Non-Linearly Separable Distributed Computing: A Sparse Tensor Factorization Approach

Ali Khalesi
IPSA & LINCS Lab, Paris, France
Email: ali.khalesi@ipsa.fr

Ahmad Tanha, Derya Malak, and Petros Elia
EURECOM, Biot, France
Email: {tanha, malak, elia}@eurecom.fr

***Abstract*—The work considers the N-server distributed computing setting with K users requesting functions that are arbitrary multi-variable polynomial evaluations of L real (potentially non-linear) basis subfunctions. Our aim is to seek efficient task-allocation and data-communication techniques that reduce computation and communication costs. Towards this, we take a tensor-theoretic approach, in which we represent the requested non-linearly decomposable functions using a properly designed tensor $\bar{\mathcal{F}}$, whose sparse decomposition into a tensor $\bar{\mathcal{E}}$ and matrix $\mathbf{D}$ directly defines the task assignment, connectivity, and communication patterns. We here design an achievable scheme, employing novel fixed-support SVD-based tensor factorization methods and careful multi-dimensional tiling of subtensors, yielding computation and communication protocols whose costs are derived here, and which are shown to perform substantially better than the state of art.**

I. Introduction

As computational workloads continue to grow in size and complexity, distributed computing has become essential [1] in handling these workloads, particularly now with the emergence of massive machine learning systems. While frameworks such as MapReduce [2] and Spark [3] enable large-scale parallel processing across distributed servers to support a wide range of computation tasks (see e.g., [4]–[21]), it is indeed the case that limited computation and communication resources impose a fundamental bottleneck that governs the design of distributed computing schemes.

Motivated by these intertwined bottlenecks and by the increasing complexity of the computed functions, we study the problem of distributed computation of *non-linearly separable functions*, in multi-user computing systems with bounded computational and communication resources. In particular, we consider N distributed servers with limited computational and communication capabilities, and K users, each requesting the evaluation of their own polynomial of L real (potentially non-linear) basis subfunctions. In this setting, a coordinator assigns computation tasks across the servers, aiming for lossless recovery of the user demands, while also minimizing the required number of servers N, or equivalently maximizing the *achievable system rate* K/N. For this novel setting, we seek to provide novel task allocation and data communication schemes, and present bounds on the achievable system rate.

This research was partially supported by European Research Council ERC-StG Project SENSIBILITÉ under Grant 101077361, by the Huawei France-Funded Chair Toward Future Wireless Networks, and by the Program "PEPR Networks of the Future" of France 2030.

Before detailing our setting, consider this simple example that illustrates the key constraints and metrics involved.

Example 1. *Let us first focus on the nature of the functions. Consider a setting with $L = 4$ basis subfunctions*

$$f_1(\cdot) \triangleq e^{x_1},\ f_2(\cdot) \triangleq \log(x_2),\ f_3(\cdot) \triangleq \sqrt{x_2},\ f_4(\cdot) \triangleq \cos(x_3)$$

which operate on fixed input vectors $x_1, x_2, x_3 \in \mathbb{R}^B$, each resulting[1] in the corresponding output files $\{W_\ell = f_\ell(\cdot)\}_{\ell\in[4]}$. Consider a requested function

$$F_1(\mathbf{W}) = 7\,W_1^2W_2^3 + 8\,W_1W_3^2W_4 + 6\,W_3W_4^4 + 4\,W_1^4W_4^2$$

which is clearly non-linearly separable over the variables in $\mathbf{W} = (W_1, W_2, W_3, W_4)$. As we will see, our key parameters will be the number of basis subfunctions L (here $L = 4$), as well as the maximum allowable degree *P_ℓ of each subfunction output $W_\ell,\ \ell = 1, \cdots, L$. In our example, the maximum degrees (exponent) of each output file W_1, W_2, W_3, W_4 cannot exceed $P_1 = 4, P_2 = 3, P_3 = 2, P_4 = 4$, so for instance, the maximum degree for W_3 is $P_3 = 2$, found in the second additive term of the requested function. Let us assume a second requested function — now by a second user — of the form*

$$F_2(\mathbf{W}) = 3\,W_2W_3^3 + 2\,W_1^3W_3 + 11\,W_1^2W_2 + 13\,W_2^2W_4^3\,.$$

We note that the declared maximum degrees $4, 3, 2, 4$ — which constrain every requested function — are not respected, as this second polynomial entails maximum degrees $3, 2, 3, 3$.

Assuming $K = 2$ users, respectively requesting $F_1(\mathbf{W})$ and $F_2(\mathbf{W})$, let us also assume $N = 3$ servers, and a network topology (see Figure 1) that allows each server to communicate with up to $\Delta \leq K$ users (let us assume in this example that $\Delta = 1$), where each user must linearly combine the received signals to retrieve their own function. Another important parameter is $\Gamma \leq L$ – the computational constraint on the number of subfunctions that can be computed at each server. If we assumed $\Gamma = 2$, we would not be able to compute $F_1(\mathbf{W})$ because its second term $8\,W_1W_3^2W_4$ could never be reproduced. We must have $\Gamma = 3$. There will finally be another computational constraint, on the ranges of powers $\Lambda_\ell,\ \ell = 1, \cdots, L$ of each output variable W_ℓ that can be computed at any one server. This is explained later on.

The challenge here will be to assign subfunctions and specific powers of these subfunctions for specific servers to

[1] Naturally here, operations are component-wise.

compute, and to determine the linear encoding and decoding processes at the servers and users, respectively. This same problem, it is worth noting, could conceivably be resolved by "linearization", converting our problem to the linearly-separable setting found[2] in [19]. Part of our contribution here is to present a more powerful approach which involves converting the above problem to a sparse tensor decomposition problem. This new approach can have (as detailed in the journal version of this work [22]) very substantial gains over the linearized approach.

A. Related Works

There have been substantial efforts to attain the fundamental limits of distributed computing of linearly separable classes of functions, including those on gradient coding [7]–[9], [11], matrix multiplication [15]–[18], polynomial computing [19], [23]–[26]. Among existing models, the proposed setting is closest to that considered in [19], where the authors devised an equivalence between the distributed computing problem for linearly separable functions subject to sparsity constraints on the factor supports, and a longstanding open problem in sparse matrix factorization, for which they devised tessellation-based constructions accounting for support constraints.

Our emphasis on non-linearly separable functions is motivated by emerging machine learning applications [12], [13], most notably large language models, which require distributed computation of non-linear functions. Such functions can include activation functions [27] and attention mechanisms [28] that involve higher-order terms of underlying basis subfunctions and their multiplications [29].

B. Our Contributions

In this work, we study the problem of lossless distributed computing of non-linearly separable functions. Our contributions are summarized as follows.

- **Tensor representation of user demands:** We utilize the inherent structure of user demands to represent them using a full-rank tensor $\bar{\mathcal{F}}$. By embedding the range of exponents of a set of basis subfunctions into higher-order tensor modes, the proposed framework captures a broad class of non-linearly separable functions.
- **Sparse tensor factorization framework:** We introduce the sparse factorization of $\bar{\mathcal{F}}$ as $\bar{\mathcal{F}} = \bar{\mathcal{E}} \times_1 \mathbf{D}$ (the operation $\times_1$ is detailed in Section IV). The encoding tensor $\bar{\mathcal{E}}$ specifies, for each server n, a subset of basis subfunctions and a subset of power terms to be computed by that server, while the decoding matrix $\mathbf{D}$ specifies a user set of size at most Δ to which each server transmits. Naturally the sparsity of $\bar{\mathcal{E}}$ and $\mathbf{D}$ respectively specify the computation and communication costs.
- **A new achievable scheme design:** Focusing on lossless tensor factorization, we decompose $\bar{\mathcal{F}}$ into properly sized and carefully positioned subtensors, which are then factorized into subtensors of $\bar{\mathcal{E}}$ and $\mathbf{D}$, and lower bound the system rate, defined as K/N (Theorem 1), by leveraging novel tools from fixed-support sparse matrix factorization and multilinear singular value decomposition (SVD) [30], [31]. Our approach yields an exponential reduction in the required number of servers for lossless reconstruction of user demands compared to the sparse matrix factorization solution [19], as demonstrated in Example 2.

[2]For our above example, this would entail $L' = 8$ subfunctions, where the first subfunction would be $W_1^2 W_2^3$, the second $W_1 W_3^2 W_4$, and so on.

Notations. For $n \in \mathbb{Z}^+$, we let $[n] \triangleq \{1, \ldots, n\}$. For a, $b \in \mathbb{Z}^+$ such that $a < b$, $[[a : b]]$ is an ordered set of integers, ranging from a to b, and $a \mid b$ denotes a divides b. For any matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$, then $\mathbf{X}(i,j)$, $\mathbf{X}(i,:)$, and $\mathbf{X}(:,j)$ represent its (i,j)-th entry, the i-th row, and j-th column, respectively, for $i \in [m]$, $j \in [n]$. $[\mathbf{A}, \mathbf{B}]$ indicates the horizontal concatenation of the two matrices. All the above matrix notations are extended to tensors. $\mathbb{1}(.)$ is the indicator function. For a vector $\mathbf{x} \in \mathbb{R}^N$, $\|\mathbf{x}\|_0$ denotes the number of non-zero elements, and $\|\mathbf{x}\|$ the Euclidean norm.

II. System Model

We consider a practical distributed computing framework with N servers and K users, where each user requests the evaluation of an arbitrary multivariate polynomial function. Each of these polynomial functions are composed of L real-valued, potentially non-linear basis subfunctions $\{f_\ell(\cdot)\}_{\ell \in [L]}$. A coordinator node orchestrates the distributed computation of these functions through three phases, described next.

a) Demand Phase: During the initial demand phase, each user $k \in [K]$ independently requests the computed output of a single real-valued function that takes the multivariate form

$$F_k(W_1, \ldots, W_L) = \sum_{\underline{\mathbf{p}} \in \prod_{\ell \in [L]} [P_\ell]} c_{k,\underline{\mathbf{p}}} \prod_{\ell \in [L]} W_\ell^{p_\ell} \tag{1}$$

where $\{W_\ell \triangleq f_\ell(x)\}_{\ell \in [L]}$ is the resulting real-valued *output file*, representing evaluations of $\{f_\ell(\cdot)\}_{\ell \in [L]}$ on a common fixed input vector $x \in \mathbb{R}^d$, where x denotes the underlying data instance (e.g., a feature vector or signal) on which all computations are performed. Furthermore, $c_{k,\underline{\mathbf{p}}} \in \mathbb{R}$ denotes the *basis coefficient* corresponding to the monomial $\prod_{\ell \in [L]} W_\ell^{p_\ell}$, where $\underline{\mathbf{p}} \triangleq (p_1, \ldots, p_L)$ is the exponent vector. We set $W'(\underline{\mathbf{p}}) = \prod_{\ell \in [L]} W_\ell^{p_\ell}$, so that we can transform (1) into

$$F_k = \sum_{\underline{\mathbf{p}} \in \prod_{\ell \in [L]} [P_\ell]} c_{k,\underline{\mathbf{p}}} \, W'(\underline{\mathbf{p}}) \tag{2}$$

which is the classical multi-user gradient coding problem. On the other hand, in this work, we explicitly account for multiplication costs in the high-dimensional problem (1), which introduces new algebraic challenges, as we will detail below.

b) Non-Linear Computing Phase: Subsequently, the coordinator assigns to each server n, a set of basis subfunctions $\mathcal{S}_n \subseteq [L]$ to be computed locally. Each server n then computes $\{W_\ell = f_\ell(.)\}_{\ell \in \mathcal{S}_n}$. We consider the computation cost

$$\Gamma \triangleq \max_{n \in [N]} |\mathcal{S}_n| \tag{3}$$

representing the maximum number of basis subfunctions to be locally computed at any server. Each server operates under

a computation constraint $|\mathcal{S}_n| \leq \Gamma \leq L$, which implies that up to Γ basis subfunctions can appear in each demand, i.e., $c_{k,\underline{\mathbf{p}}} = 0$ in (1) when the support of $\underline{\mathbf{p}}$ involves more than Γ components, reflecting the per-server computation limit.

The coordinator also assigns to each server $n \in [N]$ the corresponding set of exponent vectors $\mathcal{P}_n \subseteq \prod_{\ell\in[L]}[P_\ell]$. Each server n then performs multiplications to compute the power terms $\{W_\ell^{p_\ell}\}_{\ell\in\mathcal{S}_n}$, and consequently, to obtain the corresponding monomial $\prod_{\ell\in\mathcal{S}_n} W_\ell^{p_\ell}$ for each $\underline{\mathbf{p}} \in \mathcal{P}_n$, thus rendering the computation phase non-linear.

To devise the multiplication cost, we assume that for any given W_ℓ, $\ell \in \mathcal{S}_n$, each server n computes the following ordered set of exponents with a cardinality[3] Λ_ℓ, $\ell \in \mathcal{S}_n$:

$$[[q\Lambda_\ell + 1 : (q+1)\Lambda_\ell]] \ , \quad q \in \mathbb{N} \ . \tag{4}$$

Alternatively, each server computes a power term W_ℓ^α in a demanded function, where the exponent α is decomposed as

$$\alpha \triangleq q\Lambda_\ell + r \ , \quad q \in \mathbb{N} \ , \quad r \in [[0 : \Lambda_\ell - 1]] \ . \tag{5}$$

Using the server's allowed range in (4), we decompose $W_\ell^\alpha = W_\ell^{q\Lambda_\ell+1} W_\ell^{(r-1)}$, and hence the cost of evaluating W_ℓ^α is

$$\lfloor \log_2(q\Lambda_\ell + 1) \rfloor + (r-1) \tag{6}$$

where the logarithmic cost is due to repeated squaring, i.e., computing successive powers of two until reaching the desired anchor exponent, which is negligible compared to the $(r-1)$ multiplications required within the range of exponents. Hence, the overall complexity induced by self-multiplications of basis function $f_\ell(\cdot)$ per server is in order $\mathcal{O}(\Lambda_\ell)$, versus $\mathcal{O}(\alpha)$ for naive repeated multiplications. For example, with $\alpha = 851$ and $\Lambda_\ell = 100$, from (6), the computation requires only 59 multiplications, versus $\alpha - 1 = 850$ multiplications for the naive approach. In general, $\Gamma \leq L \ll \Lambda_\ell$ allows us to conclude that the cost of evaluating the monomial $\prod_{\ell\in\mathcal{S}_n} W_\ell^{p_\ell}$ from $\{W_\ell^{p_\ell}\}_{\ell\in\mathcal{S}_n}$ for $\underline{\mathbf{p}} \in \mathcal{P}_n$ at server n is negligible compared to the cost of evaluating $\{W_\ell^{p_\ell}\}_{\ell\in\mathcal{S}_n}$. Thus, we consider the multiplication cost at each server

$$\Lambda_\ell \triangleq \max_{\underline{\mathbf{p}}\in\mathcal{P}_n} p_\ell - \min_{\underline{\mathbf{p}}\in\mathcal{P}_n} p_\ell + 1 \ , \quad \ell \in \mathcal{S}_n \tag{7}$$

representing the range of exponents of $\{W_\ell\}_{\ell\in\mathcal{S}_n}$ that must be computed, which in turn determines the maximum number of multiplications required locally, at each server, among all multiplicative terms in (1). Each server also operates under a multiplication constraint $\Lambda_\ell \leq P_\ell$ for each $\ell \in [L]$.

c) Communication Phase: Upon completing the local computations, server $n \in [N]$ forms signals

$$z_n \triangleq \sum\nolimits_{\underline{\mathbf{p}}\in\mathcal{P}_n} e_{n,\underline{\mathbf{p}}} \prod\nolimits_{\ell\in[L]} W_\ell^{p_\ell}, \quad n \in [N] \tag{8}$$

as dictated by the encoding coefficients $e_{n,\underline{\mathbf{p}}} \in \mathbb{R}, n \in [N]$. Note that $p_\ell = 0$, $\ell \notin \mathcal{S}_n$, reflects the absence of certain basis subfunctions and their powers in the multiplicative terms for computation by server n. Subsequently, server n proceeds to transmit z_n to a subset of users $\mathcal{T}_n \subseteq [K]$, via an error-free shared link. Finally, during the decoding part of the last phase, each user $k \in [K]$ linearly combines its received signals to get

$$F'_k \triangleq \sum\nolimits_{n\in[N]} d_{k,n} z_n \tag{9}$$

dictated by the decoding coefficients $d_{k,n} \in \mathbb{R}, n \in [N]$, where $d_{k,n} = 0$, $k \notin \mathcal{T}_n$. We consider the communication cost

$$\Delta \triangleq \max_{n\in[N]} |\mathcal{T}_n| \tag{10}$$

denoting the maximum number of users that each server can communicate to, where $\Delta \leq K$.

We highlight that the constraints Γ, Δ, and Λ_ℓ are strict, meaning that they must be satisfied for every instance of the problem. Thus, the costs from (3), (10), and (7) yield a system with normalized constraints

$$\gamma = \frac{\Gamma}{L} \ , \quad \delta = \frac{\Delta}{K} \ , \quad \left\{\lambda_\ell = \frac{\Lambda_\ell}{P_\ell}\right\}_{\ell\in[L]} \tag{11}$$

where $\gamma, \delta, \lambda_\ell \in [0,1]$, $\ell \in [L]$. Accordingly, for each server n, we must specify the basis subfunctions $\mathcal{S}_n$, the exponent vectors $\mathcal{P}_n$, and the users $\mathcal{T}_n$. Having to serve many users with fewer servers naturally places a burden on the system, bringing to the fore the concept of the *system rate*

$$R \triangleq \frac{K}{N} \ . \tag{12}$$

In a system parametrized by $(K, N, L, \Gamma, \Delta, \{P_\ell, \Lambda_\ell\}_{\ell\in[L]})$, depicted in Figure 1, our task is to devise an achievable scheme for the error-free recovery of any set of desired functions, subject to constraints on computation, communication, and multiplication loads. For general values of $\{\Lambda_\ell, P_\ell\}_{\ell\in[L]}$ and for $\Gamma \geq 1$, a possible way to solve this problem is to exploit [19], and embed (1) into the linearly-separable form in (2) with the corresponding output files $W'(\underline{\mathbf{p}})$ for each $\underline{\mathbf{p}}$, which requires $L' \triangleq \prod_{\ell\in[L]} P_\ell$ basis subfunctions to embed all multiplicative terms imposed by (1). The required

$$N = \frac{K}{\Delta}\frac{L'}{\Gamma}\min(\Delta, \Gamma) \tag{13}$$

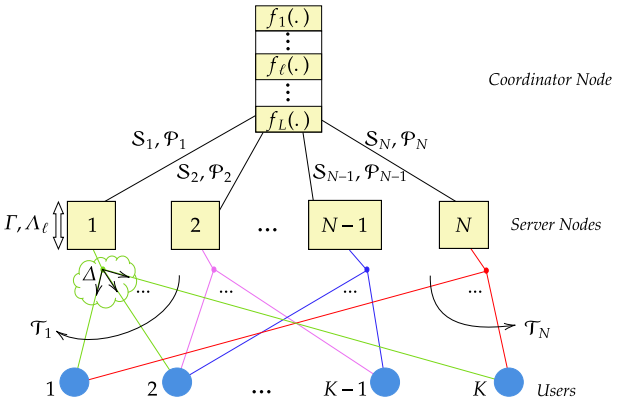

Fig. 1. The lossless $(K, N, L, \Gamma, \Delta, \{P_\ell, \Lambda_\ell\}_{\ell\in[L]})$ distributed computing setting with a coordinator node, N servers, and K users.

[3]With heterogeneous server computational abilities, the cardinality would instead depend on both ℓ and n.

from [19, Theorem 1], directly implies that in this linearized alternative, the required N can grow exponentially with L.

We proceed to present our achievable scheme, yielding an upper bound on the number of required servers N.

III. Main Result

We next present the achievable rate result for the proposed lossless $(K, N, L, \Gamma, \Delta, \{P_\ell, \Lambda_\ell\}_{\ell\in[L]})$ distributed computing setting, by upper bounding the required number of servers N, which is the common dimension between $\bar{\mathcal{E}}$, $\mathbf{D}$, directly impacting our sparse tensor factorization approach. The result holds without any restriction on the dimensions, provided that each subtensor of $\bar{\mathcal{F}}$ has full rank, a condition that is easily justified in our real-valued function settings.

Theorem 1. *The achievable rate of the lossless $(K, N, L, \Gamma, \Delta, \{P_\ell, \Lambda_\ell\}_{\ell\in[L]})$ distributed computing system, under $P_\ell = P, \Lambda_\ell = \Lambda$ for all $\ell \in [L]$ and $(\Delta|K,\ \Lambda|P)$, takes the form $R = K/N$, where*

$$N \leq \frac{K}{\Delta}\binom{L}{\Gamma}\min\big(\Delta, \Lambda^\Gamma\big)\big(\frac{P}{\Lambda}\big)^\Gamma . \tag{14}$$

Sketch of proof: Due to a lack of space (partly because our proofs require a sizable set of definitions), we present the full proof in the journal version of this work (cf. [22]), which describes the achievability of the corresponding decomposition $\bar{\mathcal{F}} = \bar{\mathcal{E}} \times_1 \mathbf{D}$, and how this is translated into our distributed computing setting while meeting the constraints determined by $(\Gamma, \Delta, \{\Lambda_\ell\}_{\ell\in[L]})$.

We next present a proof sketch, which captures the key insights. Our achievable scheme consists of three main steps. In the first step, the coordinator divides $\bar{\mathcal{F}}$ into properly sized hypercubic subtensors (corresponding to 'tiles') where the tiles must cover all elements of $\bar{\mathcal{F}}$, and maintain a width limit Δ and modal constraint $\{\Lambda_\ell\}_{\ell\in[L]}$ for Γ modes of L modes (cf. Definition 2 in [22]). In the second step, the coordinator performs a multilinear SVD for each tile of $\bar{\mathcal{F}}$, which defines the way each tile is the mode-1 product of a subtensor by a submatrix (the so-called right and left factors of the multilinear SVD, as detailed in [22]). Finally, in the third step, the right and left factors are carefully positioned to form the desired $\bar{\mathcal{E}}$ and $\mathbf{D}$, which in turn, define the computation and communication protocols of the distributed computing problem, respectively. To elaborate on N achieved by the proposed scheme, we consider a mode-1 matrix unfolding of tensor $\bar{\mathcal{F}}$ and utilize its rank properties.

IV. Problem Formulation in Tensor Form, and Achievability

In the $(K, N, L, \Gamma, \Delta, \{P_\ell, \Lambda_\ell\}_{\ell\in[L]})$ framework, the desired functions in (1) are fully represented by a tensor $\bar{\mathcal{F}} \in \mathbb{R}^{K\times P_1\times\ldots\times P_L}$ of all function coefficients $\{c_{k,\underline{\mathbf{p}}}\}$ across all the users, which is an order-$(L+1)$ tensor, i.e., a multi-way array with $L+1$ modes. With $\bar{\mathcal{F}}$ in place, we must decide on the computation assignment (encoding) and the communication protocol (decoding). For the error-free case, from [31], this task is equivalent — directly from (8),(9)— to solving a sparse tensor factorization problem subject to the sparsity constraints Γ, $\{\Lambda_\ell\}_{\ell=1}^L$, and Δ, as specified in (3), (7), and (10), respectively. This problem takes the form

$$\bar{\mathcal{F}} = \bar{\mathcal{E}} \times_1 \mathbf{D} \tag{15}$$

where $\bar{\mathcal{E}} \in \mathbb{R}^{N\times P_1\times\ldots\times P_L}$ is the computing tensor, capturing the non-linear encoding tasks of servers that holds the coefficients $e_{n,\underline{\mathbf{p}}}$ from (8), $\mathbf{D} = (d_{k,n}) \in \mathbb{R}^{K\times N}$ is the communication matrix derived from (9), capturing the communication and linear decoding task done by each user, and finally, the operation $\times_1$ denotes the mode-1 product[4] of $\bar{\mathcal{E}}$ and $\mathbf{D}$ and is comprised of three consecutive operations:

$$\bar{\mathcal{E}} \to \bar{\mathcal{E}}_{(1)}\,,\ \bar{\mathcal{F}}_{(1)} = \mathbf{D}\bar{\mathcal{E}}_{(1)}\,,\ \bar{\mathcal{F}}_{(1)} \to \bar{\mathcal{F}} \tag{16}$$

where the matrix $\bar{\mathcal{E}}_{(1)} \in \mathbb{R}^{N\times P_1P_2\ldots P_L}$ represents the mode-1 unfolding of the tensor $\bar{\mathcal{E}} \in \mathbb{R}^{N\times P_1\times\ldots\times P_L}$.

To motivate the underlying idea behind the tensor construction, we next form tensors of desired function outputs, transmissions, and retrieved function outputs, as detailed below.

a) Desired Function Outputs in Vector Form: We exploit the representation in (1), and consider the following vector of desired function outputs

$$\mathbf{f} \triangleq [F_1, F_2, \ldots, F_K]^\intercal \in \mathbb{R}^K \tag{17}$$

and for all $\underline{\mathbf{p}} \in \prod_{\ell\in[L]}[P_\ell]$, the tensor of basis coefficients

$$\bar{\mathcal{F}}_k(\underline{\mathbf{p}}) \triangleq c_{k,\underline{\mathbf{p}}}\,,\ \bar{\mathcal{F}}_k \in \mathbb{R}^{P_1\times P_2\times\ldots\times P_L},\ k \in [K] \tag{18}$$

as well as the tensor of multiplicative terms of $\{W_\ell\}_{\ell\in[L]}$

$$\bar{\mathcal{W}}(\underline{\mathbf{p}}) \triangleq \prod\nolimits_{\ell\in[L]} W_\ell^{p_\ell},\ \bar{\mathcal{W}} \in \mathbb{R}^{P_1\times P_2\times\ldots\times P_L}\ . \tag{19}$$

Next, using (17)-(19) leads us to the vector of function outputs

$$\mathbf{f} = \mathit{stack}_1(\bar{\mathcal{F}}_1, \ldots, \bar{\mathcal{F}}_K) \times_{[[2:L+1]]}^{[[1:L]]} \bar{\mathcal{W}} \tag{20}$$

where, $\mathit{stack}_1(\bar{\mathcal{F}}_1, \ldots, \bar{\mathcal{F}}_K)$ forms an order-$(L+1)$ tensor $\bar{\mathcal{F}} \in \mathbb{R}^{K\times P_1\times\ldots\times P_L}$ composed by stacking the K tensors $\{\bar{\mathcal{F}}_k\}_{k\in[K]}$ (each of order L) along a new first mode. In the above, the operation $\times_{[[2:L+1]]}^{[[1:L]]}$ simply generalizes the tensor contraction product[5] (cf. [31]) by capturing an ordered set of common modes between two given tensors $\bar{\mathcal{F}} \in \mathbb{R}^{K\times P_1\times P_2\cdots\times P_L}$ and $\bar{\mathcal{W}} \in \mathbb{R}^{P_1\times P_2\cdots\times P_L}$. This generalized tensor contraction product yields a lower order tensor. Particularly, the $\times_{[[2:L+1]]}^{[[1:L]]}$-contraction product of $\bar{\mathcal{F}}$ and $\bar{\mathcal{W}}$ is obtained by contracting modes $[[2 : L+1]]$ in $\bar{\mathcal{F}}$ and $[[1 : L]]$ in $\bar{\mathcal{W}}$, corresponding to the indices $\underline{\mathbf{p}} \in \prod_{\ell\in[L]}[P_\ell]$, yielding an order-1 tensor (i.e., a vector) $\mathbf{f} \in \mathbb{R}^K$ with entries

$$F_k = \sum_{\underline{\mathbf{p}}\in\prod_{\ell\in[L]}[P_\ell]} \bar{\mathcal{F}}(k, \underline{\mathbf{p}})\,\bar{\mathcal{W}}(\underline{\mathbf{p}}), \quad k \in [K]. \tag{21}$$

b) Transmissions in Vector Form: In the communication phase, similarly to above, from (8), the transmission from

[4]The mode-n product is defined similarly to (16) with respect to the mode-n unfolding of the tensor (cf. Definition 2 in [22]).

[5]Tensor contraction product operation extends matrix multiplication to higher-order tensors by summing over shared modes.

server n takes the following form $z_n = \bar{\mathcal{E}}_n \times_{[[1:L]]}^{[[1:L]]} \bar{\mathcal{W}} \in \mathbb{R}$, thus yielding the overall transmission vector

$$\mathbf{z} \triangleq [z_1, z_2, \ldots, z_N] = \bar{\mathcal{E}} \times_{[[2:L+1]]}^{[[1:L]]} \bar{\mathcal{W}}, \quad \mathbf{z} \in \mathbb{R}^N \tag{22}$$

where for all $\underline{\mathbf{p}} \in \prod_{\ell\in[L]}[P_\ell]$, the encoding coefficients $e_{n,\underline{\mathbf{p}}}$ from (8) satisfy

$$\bar{\mathcal{E}}_n(\underline{\mathbf{p}}) \triangleq e_{n,\underline{\mathbf{p}}}, \ \bar{\mathcal{E}}_n \in \mathbb{R}^{P_1\times P_2\times\ldots\times P_L}. \tag{23}$$

c) Retrieved Function Outputs in Vector Form: In the decoding phase, from (9), each retrieved function output takes the form $F'_k = \mathbf{d}_k^\intercal \mathbf{z} \in \mathbb{R}$, thus resulting in the vector of all outputs taking the form

$$\mathbf{f}' \triangleq [F'_1, F'_2, \ldots, F'_K] = [\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_K]^\intercal \mathbf{z} \in \mathbb{R}^K \tag{24}$$

where the decoding coefficients $d_{k,n}$ from (9) satisfy

$$\mathbf{d}_k \triangleq [d_{k,1}, d_{k,2}, \ldots, d_{k,N}]^\intercal \in \mathbb{R}^N, \ k \in [K]. \tag{25}$$

The tensors of basis coefficients in (18), the tensors of encoding coefficients (23), and the vector of decoding coefficients in (25) allow us to form the respective tensors

$$\bar{\mathcal{F}} \triangleq \mathit{stack}_1(\bar{\mathcal{F}}_1, \ldots, \bar{\mathcal{F}}_K) \in \mathbb{R}^{K\times P_1\times P_2\times\ldots\times P_L}, \tag{26}$$

$$\bar{\mathcal{E}} \triangleq \mathit{stack}_1(\bar{\mathcal{E}}_1, \ldots, \bar{\mathcal{E}}_N) \in \mathbb{R}^{N\times P_1\times P_2\times\ldots\times P_L}, \tag{27}$$

$$\mathbf{D} \triangleq [\mathbf{d}_1, \ldots, \mathbf{d}_K]^\intercal \in \mathbb{R}^{K\times N}. \tag{28}$$

We are now ready to establish the connection between the tensor forms in (26), (27), and (28) and the tensor factorization problem. We note that employing the definitions of $\mathbf{f}$ and f' from (20) and (24), respectively, and setting the recovery error to zero, i.e., $\|\mathbf{f}'-\mathbf{f}\|^2 = 0$, we see that resolving our distributed computing problem requires that $\bar{\mathcal{F}}$ be decomposed as (15).

In terms of the corresponding connection to the sparsity of $\mathbf{D}$ and $\bar{\mathcal{E}}$, we recall from (3), our metric Γ, which directly from (23) and (27), implies the computation constraint as

$$\max_{n\in[N]} \sum_{\ell\in[L]} \Big| \mathbb{1}\Big(\mathrm{supp}\big(\bar{\mathcal{E}}(n, \underbrace{:, \ldots, :}_{\ell-1 \text{ terms}}, p_\ell, \underbrace{:, \ldots, :}_{L-\ell \text{ terms}}) \big) \neq \emptyset \Big) \Big| \leq \Gamma$$

where $p_\ell \in [P_\ell]$. Furthermore, from (7), we recall Δ, which from (25) and (28), implies a communication constraint

$$\max_{n\in[N]} \big|\mathrm{supp}\big(\mathbf{D}(:,n)\big)\big| \leq \Delta.$$

Finally from (10), we recall Λ_ℓ, which from (20)–(27), suggests, for all $\ell \in [L]$, $p_\ell \in [P_\ell]$, the multiplication constraints

$$\|\bar{\mathcal{E}}(n, p_1, p_2, \ldots, p_{\ell-1}, :, p_{\ell+1}, \ldots, p_L)\|_0 \leq \Lambda_\ell.$$

A. Example of Tensor Formulation and Achievability

We next present a basic example and compare our scheme with the matrix factorization approach in [19].

Example 2. *Consider the* $(K = 4, N, L = 2, \Gamma = 2, \Delta = 2, \{P_\ell = 4, \Lambda_\ell = 2\}_{\ell\in[L]})$ *setting, with* N *servers tasked with computing non-linear functions of* $L = 2$ *basis subfunctions*

$$F_k(.) = \sum_{(p_1,p_2)\in[4]\times[4]} c_{k,\underline{\mathbf{p}}}\, W_1^{p_1} W_2^{p_2}, \quad k \in [4]. \tag{29}$$

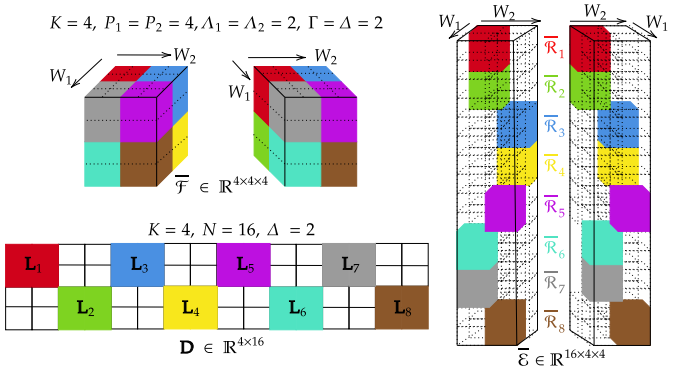

Fig. 2. Corresponding to Example 2, this figure illustrates the partitioning of $\bar{\mathcal{F}}$ into 8 tiles of size $(2\times2\times2)$, and the sparse tiling of $\mathbf{D}$ and $\bar{\mathcal{E}}$ with tiles $\mathbf{L}_j$ and $\bar{\mathcal{R}}_j$, respectively, resulting in the full tiling of $\bar{\mathcal{F}} = \bar{\mathcal{E}} \times_1 \mathbf{D}$.

The coefficients $c_{k,\underline{\mathbf{p}}}$ *are described by tensor* $\bar{\mathcal{F}}$*, as demonstrated in Figure 2. Given the design constraints, we aim to guarantee lossless reconstruction of (29), and we directly conclude from* (14) *that we need* $N \leq 16$ *servers. To tackle this challenge of reconstruction, we need to construct*

1) *The* $(N \times P_1 \times P_2) = (16 \times 4 \times 4)$ *computing tensor* $\bar{\mathcal{E}}$*, specifying the computational tasks of each server.*
2) *The* $(K \times N) = (4 \times 16)$ *communication matrix* $\mathbf{D}$*, determining the server-user connections.*

These originate from the decomposition of $(K \times P_1 \times P_2) = (4 \times 4 \times 4)$ *tensor* $\bar{\mathcal{F}}$ *(cf.* (26)*) as* $\bar{\mathcal{F}} = \bar{\mathcal{E}} \times_1 \mathbf{D}$*, representing the requested functions. The solution is then as follows.*

1) *Initially, we partition* $\bar{\mathcal{F}}$ *into* $\frac{K}{\Delta}\frac{P_1}{\Lambda_1}\frac{P_2}{\Lambda_2} = 2\cdot2\cdot2 = 8$ *disjoint subtensors* $\bar{\mathcal{S}}_j \in \mathbb{R}^{\Delta\times\Lambda_1\times\Lambda_2} = \mathbb{R}^{2\times2\times2}$, $j \in [8]$.
2) *Next, using the standard tensor decomposition form (cf. [22]), we decompose each* $\bar{\mathcal{S}}_j$ *as* $\bar{\mathcal{S}}_j = \bar{\mathcal{R}}_j \times_1 \mathbf{L}_j$*, where* $\bar{\mathcal{R}}_j \in \mathbb{R}^{2\times2\times2}, \mathbf{L}_j \in \mathbb{R}^{2\times2}$ *for all* $j \in [8]$*, noting that such full decomposition is feasible since the maximum rank of each* $\bar{\mathcal{S}}_j$ *is* $\min(\Delta, \Lambda_1\Lambda_2) = 2$.
3) *Finally, we construct* $\mathbf{D} \in \mathbb{R}^{4\times16}$ *and* $\bar{\mathcal{E}} \in \mathbb{R}^{16\times4\times4}$ *by tiling them with* $\mathbf{L}_j$ *and* $\bar{\mathcal{R}}_j$*, respectively, as in Figure 2.*

Example 2 provides a glimpse of the general principle behind creating our scheme. In brief, corresponding to (14), we begin by splitting our $K \times P_1 \times P_2$ tensor $\bar{\mathcal{F}}$ into $\frac{K}{\Delta}\frac{P_1}{\Lambda_1}\frac{P_2}{\Lambda_2}$ subtensors of size $\Delta \times \Lambda_1 \times \Lambda_2$. We decompose these subtensors into submatrices $\{\mathbf{L}_j\}_{j\in[8]}$ and into subtensors $\{\bar{\mathcal{R}}_j\}_{j\in[8]}$ that form tiles of $\mathbf{D}$ and $\bar{\mathcal{E}}$, respectively. The tile placement must respect the sparsity constraints $(\Gamma, \Delta, \Lambda_1, \Lambda_2)$ and must yield $\bar{\mathcal{E}} \times_1 \mathbf{D} = \bar{\mathcal{F}}$. Regarding the required number of servers, the general rule is that N is the number of subtensors, multiplied by the rank of each subtensor. Since we had 8 subtensors, each of rank 2, we need $N = 16$ servers. In contrast, it is easy to show that the linearized approach — employing sparse matrix factorization — in [19], would entail $L_M = \prod_{\ell\in[L]} P_\ell = 16$ basis subfunctions and approximately double the number of servers, for the same Γ and Δ.

REFERENCES

[1] J. Verbraeken, M. Wolting, J. Katzy, J. Kloppenburg, T. Verbelen, and J. S. Rellermeyer, "A survey on distributed machine learning," *ACM Comput. Surv.*, vol. 53, no. 2, pp. 1–33, 2020.

[2] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[3] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets," in *2nd USENIX Wksh. Hot Topics Cloud Comput. (HotCloud)*, Sydney, Australia, Aug. 2010.

[4] D. Malak, "Distributed structured matrix multiplication," in *Proc., IEEE Int. Symp. Inf. Theory (ISIT)*, 2024, pp. 2550–2555.

[5] F. Brunero, K. Wan, G. Caire, and P. Elia, "Coded distributed computing for sparse functions with structured support," in *Proc., Inf. Theory Wksh. (ITW)*, Saint-Malo, France, Apr. 2023, pp. 474–479.

[6] J. So, R. E. Ali, B. Güler, J. Jiao, and A. S. Avestimehr, "Securing secure aggregation: Mitigating multi-round privacy leakage in federated learning," in *Proc. AAAI Conf. Artif. Intell.*, vol. 37, Washington, DC, USA, Feb. 2023, pp. 9864–9873.

[7] R. Tandon, Q. Lei, A. G. Dimakis, and N. Karampatziakis, "Gradient coding: Avoiding stragglers in distributed learning," in *Proc., Int. Conf. Mach. Learn. (ICML)*. Sydney, Australia: PMLR, Aug. 2017, pp. 3368–3376.

[8] N. Raviv, I. Tamo, R. Tandon, and A. G. Dimakis, "Gradient coding from cyclic MDS codes and expander graphs," *IEEE Trans. Inf. Theory*, vol. 66, no. 12, pp. 7475–7489, 2020.

[9] M. Ye and E. Abbe, "Communication-computation efficient gradient coding," in *Proc., Int. Conf. Mach. Learn. (ICML)*. Stockholm, Sweden: PMLR, Jul. 2018, pp. 5610–5619.

[10] S. Wang, J. Liu, N. Shroff, and P. Yang, "Fundamental limits of coded linear transform," *arXiv preprint arXiv:1804.09791*, 2018.

[11] N. Charalambides, H. Mahdavifar, and A. O. Hero, "Generalized fractional repetition codes for binary coded computations," *IEEE Trans. Inf. Theory*, vol. 71, no. 3, pp. 2170–2194, 2025.

[12] S. Vithana and S. Ulukus, "Private read-update-write with controllable information leakage for storage-efficient federated learning with top-r sparsification," *IEEE Trans. Inf. Theory*, vol. 70, no. 5, pp. 3669–3692, 2023.

[13] M. Kavian, R. Chor, M. Sefidgaran, and A. Zaidi, "Heterogeneity matters even more in distributed learning: Study from generalization perspective," *arXiv preprint arXiv:2503.01598*, 2025.

[14] Q. Yan, S. Yang, and M. Wigger, "Storage-computation-communication tradeoff in distributed computing: Fundamental limits and complexity," *IEEE Trans. Inf. Theory*, vol. 68, no. 8, pp. 5496–5512, 2022.

[15] S. Dutta, V. Cadambe, and P. Grover, "Short-dot: Computing large linear transforms distributedly using coded short dot products," *Proc., Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 29, Dec. 2016.

[16] O. Ordentlich and Y. Polyanskiy, "Optimal quantization for matrix multiplication," *arXiv preprint arXiv:2410.13780*, 2025.

[17] A. Ramamoorthy, L. Tang, and P. O. Vontobel, "Universally decodable matrices for distributed matrix-vector multiplication," in *Proc., IEEE Int. Symp. Inf. Theory (ISIT)*, Paris, France, Jul. 2019, pp. 1777–1781.

[18] A. Tanha, M. R. D. Salehi, M. Dutta, and D. Malak, "Cooperative coded matrix multiplication in secrecy-constrained vehicular networks," in *IEEE 103rd Veh. Technol. Conf. (VTC)*, 2026, to appear.

[19] A. Khalesi and P. Elia, "Tessellated distributed computing," *IEEE Trans. Inf. Theory*, vol. 71, no. 6, pp. 4754–4784, 2025.

[20] J. Maheri and P. Elia, "Constructing hamiltonian decompositions of complete k-uniform hypergraphs," in *Proc., IEEE Int. Symp. Inf. Theory (ISIT)*, Ann Arbor, MI, USA, Jun. 2025, pp. 1–6.

[21] J. Maheri, K. K. K. Namboodiri, and P. Elia, "Universal and asymptotically optimal data and task allocation in distributed computing," *arXiv preprint 2601.05873*, 2026.

[22] A. Khalesi, A. Tanha, D. Malak, and P. Elia, "Non-linearly separable distributed computing: A sparse tensor factorization approach," *arXiv preprint*, Jan. 2026.

[23] K. Wan, H. Sun, M. Ji, and G. Caire, "Distributed linearly separable computation," *IEEE Trans. Inf. Theory*, vol. 68, no. 2, pp. 1259–1278, 2022.

[24] A. Tanha and D. Malak, "The influence of placement on transmission in distributed computing of Boolean functions," in *Proc., IEEE Int. Wksh. Signal Process. Adv. Wireless Commun. (SPAWC)*, Lucca, Italy, Sep. 2024.

[25] A. Khalesi, A. Tanha, D. Malak, and P. Elia, "Tessellated distributed computing of non-linearly separable functions," Recent Results, Wksh. Distrib. Comput. Optim. Learn (WDCL), Munich, Germany, Sep. 2025.

[26] E. Peter, K. Karakkad, D. Malak, and P. Elia, "New achievability schemes for distributed computing of linearly separable functions," in *Intl. Zurich Semin. Inf. Commun. (IZS)*, Zurich, Switzerland, Feb. 2026.

[27] D. Haziza, B. Steiner, E. Yang, M. Sirotenko, B. Jacob, and A. Vaswani, "2:4 sparsity for activation functions in LLMs," in *Proc., Int. Conf. Learn. Represent. (ICLR)*, Singapore, May 2025.

[28] A. Hassani, F. Zhou, A. Kane, J. Huang, C.-Y. Chen, M. Shi, S. Walton, M. Hoehnerbach, V. Thakkar, M. Isaev, Q. Zhang, B. Xu, H. Wu, W. mei Hwu, M.-Y. Liu, and H. Shi, "Generalized neighborhood attention: Multi-dimensional sparse attention at the speed of light," *arXiv preprint arXiv:2504.16922*, 2025.

[29] S. M. Jayakumar, W. M. Czarnecki, J. Menick, J. Schwarz, J. Rae, S. Osindero, Y. W. Teh, T. Harley, and R. Pascanu, "Multiplicative interactions and where to find them," in *Proc., Int. Conf. Learn. Represent. (ICLR)*, Addis Ababa, Ethiopia, Apr. 2020.

[30] Q.-T. Le, E. Riccietti, and R. Gribonval, "Spurious valleys, np-hardness, and tractability of sparse matrix factorization with fixed support," *SIAM J. Matrix Anal. Appl.*, vol. 44, no. 2, pp. 503–529, 2023.

[31] L. De Lathauwer, B. De Moor, and J. Vandewalle, "A multilinear singular value decomposition," *SIAM J. Matrix Anal. Appl.*, vol. 21, no. 4, pp. 1253–1278, 2000.