which completes the proof of Theorem 2.

V. Conclusion

We have studied the private distributed computing problem and designed a general computation framework that incorporates multi-stage computing tasks with multiple inputs, which can be expressed as a *multi-input arbitrary-degree* polynomial function. Then we propose a privacy-preserving and straggler-robustness coding scheme based on Lagrange polynomial coding, and prove its optimality in terms of downlink efficiency. Furthermore, we prove the capacity $C = \frac{N-S-d(L-1)-1}{d(N-S)}$, where C is the supremum of downlink communication efficiency over all feasible encoding schemes. Given the general computation framework, combining coded computing with federated learning and task offloading are our future work.

References

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, et al., "Tensorflow: A system for large-scale machine learning," *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, 2016.

[2] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, K. Ramchandran, "Speeding up distributed machine learning using codes," *IEEE Transactions on Information Theory*, vol. 64, no. 3, pp. 1514-1529, March 2018.

[3] Q. Yu, M. A. Maddah-Ali and A. S. Avestimehr, "Straggler Mitigation in Distributed Matrix Multiplication: Fundamental Limits and Optimal Coding," *IEEE Transactions on Information Theory*, vol. 66, no. 3, pp. 1920-1933, March 2020.

[4] Malihe Aliasgari, Osvaldo Simeone, and Joerg Kliewer, "Private and secure distributed matrix multiplication with flexible communication load," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 2722-2734, 2020.

[5] Wei-Ting Chang, and Ravi Tandon, "On the capacity of secure distributed matrix multiplication," *2018 IEEE Global Communications Conference (GLOBECOM)*, Abu Dhabi, United Arab Emirates, 2018, pp. 1-6.

[6] J. Kakar, S. Ebadifar and A. Sezgin, "On the Capacity and Straggler-Robustness of Distributed Secure Matrix Multiplication," *IEEE Access*, vol. 7, pp. 45783-45799, 2019.

[7] A. Reisizadeh, S. Prakash, R. Pedarsani and A. S. Avestimehr, "Coded Computation Over Heterogeneous Clusters," *IEEE Transactions on Information Theory*, vol. 65, no. 7, pp. 4227-4242, July 2019.

[8] N. Woolsey, R. R.Chen, M. Ji, "A new combinatorial coded design for heterogeneous distributed computing," *IEEE Transactions on Communications"*, 2021.

[9] T. Jahani-Nezhad, M. A. Maddah-Ali, "Berrut Approximated Coded Computing: Straggler Resistance Beyond Polynomial Computing," *arXiv preprint arXiv:2009.08327*, 2020.

[10] B. Hasircioglu, J. Gomez-Vilardebo, D. Gunduz, "Bivariate polynomial coding for exploiting stragglers in heterogeneous coded computing systems," *arXiv preprint arXiv:2001.07227*, 2020.

[11] S. Prakash, S. Dhakal, M. Akdeniz, Y. Yona, S. Talwar, S. Avestimehr, N. Himayat, "Coded Computing for Low-Latency Federated Learning Over Wireless Edge Networks," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 1, pp. 233-250, Jan. 2021.

[12] S. Dhakal, S. Prakash, Y. Yona, S. Talwar and N. Himayat, "Coded Federated Learning," *2019 IEEE Globecom Workshops (GC Wkshps)*, Waikoloa, HI, USA, 2019.

[13] Y. Yang, M. Interlandi, P. Grover, S. Kar, S. Amizadeh, M. Weimer, "Coded elastic computing," *2019 IEEE International Symposium on Information Theory (ISIT)*, Paris, France, 2019.

[14] Q. Yu, S. Li, N. Raviv, S. M. M. Kalan, M. Soltanolkotabi, A. S. Avestimehr, "Lagrange Coded Computing: Optimal Design for Resiliency, Security, and Privacy," *The 22nd International Conference on Artificial Intelligence and Statistics*, April 2019.