Lossless Tessellated Distributed Computing

Ali Khalesi and Petros Elia

***Abstract*—The work considers the N-server distributed computing scenario with K users requesting functions that are linearly-decomposable over an arbitrary basis of L real (potentially non-linear) subfunctions, and the aim is to receive the function outputs with zero error, reduced computing cost (γ; the fraction of subfunctions each server must compute), and reduced communication cost (δ; the fraction of users each server must connect to). For a matrix $F \in \mathbb{R}^{K\times L}$ representing the linearly-decomposable form of the K requested functions, our problem is made equivalent to the open problem of zero-error sparse matrix factorization that seeks $F = DE$ over a special subset of γ-sparse and δ-sparse $E \in \mathbb{R}^{N\times L}$ and $D \in \mathbb{R}^{K\times N}$ matrices that respectively define which servers compute each subfunction, and which users connect to each server. We here design an achievable scheme designing $\mathbf{E}, \mathbf{D}$ by utilizing a fixed-support SVD-based matrix factorization method that first splits F into properly sized and carefully positioned submatrices, and then decomposes these into properly designed submatrices of $\mathbf{D}$ and $\mathbf{E}$. For the zero-error case and under basic dimensionality and single-shot assumptions, this work reveals that the optimal number of servers can be upper-bounded as $N_{\text{opt}} \leq \min(\Delta,\Gamma)\lfloor K/\Delta\rfloor\lfloor L/\Gamma\rfloor + \min(\mathbf{mod}(K,\Delta),\Gamma)\lfloor L/\Gamma\rfloor + \min(\mathbf{mod}(L,\Gamma),\Delta)\lfloor K/\Delta\rfloor + \min(\mathbf{mod}(K,\Delta),\mathbf{mod}(L,\Gamma))$. In the special case, it reveals that the maximum possible ratio K/N or the zero-error capacity of this system for any feasible single-shot scheme satisfying $\delta^{-1},\gamma^{-1} \in \mathbb{N}$ is $C = \max(K/L\delta, \gamma)$.**

***Keywords*—Distributed Computing, Sparse Matrix Factorization, Low-rank Matrix Approximation, Tessellation, Matrix Decomposition, Random Matrix Theory, Distributed Parallel Computing, Capacity, Bulk Synchronous Parallel, MapReduce**

I. Introduction

A. Multi-User Linearly-Decomposable Distributed Computing

We focus on the very broad and arguably practical setting of *multi-user, multi-server distributed computation of linearly-decomposable real functions*, which nicely captures several classes of computing problems that include distributed gradient coding problems [1]–[3], the distributed linear-transform computation problem [4], [5], the distributed matrix multiplication or the distributed multivariate polynomial computation problems [6], [7], as well as the distributed computing problem of training large-scale machine learning algorithms and deep neural networks with massive data [8]. These constitute a broad collection of problems where both computation and communication costs are crucial [9], [10].

This work was supported by the European Research Council (ERC) through the EU Horizon 2020 Research and Innovation Program under Grant 725929 (Project DUALITY), ERC-PoC project LIGHT (Grant 101101031), as well as by the Huawei France-funded Chair towards Future Wireless Networks. The authors are with the Communication Systems Department at EURECOM, 450 Route des Chappes, 06410 Sophia Antipolis, France (email: khalesi@eurecom.fr; elia@eurecom.fr).

Our setting, as is depicted in Fig. 1, initially considers a *master node* that coordinates, in three phases, a set of N distributed *servers* that compute functions requested by the K *users*. During the initial *demand* phase, each user $k \in \{1, 2, \ldots, K\}$ independently requests the computed output of a single real function $F_k(.)$. Under the real-valued linear decomposability assumption[1], these functions take the basic form

$$F_k(.) = \sum_{\ell=1}^{L} f_{k,\ell} f_\ell(.) = \sum_{\ell=1}^{L} f_{k,\ell} W_\ell \tag{1}$$

where $f_\ell(\cdot)$ denotes a *(basis or component) subfunction*, where $f_{k,\ell}$ denotes a real-valued combining coefficient, and where $W_\ell = f_\ell(x), x \in \mathcal{D}$ denotes the real-valued *output file* of $f_\ell(\cdot)$ for an input x from any domain set $\mathcal{D}$.

Subsequently, during the *computing phase*, the master assigns to each server $n \in [N]$, a set of subfunctions $\mathcal{S}_n \subseteq [L]$ to compute locally[2] in order to generate the corresponding W_ℓ, and then during the *communication phase*, server n forms signals

$$z_{n,t} \triangleq \sum_{\ell\in[L]} e_{n,\ell,t} W_\ell, \;\; n \in [N],\; t \in [T] \tag{2}$$

as dictated by the *encoding coefficients* $e_{n,\ell,t} \in \mathbb{R}, n \in [N], t \in [T], \ell \in [L]$, and proceeds to transmit $z_{n,t}$ during time-slot $t = 1, 2, \ldots, T$ to a subset of users $\mathcal{T}_{n,t} \subseteq [K]$, via a dedicated error-free broadcast channel. Finally, during the decoding part of the last phase, each user k linearly combines its received signals to get

$$F'_k \triangleq \sum_{n\in[N],t\in[T]} d_{k,n,t} z_{n,t} \tag{3}$$

as dictated by the *decoding coefficients* $d_{k,n,t} \in \mathbb{R}, n \in [N], t \in [T], k \in [K]$. Naturally $d_{k,n,t} = 0, \forall k \notin \mathcal{T}_{n,t}$ simply because user $k \in [K]$ does not receive any symbol from server $n \in [N]$ during time $t \in [T]$. Note that both encoding and decoding coefficients are determined by the master node after the demand phase, and are independent of the instance of the input to the requested functions.

[1]This nicely captures linearly separable functions (see for example [11]) where each $F_k(.)$, taking L subfunction as input, can be written as a linear combination of L *univariate* subfunctions. In our work, these subfunctions need not be univariate. Furthermore, the real-valued exposition entails a variety of additional advantages over finite-field approaches [11]–[14], such as accuracy advantages stemming from using real-valued fixed point data representations, as well as advantages regarding computation overflows, quantization errors and scalability barriers [15]–[17].

[2]We will interchangeably use $\mathcal{S}_n$ to describe sets of indices (of subfunctions), as well as the subfunctions themselves.

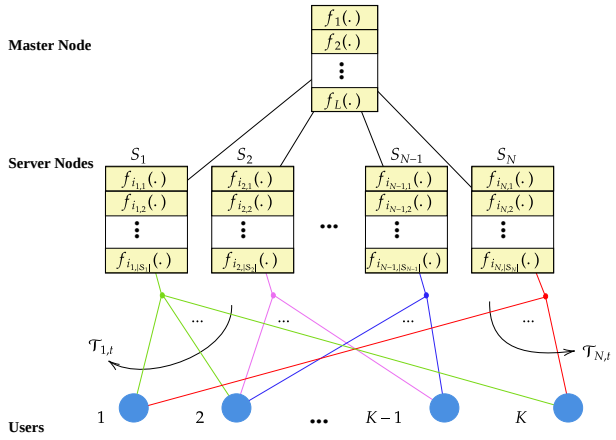

Fig. 1. The K-user, N-server, T-shot setting. Each server n computes the subfunctions in $\mathcal{S}_n = \{f_{i_{n,1}}(.), f_{i_{n,2}}(.), \ldots, f_{i_{n,|\mathcal{S}_n|}}(.)\}$ and communicates to users in $\mathcal{T}_{n,t}$, under computational constraint $|\mathcal{S}_n| \leq \Gamma \leq L$ and communication constraint $|\mathcal{T}_n| \leq \Delta \leq K$, yielding a system with normalized constraints $\gamma = \frac{\Gamma}{L}, \delta = \frac{\Delta}{K}$ where $\gamma, \delta \in [0, 1]$.

In case of error, we consider

$$\mathcal{E} = \sum_{k=1}^{K} |F'_k - F_k|^2, \ \mathcal{E} \in \mathbb{R}, \quad \forall k \in [K] \tag{4}$$

to be the Euclidean distortion during function retrieval. For $\mathcal{T}_n = \cup_{t=1}^{T} \mathcal{T}_{n,t}$, we consider the computation and communication costs

$$\Gamma \triangleq \max_{n\in[N]} |\mathcal{S}_n|, \quad \Delta \triangleq \max_{n\in[N]} |\mathcal{T}_n| \tag{5}$$

respectively representing the maximum number of subfunctions to be locally computed at any server[1], and the number of users that a server can communicate to. After normalization, we here consider the normalized costs

$$\gamma \triangleq \frac{\Gamma}{L} \qquad \delta \triangleq \frac{\Delta}{K}. \tag{6}$$

The three parameters are bounded[2] between 0 and 1.

Another two parameters of interest are

$$\zeta \triangleq \frac{\Delta}{L} \tag{7}$$

where ζ normalizes the number of activated communication links by the number of subfunctions.

In a system defined by K, N and L, our goal is to find schemes that can recover any set of desired functions without error, with the smallest possible computation and communication loads γ, δ. To do so, we must carefully decide which subfunctions each server computes, and which combinations of computed outputs each server sends to which users. Having to serve many users with fewer servers naturally places a burden on the system (suggesting higher γ, δ), bringing to the fore the concept of the *system rate*

$$R \triangleq \frac{K}{N} \tag{8}$$

and the corresponding *system capacity* C representing the supremum of all rates.

B. Connection to sparse matrix factorization, and related works

Toward analysing our distributed computing problem, we can see from (1) that the desired functions are fully represented by a matrix $\mathbf{F} \in \mathbb{R}^{K\times L}$ of the aforementioned coefficients $f_{k,\ell}$. With $\mathbf{F}$ in place, we must decide on the computation-assignment and communication (encoding and decoding) protocol. As we have seen in [14], [18], for the error-free case, this task is equivalent — directly from (2),(3) and (4) — to solving a (sparse) matrix factorization problem of the form

$$\mathbf{DE} = \mathbf{F} \tag{9}$$

where, as we will specify later on, the $NT \times L$ *computing-and-encoding matrix* $\mathbf{E}$ holds the coefficients $e_{n,\ell,t}$ from (2), while the $K \times NT$ *communication matrix* $\mathbf{D}$ holds the decoding coefficients $d_{k,n,t}$ from (3). Focusing on functions over finite fields, the work in [14], after making the connection between distributed computing and the above factorization problem, employed from coding theory the class of covering codes and a new class of *partial covering codes*, in order to derive bounds on the optimal communication and computation costs for the *error-free case*. In brief, by choosing $\mathbf{D}$ to be the sparse parity-check matrix of a (shown to exist) sparse partial covering code, each column of $\mathbf{E}$ was subsequently produced to be the coset leader from syndrome decoding (with the syndrome being) the corresponding column[3] of $\mathbf{F}$. This allowed for reduced communication and computation costs, where for example in the single shot scenario with a q-ary finite field, the (slightly different from here) normalized computation cost $\gamma \in (0, 1]$ was bounded as a function of the q-ary entropy function H_q to be in the range $\gamma \in [H_q^{-1}(\frac{\log_q(L)}{N}), H_q^{-1}(\frac{K}{N})]$.

A first exposition of the *real-valued* variant of our computing problem, again for the error-free case can be found in [19], which reformulated the equivalent sparse matrix factorization problem $\mathbf{DE} = \mathbf{F}$ into the well-known compressed sensing problem $\mathbf{Ax} = \mathbf{y}$ which seeks to efficiently identify unique sparse solutions to an under-determined system of equations[4]. This reformulation allowed for conditional bounds on γ of

[1]Our focus on the cost of computing the component subfunctions $f_\ell(\cdot)$ stems from the point of view that these functions typically capture computationally intensive (and generally non-linear) tasks which would dominate, in terms of load, the remaining easier linear manipulations at the servers and users during encoding and decoding.

[2]In brief, γ is the fraction of subfunctions that must be computed locally, and δ is the fraction of available links to be activated. Having $\gamma = 1$ corresponds to the centralized scenario of having to locally calculate all subfunctions, while $\delta = 1$ matches an extreme parallelized scenario that activates all available communication links.

[3]Thus for example, the first column of $\mathbf{E}$ is the coset leader to the coset corresponding to the syndrome described by the first column of $\mathbf{F}$ and the code whose parity check matrix is $\mathbf{D}$.

[4]This reformulation identifies the observed vector $\mathbf{y}$ with the vectorized $\mathbf{F}$, the sparse solution $\mathbf{x}$ with the vectorized $\mathbf{E}$, and the alphabet matrix $\mathbf{A}$ with the Kronecker product of the decoding matrix $\mathbf{D}$ with the identity matrix.

the form $\gamma \leq -\frac{1}{r}\frac{K}{N}W_1^{-1}(\frac{-2K}{erN})$, where though these bounds[1] remained loose and also conditional for two main reasons. The first reason stems from the fact that the focus of the compressed sensing machinery is mainly on the search efficiency and uniqueness of the sparse solutions, rather than on the level of sparsity itself[2]. The second reason is that, while in our computing setting our decoding matrix $\mathbf{D}$ must be a function of $\mathbf{F}$, compressed sensing places its focus on designing $\mathbf{A}$ in a manner that is oblivious to the instance of $\mathbf{y}$. This mismatch is addressed in our work here, allowing us to directly explore the fundamental principles of our computing problem.

C. New connection between distributed computing, fixed support matrix factorization, and tessellations

As we will see almost directly from (2), (3) and (4), (9), solving our distributed computing problem will be equivalent to solving the approximate matrix factorization problem

$$\hat{\mathcal{E}} = \min_{\mathbf{D},\mathbf{E}} \ \|\mathbf{DE} - \mathbf{F}\|_F^2 \tag{10}$$

under dimensionality constraints posed by K, NT, L, and under sparsity constraints on $\mathbf{D}$ and $\mathbf{E}$ posed by δ and γ respectively. These sparsity constraints will be described in detail later on.

This problem encompasses the problem of compressed sensing, and it is known to be hard [20]. In general, finding the optimal solution $(\hat{\mathbf{D}}, \hat{\mathbf{E}}) = \arg\min_{\mathbf{D},\mathbf{E}} \|\mathbf{DE}-\mathbf{F}\|_F^2$ to (10), under the aforementioned dimensionality and sparsity constraints, requires an infeasible coverage of the entire space of solutions. Otherwise, establishing optimality of an algorithmic solution, generally requires establishing uniqueness of that solution which is hard [21], [22]. Furthermore, to date, little is known in terms of clear guarantees on the optimal error performance $\hat{\mathcal{E}}$, for any given $\mathbf{F}$ and any given dimensionality and sparsity constraints on $\mathbf{D}, \mathbf{E}$.

Recently, the work in [23] explored the problem of *Fixed Support (sparse) Matrix Factorization* (FSMF) which — under the same dimensionality and sparsity constraints of the unbounded problem of (10) — seeks to find

$$\hat{\mathcal{E}}_{\mathcal{I},\mathcal{J}} = \min_{\mathbf{D},\mathbf{E}} \ \|\mathbf{DE} - \mathbf{F}\|_F^2, \tag{11}$$
$$\text{Subject to: } \text{supp}(\mathbf{D}) \subseteq \mathcal{I}, \text{supp}(\mathbf{E}) \subseteq \mathcal{J}$$

where $\mathcal{I} \subseteq [K]\times[NT]$ and $\mathcal{J} \subseteq [NT]\times[L]$ respectively define the support constraint $\text{supp}(\mathbf{D})$ and $\text{supp}(\mathbf{E})$ of $\mathbf{D}$ and $\mathbf{E}$ such that $\mathbf{D}(i,j) = 0, \ \forall (i,j) \notin \mathcal{I}$ and $\mathbf{E}(i,j) = 0, \ \forall (i,j) \notin \mathcal{J}$.

[1]Here $W_1(.)$ is the first branch of the Lambert function, while r calibrates the statistical distribution of $\mathbf{D}$.

[2]Search efficiency and uniqueness are not fundamental to our distributed computing problem. For example, what a compressed sensing exposition of our problem effectively shows is that, under the assumption that the sparsest solution for $\mathbf{E}$ has sparsity-level not more than the above $\gamma = -\frac{1}{r}\frac{K}{N}W_1^{-1}(\frac{-2K}{erN})$, and under the additional assumption that this solution is unique, then — with high probability, in the limit of large N — there is an l_1-minimization approach that will efficiently find this sparsest unique solution. For us, the efficiency of identifying $\mathbf{E}$ is of secondary importance, and the possibility of having another equally sparse $\mathbf{E}$ is not an issue.

FSMF remains a broad[3] and challenging problem, partly because, as argued in [23], it does not directly accept the existing algorithms from the unconstrained problem in (10).

After showing that ill-conditioned supports may lead certain algorithms [23] to converge to local minima (referred to as 'spurious local valleys'), the same work in [23] revealed that for some specific $\mathcal{I}, \mathcal{J}$, some algorithms can provably converge to the corresponding $\hat{\mathcal{E}}_{\mathcal{I},\mathcal{J}}$ which is shown to be unique but, clearly, optimal only *within the space of* $\mathbf{D}, \mathbf{E}$ *defined by the specific* support $\mathcal{I}, \mathcal{J}$. The work in [23] placed some of its focus on a particular class of 'disjoint' supports, corresponding to the class of those supports $\mathcal{I}, \mathcal{J}$ that (as we will clarify later on) map onto disjoint[4] regions of $\mathbf{F}$. The finding in [23] is that such 'disjoint' $\mathcal{I}, \mathcal{J}$ render (11) tractable.

Naturally, depending on $\mathcal{I}, \mathcal{J}$, even such optimal 'support-limited' solutions in (11) can have unbounded gaps $\hat{\mathcal{E}}_{\mathcal{I},\mathcal{J}} - \hat{\mathcal{E}}$ to the global optimal $\hat{\mathcal{E}}$ from (10), as we simply do not know how badly the performance deteriorates by limiting the search within the specific fixed-support set of matrices.

In summary, to date, in terms of explicit solutions to the matrix factorization problem in (10), little is known in terms of designing good supports, while in terms of optimality guarantees, these are restricted to within the specific problem in (11) where the search is for a given specific support. To date, little is known about explicitly characterizing a desired error performance $\hat{\mathcal{E}}$ under any desired sparsity constraints.

a) Summary of our contributions on the problem of multi-user distributed computing of linearly-decomposable functions: Having made the connection between matrix factorization and distributed computing, we here identify the FSMF problem to be key in the resolution of our real-valued multi-user distributed computing problem, for which we provide the following results.

In this paper we only focus on the single shot $T = 1$ and lossless case of $\mathcal{E} = 0$. By employing an achievable scheme using novel concepts and algorithms introduced in [23] and a converse using combinatorial tilling arguments [29], Theorem 1 establishes the single shot system capacity under the disjoint support assumption to take the form $C = \frac{K}{N_{opt}}$ where

$$\begin{aligned}\frac{KL}{T\max(\Gamma,\Delta)} \leq N_{opt} \leq \min(\Delta,\Gamma)\lfloor\frac{K}{\Delta}\rfloor\lfloor\frac{L}{\Gamma}\rfloor \\ + \min(\text{mod}(K,\Delta),\Gamma\rfloor)\lfloor\frac{L}{\Gamma}\rfloor \\ + \min(\text{mod}(L,\Gamma),\Delta)\lfloor\frac{K}{\Delta}\rfloor \\ + \min(\text{mod}(K,\Delta),\text{mod}(L,\Gamma)).\end{aligned}$$

This general case is of particular interest because the tessellation patterns that we must design, must accommodate for

[3]For connections between the FSMF problem with *Low rank matrix approximation* [24], *LU decomposition* [25], *Butterfly structure and fast transforms* [26], Hierarchical $\mathcal{H}$ -matrices [27] and *matrix completion* [28], the reader may read [23].

[4]Equivalently, this 'disjoint support' assumption simply says that each element of $\mathbf{F}$ is approximated only by a single SVD.

tiles of various sizes and shapes. In terms of insight, of more interest is the simplified case of (33), which applies to the relatively broad setting of $\delta^{-1}, \gamma^{-1} \in \mathbb{N}$ (corresponding to having $\Delta | K, \Gamma | L$), where the capacity now takes the insightful form

$$C = \max(\zeta, \gamma). \tag{12}$$

D. Paper Organization

The rest of the paper is organized as follows. Section II formulates the system model for the setting of multi-user distributed computing of linearly-decomposable functions. Section III addresses the error-free case, providing schemes and converses that lead to Theorem 1. Subsequently, In Section IV, we conclude the paper.

Notations: We define $[n] \triangleq \{1, 2, \ldots, n\}$. For matrices $\mathbf{A}$ and $\mathbf{B}$, $[\mathbf{A}, \mathbf{B}]$ indicates the horizontal concatenation of the two matrices. For any matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$, then $\mathbf{X}(i, j)$, $i \in [m]$, $j \in [n]$, represents the entry in the ith row and jth column, while $\mathbf{X}(i, :)$, $i \in [m]$, represents the ith row, and $\mathbf{X}(:, j)$, $j \in [n]$ represents the jth column of $\mathbf{X}$. For two index sets $\mathcal{I} \subset [m], \mathcal{J} \in [n]$, then $\mathbf{X}(\mathcal{I}, \mathcal{J})$ represents the submatrix comprised of the rows in $\mathcal{I}$ and columns in $\mathcal{J}$. We will use $\omega(\mathbf{X})$ to represent the number of nonzero elements of some matrix (or vector) $\mathbf{X}$. We will use $\text{supp}(\mathbf{x}^\intercal)$ to represent the support of some vector $\mathbf{x}^\intercal \in \mathbb{R}^n$, describing the set of indices of non-zero elements. Also $\text{supp}(\mathbf{X})$ to represent the support of some matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$, describing the set of two-dimensional indices $(i, j) \in [m] \times [n]$ of non-zero elements. We will also use the notation $\mathcal{E}(n)$ to represent an expression which, in the limit of large n, vanishes to zero. $\mathbb{I}(\mathcal{X}, \mathcal{Y}) \in \mathbb{R}^{m \times n}$ represents a matrix that all of its elements are zero except the elements, $\mathbb{I}(i, j), i \in \mathcal{X}, j \in \mathcal{Y}$. $\text{Supp}(\mathbf{A}) \in \{0, 1\}^{m \times n}, \mathbf{A} \in \mathbb{R}^{m \times n}$, is a binary matrix representing the locations of non-zero elements of $\mathbf{A}$. $\|\mathbf{x}\|_0$ corresponds to norm zero of a vector $\mathbf{x} \in \mathbb{R}^n$. For two binary matrices $\mathbf{I}, \mathbf{J} \in \{0, 1\}^{m \times n}$, $(\mathbf{I} \cup \mathbf{J})(i, j) \triangleq \mathbf{I}(i, j) \vee \mathbf{J}(i, j)$, $(\mathbf{I} \cap \mathbf{J})(i, j) \triangleq \mathbf{I}(i, j) \wedge \mathbf{J}(i, j), \mathbf{I}'(i, j) \triangleq \neg \mathbf{I}(i, j), \mathbf{I} \backslash \mathbf{J}(i, j) = \mathbf{I} \cap \neg \mathbf{J}$, where $\vee, \wedge$ and $\neg$ are logical "or", "and" and "negation", respectively. The sign $\mathbf{1}_n$ defines an all-one n dimensional column vector in real numbers and $\mathbf{0}_m$ defines an all-zero m dimensional column vector. The operation $\odot$ represents a Hadamard product. Note also that $\|\mathbf{A}\|_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} a_{i,j}^2}$ is the Frobenius norm of the matrix $\mathbf{A}$. For a real number $x \in \mathbb{R}$, $\lceil x \rceil, \lfloor x \rfloor$, represents respectively the ceiling and floor function of x.

II. Problem Formulation:

We here describe in detail the main parameters of our model, defining matrices $\mathbf{D}, \mathbf{E}, \mathbf{F}$ and the various metrics, rigorously making the link between the distributed computing problem and the factorization in (9) (see also (29)) for our error-free case. Let us consider

$$\mathbf{f} \triangleq [F_1, F_2, \ldots, F_K]^\intercal \in \mathbb{R}^K, \tag{13}$$

$$\mathbf{f}_k \triangleq [f_{k,1}, f_{k,2}, \ldots, f_{k,L}]^\intercal \in \mathbb{R}^L, \; k \in [K], \tag{14}$$

$$\mathbf{w} \triangleq [W_1, W_2, \ldots, W_L]^\intercal \in \mathbb{R}^L \tag{15}$$

where $\mathbf{f}$ represents the vector of desired function outputs F_k from (1), where $\mathbf{f}_k$ represents the vector of function coefficients $f_{k,\ell}$ from (1) for the function requested by user k, and where $\mathbf{w}$ denotes the vector of output files $W_\ell = f_\ell(\cdot)$ again from (1). Then recalling the encoding coefficients $e_{n,\ell,t}$ and transmitted signals $z_{n,t}$ from (2), as well as the decoding coefficients $d_{k,n,t}$ and decoded functions F'_k from (3), we have

$$\mathbf{e}_{n,t} \triangleq [e_{n,1,t}, e_{n,2,t}, \ldots, e_{n,L,t}]^\intercal \in \mathbb{R}^L, \; n \in [N], \; t \in [T], \tag{16}$$

$$\mathbf{z}_n \triangleq [z_{n,1}, z_{n,2}, \ldots, z_{n,T}]^\intercal \in \mathbb{R}^T, \; n \in [N], \tag{17}$$

$$\mathbf{E}_n \triangleq [\mathbf{e}_{n,1}, \mathbf{e}_{n,2}, \ldots, \mathbf{e}_{n,T}]^\intercal \in \mathbb{R}^{T \times L}, \; n \in [N,] \tag{18}$$

$$\mathbf{d}_{k,n} \triangleq [d_{k,n,1}, d_{k,n,2}, \ldots, d_{k,n,T}]^\intercal \in \mathbb{R}^T, \; k \in [K], n \in [N], \tag{19}$$

$$\mathbf{d}_k \triangleq [\mathbf{d}_{k,1}^\intercal, \mathbf{d}_{k,2}^\intercal, \ldots, \mathbf{d}_{k,N}^\intercal]^\intercal \in \mathbb{R}^{N \times T}, \; k \in [K] \tag{20}$$

and thus (Cf. (13)) we have the output vector taking the form

$$\mathbf{f} = [\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_K]^\intercal \mathbf{w} \tag{21}$$

as well as the transmitted vector by server n taking the form

$$\mathbf{z}_n = \mathbf{E}_n \mathbf{w} = [\mathbf{e}_{n,1}, \mathbf{e}_{n,2}, \ldots, \mathbf{e}_{n,T}]^\intercal \mathbf{w}. \tag{22}$$

This allows us to form the matrices

$$\mathbf{F} \triangleq [\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_K]^\intercal \in \mathbb{R}^{K \times L}, \tag{23}$$

$$\mathbf{E} \triangleq [\mathbf{E}_1^\intercal, \mathbf{E}_2^\intercal, \ldots, \mathbf{E}_N^\intercal]^\intercal \in \mathbb{R}^{NT \times L}, \tag{24}$$

$$\mathbf{D} \triangleq [\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_K]^\intercal \in \mathbb{R}^{K \times NT} \tag{25}$$

where $\mathbf{F}$ represents the $K \times L$ matrix of all function coefficients across all the users, where $\mathbf{E}$ represents the aforementioned $NT \times L$ *computing and encoding matrix* capturing the computing and linear-encoding tasks of servers in each shot, and where $\mathbf{D}$ represents the $K \times NT$ *communication and decoding matrix* capturing the communication protocol and the linear decoding task done by each user.

To see the transition to the matrix factorization problem, we first note that from (2) and (21) we have that the overall transmitted vector $\mathbf{z} \triangleq [\mathbf{z}_1^\intercal, \mathbf{z}_2^\intercal, \ldots, \mathbf{z}_N^\intercal]^\intercal \in \mathbb{R}^{N \times T}$ takes the form

$$\mathbf{z} = [\mathbf{E}_1^\intercal, \mathbf{E}_2^\intercal, \ldots, \mathbf{E}_N^\intercal]^\intercal \mathbf{w} = \mathbf{E}\mathbf{w} \tag{26}$$

and then that given the decoding from (3), each retrieved function takes the form

$$F'_k = \mathbf{d}_k^\intercal \mathbf{z} \tag{27}$$

thus resulting in the vector of all retrieved functions taking the form $\mathbf{f}' = [\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_K]^\intercal \mathbf{z}$. Our aim is to set the recovery error

$$\mathcal{E} \triangleq \|\mathbf{f}' - \mathbf{f}\|^2 \tag{28}$$

equal to zero.

Directly from the above, we see that for the error-free case of $\mathcal{E} = 0$, which means that we must decompose $\mathbf{F}$

$$\mathbf{F} = \mathbf{D}\mathbf{E} \tag{29}$$

as seen in (9).

In terms of the corresponding connection to the sparsity of $\mathbf{D}$ and $\mathbf{E}$, we recall from (5) our metrics $\Gamma \triangleq \max_{n\in[N]} |\mathcal{S}_n|$ and $\Delta \triangleq \max_{n\in[N]} |\mathcal{T}_n|$, which, directly from (20)–(22) and from (23)–(25), imply that

$$\max_{n\in[N]} |\cup_{t=1}^{T} \text{supp}(\mathbf{D}(:,(n-1)T+t)| \leq \Delta \tag{30}$$

and that

$$\max_{n\in[N]} |\cup_{t=1}^{T} \text{supp}(\mathbf{E}((n-1)T+t,:))| \leq \Gamma \tag{31}$$

and thus we see how the normalized costs $\delta = \frac{\Delta}{K}, \gamma = \frac{\Gamma}{L}$ from (6) form the upper bound on the fraction of non-zero elements of the columns of $\mathbf{D}$ and rows of $\mathbf{E}$ respectively.

Finally, from (8) we recall the system rate $R = \frac{K}{N}$, the corresponding system capacity C representing the supremum of all rates for error-free function reconstruction.

III. Main Results

Definition 1. [Disjoint Support Assumption] We say that two matrices $\mathbf{D} \in \mathbb{R}^{K\times NT}, \mathbf{E} \in \mathbb{R}^{NT\times L}$, accept the *disjoint support assumption* if and only if for any two columns $\mathbf{D}(:,i), \mathbf{D}(:,i'), i, i' \in [NT]$ of $\mathbf{D}$ and the respective two rows $\mathbf{E}(i,:), \mathbf{E}(i',:)$ of $\mathbf{E}$, then $\text{supp}(\mathbf{D}(:,i)\mathbf{E}(i,:)) = \text{supp}(\mathbf{D}(:,i')\mathbf{E}(i',:))$ or $\text{supp}(\mathbf{D}(:,i)\mathbf{E}(i,:)) \cap \text{supp}(\mathbf{D}(:,i'), \mathbf{E}(i',:)) = \emptyset$.

Theorem 1. *The optimal achievable rate of a $K, N, T = 1, \Gamma, \Delta$ lossless distributed computing setting under the Disjoint support assumption takes the form $C = K/N_{opt}$, where*

$$\begin{aligned} N_{opt} = \min(\Delta,\Gamma)\lfloor\frac{K}{\Delta}\rfloor\lfloor\frac{L}{\Gamma}\rfloor + \min(\text{mod}(K,\Delta),\Gamma\rfloor)\lfloor\frac{L}{\Gamma}\rfloor \\ + \min(\text{mod}(L,\Gamma),\Delta)\lfloor\frac{K}{\Delta}\rfloor \\ + \min(\text{mod}(K,\Delta),\text{mod}(L,\Gamma)). \end{aligned} \tag{32}$$

For the broad setting where $\delta^{-1}, \gamma^{-1} \in \mathbb{N}$, the optimal performance takes the form

$$C = \max(\zeta, \gamma) \tag{33}$$

Proof. Due to lack of space (partly because our proofs require a sizeable set of definitions), we choose to present the proof in the Appendix document (Appendix Section B available online at Dropbox) which describes the achievability of the corresponding decomposition $\mathbf{DE} = \mathbf{F}$, and how this is translated into our distributed computing setting, while abiding by the communication and computation cost constraints. Our solutions are proven optimal for $T = 1$, as the result of the converse whose proof is also presented in the Appendix document (Appendix Section C available online at Dropbox). The extended version of the theorem for multi-shot setting is available in Section III of [30]. To provide some insight, we present a proof sketch.

Sketch of the proof: The proof consists of an achievable scheme and a converse. The achievable scheme consists of three main steps. In the first step, the master node divides $\mathbf{F}$ into properly sized rectangular submatrices (corresponding to 'tiles') where the tiles must remain disjoint, must cover all elements of $\mathbf{F}$, and must maintain a width and length that does not exceed Δ and Γ respectively. In the second step, the master node performs an SVD decomposition for each tile (for each of the aforementioned submatrices of $\mathbf{F}$), so that this SVD defines the way each tile is the product of two submatrices (the so-called left factor and right factor of the SVD). Finally, in the third step, the left and right factors are carefully positioned in order to form the desired $\mathbf{D}$ and $\mathbf{E}$ matrices that in turn define the computation and communication protocol of the distributed computing problem. The converse argument consists of six lemmas. The first lemma establishes restrictions for a one-to-one mapping between the servers and the tiles of $\mathbf{F}$ in all schemes. The second Lemma establishes the necessity of having tiles that jointly cover the entire matrix F, the third lemma restricts the size of each tile under the computation and communication costs conditions, and the fourth lemma presents the equivalence between disjoint support assumption and disjoint tiles and at the end, combining the above steps via fifth and sixth lemma allows us to derive the lower bound on the optimal number of servers required to achieve error-free recovery of functions. □

IV. Conclusion

In this work, we investigated the fundamental limits of multi-user distributed computing of real-valued linearly-decomposable functions by making clear connections to the problem of fixed support matrix factorization and tessellation theory. The error-free system capacity $C = \frac{K}{N_{opt}}$ in Theorem 1 revealed the minimal computational and communication resources γ, δ, N required to accommodate a certain number of users and subfunctions. We observe two optimal operating points, $(\gamma = \frac{K}{N}, \delta = \frac{1}{K})$ and $(\gamma = \frac{1}{L}, \delta = \frac{L}{N})$, compared to baseline points $A = (\gamma = 1, \delta = 1/K)$ and $B = (\gamma = 1/L, \delta = 1)$. Point A represents a fully-centralized scheme where servers $n \in [K]$ handle all subfunctions, while others handle none, and point B corresponds to a fully-parallelized scheme where each server computes a single subfunction output and sends it to all users. These cases align with trivial decompositions $\mathbf{F} = [\mathbf{I}_K \ \mathbf{0}_{(K,K-N)}] \cdot [\mathbf{F}^\intercal \ \mathbf{0}_{(L,N-K)}]^\intercal$ and $\mathbf{F} = [\mathbf{F} \ \mathbf{0}_{(K,N-L)}] \cdot [\mathbf{I}_L \ \mathbf{0}_{(L,N-L)}]$, respectively. The first optimal point achieves a lower γ than A for the same δ when $N \geq K$, while the second achieves a lower δ than B for the same γ when $N \geq L$ [31].

REFERENCES

[1] R. Tandon, Q. Lei, A. G. Dimakis, and N. Karampatziakis, "Gradient coding: Avoiding stragglers in distributed learning," in *International Conference on Machine Learning*, pp. 3368–3376, PMLR, 2017.

[2] M. Ye and E. Abbe, "Communication-computation efficient gradient coding," in *International Conference on Machine Learning*, pp. 5610–5619, PMLR, 2018.

[3] 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.

[4] S. Dutta, V. Cadambe, and P. Grover, "Short-dot: Computing large linear transforms distributedly using coded short dot products," *Advances In Neural Information Processing Systems*, vol. 29, 2016.

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

[6] A. Ramamoorthy, L. Tang, and P. O. Vontobel, "Universally decodable matrices for distributed matrix-vector multiplication," in *2019 IEEE International Symposium on Information Theory (ISIT)*, pp. 1777–1781, IEEE, 2019.

[7] Z. Jia and S. A. Jafar, "Cross subspace alignment codes for coded distributed batch computation," *IEEE Transactions on Information Theory*, vol. 67, no. 5, pp. 2821–2846, 2021.

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

[9] M. Zinkevich, M. Weimer, L. Li, and A. Smola, "Parallelized stochastic gradient descent," in *Advances in Neural Information Processing Systems*, vol. 23, 2010.

[10] T. Chilimbi, Y. Suzue, J. Apacible, and K. Kalyanaraman, "Project Adam: Building an efficient and scalable deep learning training system," in *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*, pp. 571–582, 2014.

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

[12] K. Wan, H. Sun, M. Ji, and G. Caire, "On the tradeoff between computation and communication costs for distributed linearly separable computation," *IEEE Transactions on Communications*, vol. 69, no. 11, pp. 7390–7405, 2021.

[13] K. Wan, H. Sun, M. Ji, and G. Caire, "On secure distributed linearly separable computation," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 3, pp. 912–926, 2022.

[14] A. Khalesi and P. Elia, "Multi-user linearly-separable distributed computing," *IEEE Transactions on Information Theory*, vol. 69, no. 10, pp. 6314–6339, 2023.

[15] M. Soleymani, H. Mahdavifar, and A. S. Avestimehr, "Privacy-preserving distributed learning in the analog domain," *arXiv preprint arXiv:2007.08803*, 2020.

[16] O. Makkonen and C. Hollanti, "Analog secure distributed matrix multiplication over complex numbers," in *2022 IEEE International Symposium on Information Theory (ISIT)*, pp. 1211–1216, IEEE, 2022.

[17] M. Soleymani, H. Mahdavifar, and A. S. Avestimehr, "Analog lagrange coded computing," *IEEE Journal on Selected Areas in Information Theory*, vol. 2, no. 1, pp. 283–295, 2021.

[18] A. Khalesi and P. Elia, "Multi-user linearly separable computation: A coding theoretic approach," in *2022 IEEE Information Theory Workshop (ITW)*, pp. 428–433, 2022.

[19] A. Khalesi, S. Daei, M. Kountouris, and P. Elia, "Multi-user distributed computing via compressed sensing," in *2023 IEEE Information Theory Workshop (ITW)*, pp. 509–514, 2023.

[20] R. Gribonval and K. Schnass, "Dictionary identification—sparse matrix-factorization via $l-1$-minimization," *IEEE Transactions on Information Theory*, vol. 56, no. 7, pp. 3523–3539, 2010.

[21] L. Zheng, E. Riccietti, and R. Gribonval, "Identifiability in two-layer sparse matrix factorization," *arXiv preprint arXiv:2110.01235*, 2021.

[22] L. Zheng, E. Riccietti, and R. Gribonval, "Efficient identification of butterfly sparse matrix factorizations," *SIAM Journal on Mathematics of Data Science*, vol. 5, no. 1, pp. 22–49, 2023.

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

[24] C. Eckart and G. Young, "The approximation of one matrix by another of lower rank," *Psychometrika*, vol. 1, no. 3, pp. 211–218, 1936.

[25] C. F. Van Loan and G. Golub, "Matrix computations (johns hopkins studies in mathematical sciences)," *Matrix Computations*, vol. 5, 1996.

[26] T. Dao, A. Gu, M. Eichhorn, A. Rudra, and C. Ré, "Learning fast algorithms for linear transforms using butterfly factorizations," in *International conference on machine learning*, pp. 1517–1527, PMLR, 2019.

[27] W. Hackbusch, "A sparse matrix arithmetic based on-matrices. part i: Introduction to-matrices," *Computing*, vol. 62, no. 2, pp. 89–108, 1999.

[28] C. R. Johnson, "Matrix completion problems: a survey," in *Matrix theory and applications*, vol. 40, pp. 171–198, 1990.

[29] F. Ardila and R. P. Stanley, "Tilings," *The Mathematical Intelligencer*, vol. 32, no. 4, pp. 32–43, 2010.

[30] A. Khalesi and P. Elia, "Tessellated distributed computing," *arXiv preprint arXiv:2404.14203*, 2024.

[31] A. Khalesi and P. Elia, "Lossless tessellated distributed computing: Full version," 2025. Available online at Dropbox: https://www.dropbox.com/scl/fi/h6u2knbq6xpnuwvypanou/Lossless_Tessellated_Distributed_Computing.pdf?rlkey=cfyrn9k8l37hzx8t7lozpevp8&st=eslvpqfg&dl=0.

APPENDIX

A. Appendix: A Primer on SVD Matrix Decomposition

It is well known that SVD is the optimal way to approximate matrices.

For the case of $m > n$, this decomposition takes the form

$$\mathbf{A}_{m\times n} = \mathbf{U}_{m\times m}\mathbf{S}_{m\times n}\mathbf{V}^{\mathsf{T}}_{n\times n} \tag{34}$$

where $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices, and where $\mathbf{S}$ is an diagonal $m \times n$ matrix whose diagonal entries $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n \geq 0$ are in descending order.

*Thin **SVD***: Since $m \geq n$, the decomposition also can take the form

$$\mathbf{A}_{m\times n} = \mathbf{U}_{m\times n}\mathbf{S}_{n\times n}\mathbf{V}^{\mathsf{T}}_{n\times n} \tag{35}$$

here $\mathbf{S}_{n\times n} = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_n)$.

When the rank of $\mathbf{A}$ is $r \leq \min(m, n)$, then its SVD takes the form

$$\mathbf{A} = \mathbf{U}_{m\times r}\mathbf{S}_{r\times r}\mathbf{V}^{\mathsf{T}}_{r\times n} \tag{36}$$

$$= \begin{bmatrix}\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_r\end{bmatrix} \begin{bmatrix}\sigma_1 & & & 0 \\ & \sigma_2 & & \\ & & \ddots & \\ 0 & & & \sigma_r\end{bmatrix} \begin{bmatrix}\mathbf{v}_1^{\mathsf{T}} \\ \mathbf{v}_2^{\mathsf{T}} \\ \vdots \\ \mathbf{v}_r^{\mathsf{T}}\end{bmatrix} \tag{37}$$

$$= \sum_{i=1}^{r} \sigma_i u_i v_i^{\mathsf{T}} \tag{38}$$

where $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices, where $\mathbf{S}$ is diagonal with entries $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n \geq 0$ being the singular values in descending order, where $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_r$ are the columns of $U_{m\times r}$, and where $\mathbf{v}_1^{\mathsf{T}}, \mathbf{v}_2^{\mathsf{T}}, \ldots, \mathbf{v}_r^{\mathsf{T}}$ are columns of $V_{n\times r}$.

B. Scheme for Lossless Reconstruction and Achievability Proof for Theorem 1

We proceed to describe the design of the communication matrix $\mathbf{D}$ and the computing matrix $\mathbf{E}$ that yield $\mathbf{DE} = \mathbf{F}$, while maintaining $N, T = 1$ as well as a maximum of Δ non-zero elements in any column of $\mathbf{D}$ and a maximum of Γ non-zero elements in any row of $\mathbf{E}$. These constraints guarantee that each of the N servers can locally calculate up to Γ subfunctions and can engage in communication with at most Δ users. The design borrows concepts and definitions from the blockwise SVD approach of [23].

In this proof we first in Part B1 begin with defining the nth rank-one contribution support, representative rank-one support, or tile and their related parameters. These tiles are actually classes of rank-one contribution supports which shows how any support constraint on $\mathbf{D}, \mathbf{E}$ contributes to the supports on $\mathbf{DE}$. The aforementioned correspondence is shown via Lemma 1. Then in Part B2, we show how to design the tiles for the product matrix $\mathbf{DE}$, create and fill the non-zero tiles in $\mathbf{D}$ and $\mathbf{E}$ and place the filled tiles in $\mathbf{D}$ and in $\mathbf{E}$ for both single-shot. The number of servers required for the single-shot required the rank of each tile to the servers and then summing the ranks over all tiles. Example 1 illustrates the performance of our scheme in the cases where $\Delta \nmid K, \Gamma \nmid L, T = 1$.

1) Basic Concepts and Definitions: We first present the following definitions[1].

Definition 2 ([23])**.** Given two support constraints $\mathbf{I} \in \{0,1\}^{K\times N}$ and $\mathbf{J} \in \{0,1\}^{N\times L}$, then for any $n \in [N]$, we refer to $\mathbf{S}_n(\mathbf{I}, \mathbf{J}) \triangleq \mathbf{I}(:, n)\mathbf{J}(n, :)$ as the nth *rank-one contribution support*.

We note that when the supports are implied, we may shorten $\mathbf{S}_n(\mathbf{I}, \mathbf{J})$ to just $\mathbf{S}_n$. The aforementioned $\mathbf{I}$ and $\mathbf{J}$ will generally represent the support of $\mathbf{D}$ and $\mathbf{E}$ respectively, while $\mathbf{S}_n(\mathbf{I}, \mathbf{J})$ will generally capture some of the support of $\mathbf{DE}$ and thus of $\mathbf{F}$. On this, we have the following lemma.

Lemma 1. For $\mathbf{I} \triangleq \mathrm{supp}(\mathbf{D})$ and $\mathbf{J} \triangleq \mathrm{supp}(\mathbf{E})$, then

$$\cup_{n=1}^{N}\mathbf{S}_n(\mathbf{I}, \mathbf{J}) = \cup_{n=1}^{N}\mathbf{I}(:, n)\mathbf{J}(n, :) \supseteq \mathrm{supp}(\mathbf{DE}). \tag{39}$$

Proof. The above follows from Definition 2 and from the fact that $\mathbf{DE} = \sum_{n=1}^{N} \mathbf{D}(:, n)\mathbf{E}(n, :)$. □

We also need the following definition.

Definition 3 ([23])**.** For $\mathbf{I} \in \{0,1\}^{K\times N}$ and $\mathbf{J} \in \{0,1\}^{N\times L}$, the *equivalence classes of rank-one supports* are defined by the equivalence relation $i \sim j$ on $[N]$ which holds if and only if $\mathbf{S}_i = \mathbf{S}_j$.

The above splits the columns of $\mathbf{D}$ (and correspondingly the rows of $\mathbf{E}$) into equivalence classes such that $i \sim j$ holds if and only if $\mathbf{I}(:, i)\mathbf{J}(i, :) = \mathbf{I}(:, j)\mathbf{J}(j, :)$.

Definition 4. For two supports $\mathbf{I} \in \{0,1\}^{K\times N}, \mathbf{J} \in \{0,1\}^{N\times L}$, and for $\mathcal{C}$ being the collection of equivalence classes as in Definition 3, then for each class $\mathcal{P} \in \mathcal{C}$, we call $\mathbf{S}_\mathcal{P}$ to be the *representative rank-one support* of class $\mathcal{P}$, which will also be called the *tile* labeled[2] by $\mathcal{P}$. Furthermore for each tile $\mathbf{S}_\mathcal{P}$, let $\mathbf{c}_\mathcal{P} \triangleq \mathbf{I}(:, n), n \in \mathcal{P}$ (resp. $\mathbf{r}_\mathcal{P} \triangleq \mathbf{J}(n, :), n \in \mathcal{P}$) be the corresponding *component column* (resp. *component row*) of the representative rank-one support. Finally $\mathcal{C}_\mathcal{P} \triangleq \mathrm{supp}(\mathbf{c}_\mathcal{P}) \subset [K]$ describes the set of the indices of the non-zero elements in $\mathbf{c}_\mathcal{P}$, while $\mathcal{R}_\mathcal{P} \triangleq \mathrm{supp}(\mathbf{r}_\mathcal{P}) \subset [L]$ describes the set of indices of the non-zero elements in $\mathbf{r}_\mathcal{P}$.

Definition 5. For every set $\mathcal{C}' \subseteq \mathcal{C}$ of equivalence classes, let us define the *union of the representative rank-one supports* $\mathcal{S}_{\mathcal{C}'} \triangleq \cup_{\mathcal{P}\in\mathcal{C}'}\mathcal{S}_\mathcal{P}$ to be the point-wise logical OR of the corresponding $\mathcal{S}_\mathcal{P}$, and let $\bar{\mathcal{S}}_\mathcal{P} \triangleq \{1\}^{K\times L} - \mathcal{S}_\mathcal{P}$ be its complement.

In the above, $\mathcal{S}_{\mathcal{C}'}$ is simply the area of the product matrix covered by all the tiles $\mathcal{P}$ in $\mathcal{C}'$. Furthermore we have the following definition.

[1]We quickly recall that for a matrix $\mathbf{A}$, then $\mathbf{A}(:, n)$ represents its nth column, $\mathbf{A}(n, :)$ its nth row, $\mathrm{supp}(\mathbf{A})$ the binary matrix indicating the support of $\mathbf{A}$, while for a vector $\mathbf{a}$, then $\mathrm{supp}(\mathbf{a})$ represents the set of indices of $\mathbf{a}$ with non-zero elements.

[2]Note that $\mathbf{S}_n = \mathbf{S}_\mathcal{P}$ for any $n \in \mathcal{P}$. Also the term *tile* will be used interchangeably to represent both $\mathbf{S}_\mathcal{P}$ and $\mathcal{P}$.

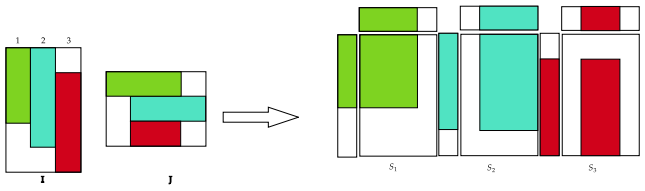

Fig. 2. The figure on the left illustrates the support constraints $\mathbf{I}$ and $\mathbf{J}$ on $\mathbf{D}$ and $\mathbf{E}$ respectively. The constraints $\mathbf{I}(:,1)$ and $\mathbf{J}(1.\ :)$ on the columns and rows of $\mathbf{D}$ and $\mathbf{E}$ respectively are colored green, $\mathbf{I}(:,2)$ and $\mathbf{J}(2.\ :)$ are colored cyan and $\mathbf{I}(:,3)$ and $\mathbf{J}(3.\ :)$ are colored red. The product of a column with a row of the same color, yields the corresponding rank-1 contribution support $\mathbf{S}_n(\mathbf{I},\mathbf{J}), n=1,2,3$, as described in Definition 2, and as illustrated on the right side of the figure.

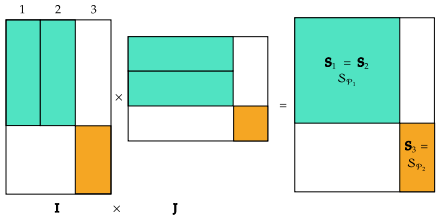

Fig. 3. This figure illustrates three different rank-1 contribution supports $\mathbf{S}_1,\mathbf{S}_2,\mathbf{S}_3$, where the first two fall into the same equivalence class $\mathcal{S}_{\mathcal{P}_1}=\mathbf{S}_1=\mathbf{S}_2$, while $\mathcal{S}_{\mathcal{P}_2}=\mathbf{S}_3$.

Definition 6 ([23])**.** The *maximum rank of a representative rank-one support of class* $\mathcal{P}\subset\mathcal{C}$ is

$$r_{\mathcal{P}} \triangleq \min(|\mathcal{C}_{\mathcal{P}}|,|\mathcal{R}_{\mathcal{P}}|). \tag{40}$$

The above simply says that for the case of $\mathbf{I}=\text{supp}(\mathbf{D})$ and $\mathbf{J}=\text{supp}(\mathbf{E})$, then the part of $\mathbf{DE}$ matrix covered by tile $\mathcal{S}_{\mathcal{P}}$ can have rank which is at most $r_{\mathcal{P}}$.

With the above in place, we proceed to describe the method used to design the two matrices $\mathbf{D},\mathbf{E}$.

2) Construction of $\mathbf{D},\mathbf{E}$*:* The steps will involve 1) Designing the sizes of the tiles for the product matrix $\mathbf{DE}$, and placing them, 2) Creating and filling the non-zero tiles in $\mathbf{D}$ and $\mathbf{E}$, 3) Placing the filled tiles in $\mathbf{D}$ and in $\mathbf{E}$. We first address the case of $T=1$, which we then generalize to larger T. There will also be two clarifying examples that follow the description of the design.

3) Designing $\mathbf{D},\mathbf{E}$ *for the case of* $T=1$*:*

a) Step 1: Designing the sizes of the tiles for the product $K\times L$ *matrix* $\mathbf{DE}$*, and placing the tiles:* As the first step, we assign the families of the equivalence classes of rank-one supports as follows

$$\mathcal{C}_1 \triangleq \{\mathcal{P}_{i,j}|\mathbf{c}_{\mathcal{P}_{i,j}}=[\mathbf{0}^\intercal_{(i-1)\Delta},\mathbf{1}^\intercal_{\Delta},\mathbf{0}^\intercal_{K-i\Delta}]^\intercal,\mathbf{r}_{\mathcal{P}_{i,j}}=[\mathbf{0}^\intercal_{(j-1)\Gamma},\mathbf{1}^\intercal_{\Gamma},\mathbf{0}^\intercal_{L-i\Gamma}],(i,j)\in[\lfloor\frac{K}{\Delta}\rfloor]\times[\lfloor\frac{L}{\Gamma}\rfloor]\}, \tag{41}$$

$$\mathcal{C}_2 \triangleq \{\mathcal{P}_i|\mathbf{c}_{\mathcal{P}_i}=[\mathbf{0}^\intercal_{(i-1)\Delta},\mathbf{1}^\intercal_{\Delta},\mathbf{0}^\intercal_{K-i\Delta}]^\intercal,\mathbf{r}_{\mathcal{P}_i}=[\mathbf{0}^\intercal_{L-\text{mod}(L,\Gamma)},\mathbf{1}^\intercal_{\text{mod}(L,\Gamma)}],i\in[\lfloor\frac{K}{\Delta}\rfloor]\}, \tag{42}$$

$$\mathcal{C}_3 \triangleq \{\mathcal{P}_j|\mathbf{c}_{\mathcal{P}_j}=[\mathbf{0}^\intercal_{K-\text{mod}(K,\Delta)},\mathbf{1}^\intercal_{\text{mod}(K,\Delta)}]^\intercal,\mathbf{r}_{\mathcal{P}_j}=[\mathbf{0}^\intercal_{(j-1)\Gamma},\mathbf{1}^\intercal_{\Gamma},\mathbf{0}^\intercal_{L-i\Gamma}],j\in[\lfloor\frac{L}{\Gamma}\rfloor]\}, \tag{43}$$

$$\mathcal{C}_4 \triangleq \{\mathcal{P}|\mathbf{c}_{\mathcal{P}}=[\mathbf{0}^\intercal_{K-\text{mod}(K,\Delta)},\mathbf{1}^\intercal_{\text{mod}(K,\Delta)}]^\intercal,\mathbf{r}^\intercal_{\mathcal{P}}=[\mathbf{0}^\intercal_{L-\text{mod}(L,\Gamma)},\mathbf{1}^\intercal_{\text{mod}(L,\Gamma)}]\}. \tag{44}$$

The number of representative rank-one supports in each class, is

$$|\mathcal{C}_1|=\lfloor\frac{K}{\Delta}\rfloor\lfloor\frac{L}{\Gamma}\rfloor,\ |\mathcal{C}_2|=\lfloor\frac{K}{\Delta}\rfloor,\ |\mathcal{C}_3|=\lfloor\frac{L}{\Gamma}\rfloor,\ |\mathcal{C}_4|=1. \tag{45}$$

Also, from (40) and Definition 6, we see that the maximum rank of each representative rank-one support takes the form

$$r_{\mathcal{P}}=\begin{cases}\min(\Delta,\Gamma), & \text{if } \mathcal{P}\in\mathcal{C}_1,\\ \min(\text{mod}(K,\Delta),\Gamma), & \text{if } \mathcal{P}\in\mathcal{C}_2,\\ \min(\text{mod}(L,\Gamma),\Delta), & \text{if } \mathcal{P}\in\mathcal{C}_3,\\ \min(\text{mod}(K,\Delta),\text{mod}(L,\Gamma)), & \text{if } \mathcal{P}\in\mathcal{C}_4.\end{cases} \tag{46}$$

Having established the classes and the corresponding (position of the) tiles of $\mathbf{F}$, we proceed to fill (and crop) the $\mathbf{F}$ tiles, as follows:

$$\mathbf{F}_{\mathcal{P}} \triangleq (\mathbf{F}\odot\mathbf{S}_{\mathcal{P}})(\mathcal{R}_{\mathcal{P}},\mathcal{C}_{\mathcal{P}}),\ \forall\mathcal{P}\in\cup_{i=1}^4\mathcal{C}_i. \tag{47}$$

The first step $\mathbf{F}\odot\mathbf{S}_{\mathcal{P}}$ simply fills up $\mathbf{S}_{\mathcal{P}}$ with the corresponding entries of $\mathbf{F}$, and the second step $(\mathbf{F}\odot\mathbf{S}_{\mathcal{P}})(\mathcal{R}_{\mathcal{P}},\mathcal{C}_{\mathcal{P}})$ crops these[1].

b) Step 2: Creating and filling the non-zero tiles in $\mathbf{D}$ *and* $\mathbf{E}$*:* This step starts with the SVD decomposition as (described in Section A) of the cropped tile $\mathbf{F}_{\mathcal{P}}$ where this decomposition takes the form

$$\mathbf{F}_{\mathcal{P}}=\mathbf{D}_{\mathcal{P}}\mathbf{E}_{\mathcal{P}} \tag{48}$$

where $\mathbf{D}_{\mathcal{P}}\in\mathbb{R}^{|\mathcal{R}_{\mathcal{P}}|\times r_{\mathcal{P}}},\mathbf{E}_{\mathcal{P}}\in\mathbb{R}^{r_{\mathcal{P}}\times|\mathcal{C}_{\mathcal{P}}|}$. In particular $\mathbf{F},\mathbf{D}$ and $\mathbf{E}$ are associated to $\mathbf{A}$, $\mathbf{US}$ and $\mathbf{V}$ in all complete SVD decompositions of (34), (35) and (36). Naturally $\text{rank}(\mathbf{F}_{\mathcal{P}})\leq r_{\mathcal{P}}$.

c) Step 3: Placing the filled cropped tiles $\mathbf{D}_{\mathcal{P}}$ *and* $\mathbf{E}_{\mathcal{P}}$ *in* $\mathbf{D}$ *and* $\mathbf{E}$*:* Let

$$\cup_{i=1}^4\mathcal{C}_i=\{\mathcal{P}_1,\mathcal{P}_2,\ldots,\mathcal{P}_m\},\ m\in\mathbb{N} \tag{49}$$

describe the enumeration we give to each tile. Then the position that each cropped tile takes inside $\mathbf{D}$, is given by

$$\mathcal{R}_{\mathcal{P}_j},\ [\sum_{i=1}^{j-1}T\lceil\frac{r_{\mathcal{P}_i}}{T}\rceil+1:\sum_{i=1}^{j}T\lceil\frac{r_{\mathcal{P}_i}}{T}\rceil],\ \forall\mathcal{P}_j\in\cup_{i=1}^4\mathcal{C}_i \tag{50}$$

and the position of each cropped tile in $\mathbf{E}$ is given by

$$[\sum_{i=1}^{j-1}T\lceil\frac{r_{\mathcal{P}_i}}{T}\rceil+1:\sum_{i=1}^{j}T\lceil\frac{r_{\mathcal{P}_i}}{T}\rceil],\ \mathcal{C}_{\mathcal{P}_j},\ \forall\mathcal{P}_j\in\cup_{i=1}^4\mathcal{C}_i. \tag{51}$$

[1]We here see a small distinction between the previously uncropped tiles, and the tiles here that are cropped. These cropped tiles will be the outcomes of an SVD decomposition of submatrices of $\mathbf{F}$, which will yield our SVD-based factorization.

In particular, for $T=1$ this yields

$$\mathbf{D}(\mathcal{R}_{\mathcal{P}_j}, [\sum_{i=1}^{j-1} r_{\mathcal{P}_i}+1, \sum_{i=1}^{j} r_{\mathcal{P}_i}]) = \mathbf{D}_{\mathcal{P}_j} \tag{52}$$

and

$$\mathbf{E}([\sum_{i=1}^{j-1} r_{\mathcal{P}_i}+1, \sum_{i=1}^{j} r_{\mathcal{P}_i}], \mathcal{C}_{\mathcal{P}_j}) = \mathbf{E}_{\mathcal{P}_j} \tag{53}$$

while naturally the remaining non-assigned elements of $\mathbf{D}$ and $\mathbf{E}$ are zero. Finally, as one can readily verify, the above design corresponds to

$$\begin{aligned} N &= \sum_{i\in[4]} \sum_{\mathcal{P}\in\mathcal{C}_i} r_{\mathcal{P}} |\mathcal{C}_i| && (54)\\ &= \min(\Delta,\Gamma)\lfloor\frac{K}{\Delta}\rfloor\lfloor\frac{L}{\Gamma}\rfloor && (55)\\ &+ \min(\mathrm{mod}(K,\Delta),\Gamma)\lfloor\frac{L}{\Gamma}\rfloor \\ &+ \min(\mathrm{mod}(L,\Gamma),\Delta)\lfloor\frac{K}{\Delta}\rfloor && (56)\\ &+ \min(\mathrm{mod}(K,\Delta),\mathrm{mod}(L,\Gamma)) && (57) \end{aligned}$$

corresponding to Theorem 1 for the case of $T=1$. To evaluate the performance of our scheme as described in (33), we note that the capacity follows as

$$C_0 \overset{(a)}{=} \frac{K}{N_{\text{opt}}} \overset{(b)}{=} \frac{\Delta\Gamma}{L\min(\Delta,\Gamma)} \overset{(c)}{=} (1/L)(\Delta\Gamma/\min(\Delta,\Gamma)),$$

where (a) follows by definition, (b) follows from Theorem (1), and (c) follows after basic algebraic manipulations. The claim is then completed directly after applying (7) and (6).

C. Appendix:Proof of The Converse for Theorem 1

The converse that we provide here will prove that the scheme for the single shot case is exactly optimal when $\Gamma \geq \Delta, \Gamma | L, T|\Delta$ or $\Delta \geq \Gamma, \Delta | K, T|\Gamma$ and also the scheme for the multi-shot case is optimal when $T \geq \min(\Delta,\Gamma)$. We begin with three lemmas that will be useful later on. First, Lemma 2 will state the necessity of having a one-to-one correspondence between each rank-one contribution support[1] and each server, then Lemma 3 will state the necessity of having a tessellation pattern that covers the whole area of $\mathbf{F}$, and then Lemma 4 will elaborate more on size limits of each tile as a consequence of the communication and computation constraints. Lemma 5 then establishes the conceptual equivalence between Definition 1 and disjoint tiles. Lemma 6, lower bounds the number of tiles is each equivalence class by the number of subtiles (cf. Definition 6) corresponding to a tile. Subsequently, Lemma 7, gives the necessary number of subtiles, so that a covering scheme can be constructed, then via combining the last two of the aforementioned lemmas, we give a lower bound on the number of rank-one contribution supports for any covering scheme with proper sizes, and then using Lemma 2, we finalize our converse by giving a lower-bound on the number of servers.

Before presenting the lemmas, let us recall that rank-one contribution supports were defined in Definition 2, that representative supports (i.e. tiles) were defined in Definition 4, as well as let us recall from the same definition that the collection of all classes is represented by $\mathcal{C}$.

The following lemma, while stating the obvious, will be useful in associating the number of servers to the number of rank-one contribution supports.

Lemma 2. For any $\mathbf{D}, \mathbf{E}$ with respective supports $\mathbf{I} = \mathrm{supp}(\mathbf{D}) \in \{0,1\}^{K\times N}$ and $\mathbf{J} = \mathrm{supp}(\mathbf{E}) \in \{0,1\}^{N\times L}$, there exists a one-to-one mapping between the server indices $n \in [N]$ and rank-one contribution supports $\mathbf{S}_n(\mathbf{I},\mathbf{J})$.

Proof. The proof is direct by first recalling Definition 2 which, for any $n \in [N]$, says that $\mathbf{S}_n(\mathbf{I},\mathbf{J}) = \mathbf{I}(:,n)\mathbf{J}(n,:)$, and then by recalling from (16),(19),(24) and (25) that, in the single-shot setting, each server $n \in \mathbb{N}$ corresponds to the nth column of $\mathbf{D}$ (itself corresponding to $\mathbf{I}(:,n)$) and the nth row of $\mathbf{E}$ (corresponding to $\mathbf{J}(n,:)$). □

We proceed with the next lemma, which simply says that every element of $\mathbf{F}$ must belong to at least one representative support (tile).

Lemma 3. In any lossless function reconstruction scheme corresponding to $\mathbf{DE} = \mathbf{F}$, for each $(i,j) \in [K]\times[L]$, there exists a class $\exists \mathcal{P} \in \mathcal{C}$ such that $(i,j) \in \mathcal{R}_{\mathcal{P}} \times \mathcal{C}_{\mathcal{P}}$.

Proof. The lemma aims to prove that there exists no element $\mathbf{F}(i,j)$ of $\mathbf{F}$ that has not been mapped to a tile $\mathbf{F}_{\mathcal{P}}$. We will prove that for each $(i,j) \in [K]\times[L]$ then $\exists \mathcal{P} \in \mathcal{C} \ : \ (i,j) \in \mathcal{R}_{\mathcal{P}} \times \mathcal{C}_{\mathcal{P}}$ and we will do so by contradiction. Let us thus assume that there exists $(i,j) \in [K]\times[L]$ such that $\nexists \mathcal{P} \in \mathcal{C} \ : \ (i,j) \in \mathcal{R}_{\mathcal{P}} \times \mathcal{C}_{\mathcal{P}}$, which would in turn imply — directly from (52), (53) — that $\mathbf{DE}(i,j) = 0$ as well as would imply the aforementioned fact that the non-assigned (by the process in (52), (53)) elements of $\mathbf{D}$ and $\mathbf{E}$, are zero. Now we see that

$$\begin{aligned} \|(\mathbf{DE}-\mathbf{F})\mathbf{w}\|_2^2 &= \|\sum_{k=1}^{K} (\mathbf{DE}-\mathbf{F})(k,:)\mathbf{w}\|_2^2 \\ &= \sum_{k=1,k\neq i}^{K} \sum_{\ell=1,\ell\neq j}^{L} [(\mathbf{DE}-\mathbf{F})(k,\ell)\mathbf{w}(\ell)]^2 \\ &+ (\mathbf{DE}-\mathbf{F})^2(i,j)\mathbf{w}^2(j) \\ &+ 2(\mathbf{DE}-\mathbf{F})(i,j)\mathbf{w}(j) \\ &\sum_{k=1,k\neq i}^{K} \sum_{\ell=1,\ell\neq j}^{L} (\mathbf{DE}-\mathbf{F})(k,\ell)\mathbf{w}(\ell) \\ &= \sum_{k=1,k\neq i}^{K} \sum_{\ell=1,\ell\neq j}^{L} [(\mathbf{DE}-\mathbf{F})(\ell,k)\mathbf{w}(\ell)]^2 \\ &+ \mathbf{F}^2(i,j)\mathbf{w}^2(j) \\ &- 2\mathbf{F}(i,j)\mathbf{w}(j) \end{aligned}$$

[1]Recall that the nth rank-one contribution support takes the form $\mathbf{S}_n(\mathbf{I},\mathbf{J}) = \mathbf{I}(:,n)\mathbf{J}(n,:)$.

$$\sum_{k=1,k\neq i}^{K}\sum_{\ell=1,\ell\neq j}^{L}(\mathbf{DE}-\mathbf{F})(\ell,k)\mathbf{w}(\ell). \quad (58)$$

Let us now recall that lossless function reconstruction implies that $\|(\mathbf{DE}-\mathbf{F})\mathbf{w}\|_2^2 = 0$ for all $\mathbf{F} \in \mathbb{R}^{K\times L}$ and all $\mathbf{w} \in \mathbb{R}^L$. Under the special case of $\mathbf{w}(\ell) = 0, \forall \ell \in [L]\backslash\{j\}$ and $\mathbf{w}(j)\mathbf{F}(i,j) \neq 0$, we see — directly from (58) — that $\|(\mathbf{DE}-\mathbf{F})\mathbf{w}\|_2^2 = \mathbf{F}^2(i,j)\mathbf{w}^2(j) \neq 0$, which contradicts the lossless assumption, thus concluding the proof of the lemma. □

The next lemma now limits the sizes of each tile.

Lemma 4. For any feasible scheme yielding $\mathbf{DE} = \mathbf{F}$, then each representative support $\mathcal{P} \in \mathcal{C}$ satisfies

$$0 < \|\mathbf{S}_{\mathcal{P}}(k,:)\|_0 \leq \Gamma, \forall k \in \mathcal{R}_{\mathcal{P}}\,,$$
$$0 < \|\mathbf{S}_{\mathcal{P}}(:,l)\|_0 \leq \Delta, \forall l \in \mathcal{C}_{\mathcal{P}}$$

which means that each $\mathbf{S}_{\mathcal{P}}$ can have at most Γ non-zero elements in each row and Δ non-zero elements in each column.

Proof. We first recall from Definition 2 that $\text{supp}(\mathbf{D}) = \mathbf{I}, \text{supp}(\mathbf{E}) = \mathbf{J}$. We also recall that $\max_{n\in[N]} |\cup_{t=1}^{T} \text{supp}(\mathbf{D}(:,(n-1)T+t))| \leq \Delta$ and $\max_{n\in[N]} |\cup_{t=1}^{T} \text{supp}(\mathbf{E}((n-1)T+t,:))| \leq \Gamma$ (cf. (30),(31)) must hold for any feasible scheme. Hence for all $n \in [NT]$, we have that $\|\mathbf{I}(:,n)\|_0 \leq \Delta, \|\mathbf{J}(n,:)\|_0 \leq \Gamma$, and consequently since $\mathbf{S}_n = \mathbf{I}(:,n)\mathbf{J}(n,:)$ (cf. Definition 2), we must have that $\|\mathbf{S}_n(k,:)\|_0 \leq \Gamma, \forall k \in [K]$, $\|\mathbf{S}_n(:,l)\|_0 \leq \Delta, \forall l \in [L]$ $\forall n \in [N]$. Then from Definition 4, we see that for all $\mathcal{P} \in \mathcal{C}$, there exists an $n \in [N]$ such that $\mathbf{S}_n = \mathcal{S}_{\mathcal{P}}$. Note that the lower bound follows from Definition 4 where $\mathcal{R}_{\mathcal{P}}, \mathcal{C}_{\mathcal{P}}$ are defined. □

Continuing with the main proof, in order to relate the notion of tiles to the rank-one contribution supports, we need first to define the notion of sub-tiles, where each entry of a sub-tile is a matrix coordinate. We also recall that $\mathcal{R}_{\mathcal{P}}$ and $\mathcal{C}_{\mathcal{P}}$ are respectively the row and column indices of tile $\mathcal{P}$, as given in Definition 4, as well as note that we here regard $\mathcal{R}_{\mathcal{P}}$ and $\mathcal{C}_{\mathcal{P}}$ as arbitrarily ordered sets.

Definition 7. For each tile $\mathcal{P}$, the set of (at most) Δ horizontal sub-tiles takes the form

$$\mathcal{H}_{\mathcal{P},h_{\mathcal{P}}} \triangleq \{(\mathcal{R}_{\mathcal{P}}(h_{\mathcal{P}}), j) | j \in \mathcal{C}_{\mathcal{P}},\ h_{\mathcal{P}} \in [\Delta]\} \quad (59)$$

while the set of (at most) Γ vertical sub-tiles takes the form

$$\mathcal{V}_{\mathcal{P},v_{\mathcal{P}}} \triangleq \{(i, \mathcal{C}_{\mathcal{P}}(v_{\mathcal{P}})) | i \in \mathcal{R}_{\mathcal{P}},\ v_{\mathcal{P}} \in [\Gamma]\}. \quad (60)$$

In the above, $\mathcal{R}_{\mathcal{P}}(h_{\mathcal{P}})$ represents the $h_{\mathcal{P}}$-th element of $\mathcal{R}_{\mathcal{P}}$, and similarly $\mathcal{C}_{\mathcal{P}}(v_{\mathcal{P}})$ represents the $v_{\mathcal{P}}$-th element of $\mathcal{C}_{\mathcal{P}}$. We also note (cf. Lemma 4) that each horizontal (resp. vertical) sub-tile can have at most Γ (resp. Δ) elements. We also need the following function definition.

Definition 8. For $\mathcal{G}$ being the power set of all horizontal and vertical sub-tiles $\{\mathcal{H}_{\mathcal{P},h_{\mathcal{P}}}, \mathcal{V}_{\mathcal{P},v_{\mathcal{P}}}\}_{\mathcal{P}\in\mathcal{C},h_{\mathcal{P}}\in[\Delta],v_{\mathcal{P}}\in[\Gamma]}$, we define the function $\Phi(.) : \{0,1\}^{K\times L} \to \mathcal{G}$ as

$$\Phi(\mathbf{S}_{\mathcal{P}}) \triangleq \begin{cases} \{\mathcal{H}_{\mathcal{P},h_{\mathcal{P}}} | h_{\mathcal{P}} \in [\Delta]\}, & \text{if } |\mathcal{R}_{\mathcal{P}}| \leq |\mathcal{C}_{\mathcal{P}}|, \\ \{\mathcal{V}_{\mathcal{P},v_{\mathcal{P}}} | v_{\mathcal{P}} \in [\Gamma]\}, & \text{if } |\mathcal{R}_{\mathcal{P}}| > |\mathcal{C}_{\mathcal{P}}|. \end{cases} \quad (61)$$

We now proceed to bound the number of rank-one contribution supports, and we do so under our previously stated assumption of disjoint supports (cf. Definition 1), which is equivalent to disjoint tiles assumptions via the following Lemma,

Lemma 5. For two matrices $\mathbf{D}, \mathbf{E}$, the representative supports $\{\mathbf{S}_{\mathcal{P}_i}\}_{i=1}^m$ of $\mathbf{DE}$ are disjoint (i.e., $\mathbf{S}_{\mathcal{P}_i} \cap \mathbf{S}_{\mathcal{P}_j} = \mathbf{0},\ j \neq i$) if and only if $\mathbf{D}$ and $\mathbf{E}$ accept the disjoint support assumption of Definition 1.

Proof. Assuming that $\mathbf{D} \in \mathbb{R}^{K\times NT}, \mathbf{E} \in \mathbb{R}^{NT\times L}$ abide by the disjoint support assumption from Definition 1, then for all $i, i' \in [NT]$, we have that either $\text{Supp}(\mathbf{D}(:,i)\mathbf{E}(i,:)) = \text{Supp}(\mathbf{D}(:,i')\mathbf{E}(i',:))$ or that $\text{Supp}(\mathbf{D}(:,i)\mathbf{E}(i,:)) \cap \text{Supp}(\mathbf{D}(:,i')\mathbf{E}(i',:)) = \emptyset$. This in turn implies that for $\mathbf{I} = \text{Supp}(\mathbf{D}) \in \{0,1\}^{K\times NT}, \mathbf{J} = \text{Supp}(\mathbf{E}) \in \{0,1\}^{NT\times L}$, then either $\mathbf{I}(:,i)\mathbf{J}(i,:) = \mathbf{I}(:,j)\mathbf{J}(j,:)$ or $\mathbf{I}(:,i)\mathbf{J}(i,:) \cap \mathbf{I}(:,j)\mathbf{J}(j,:) = \mathbf{0}_{K\times L}$, which in turn yields the assumption in Definition 3 of disjoint representative support equivalence classes.

In reverse, if $\mathbf{DE}$ accepts the disjoint representative support assumption, and if $\mathcal{C} = \{\mathcal{P}_1, \ldots, \mathcal{P}_m\}$ is the collection of the equivalence classes, then $\forall \mathcal{P}, \mathcal{P}' \in \mathcal{C}, \mathcal{C}_{\mathcal{P}} = \mathcal{C}_{\mathcal{P}'}$ or $\mathcal{C}_{\mathcal{P}} \cap \mathcal{C}_{\mathcal{P}'} = \emptyset$ and similarly $\forall \mathcal{P}, \mathcal{P}' \in \mathcal{C}, \mathcal{R}_{\mathcal{P}} = \mathcal{R}_{\mathcal{P}'}$ or $\mathcal{R}_{\mathcal{P}} \cap \mathcal{R}_{\mathcal{P}'} = \emptyset$, which in turn implies that $\forall i, i' \in [NT]$ then $\text{supp}(\mathbf{D}(:,i)) = \text{supp}(\mathbf{D}(:,i'))$ or $\text{supp}(\mathbf{D}(:,i)) \cap \text{supp}(\mathbf{D}(:,i')) = \emptyset$, as well implies that $\forall i, i' \in [NT]$ then $\text{supp}(\mathbf{E}(i,:)) = \text{supp}(\mathbf{E}(i',:))$ or $\text{supp}(\mathbf{E}(i,:)) \cap \text{supp}(\mathbf{E}(i',:)) = \emptyset$. Consequently, if $\text{supp}(\mathbf{D}(:,i)) = \text{supp}(\mathbf{D}(:,i'))$ and $\text{supp}(\mathbf{E}(i,:)) = \text{supp}(\mathbf{E}(i,:))$ both hold, then $\text{supp}(\mathbf{D}(:,i)\mathbf{E}(i,:)) = \text{supp}(\mathbf{D}(:,i')\mathbf{E}(i',:))$ or other wise $\text{supp}(\mathbf{D}(:,i)\mathbf{E}(i,:)) \cap \text{supp}(\mathbf{D}(:,i')\mathbf{E}(i',:)) = \emptyset$, which in turn yields the assumption in Definition 1 that $\mathbf{D}$ and $\mathbf{E}$ satisfy the disjoint support assumption. □

We proceed with the next lemma.

Lemma 6. In any lossless function reconstruction scheme corresponding to $\mathbf{DE} = \mathbf{F}$, the number of rank-one contribution supports $|\mathcal{P}|$ in each class $\mathcal{P}$, satisfies $|\mathcal{P}| \geq |\Phi(\mathbf{S}_{\mathcal{P}})|$.

Proof. Recall from Definition 1 and Lemma 5 that in the context of lossless schemes, each representative support is disjoint. Then we can see that

$$\begin{aligned} |\mathcal{P}| &\geq \min(\min(|\mathcal{R}_{\mathcal{P}}|, |\mathcal{P}|), \min(|\mathcal{C}_{\mathcal{P}}|, |\mathcal{P}|)) \\ &\overset{(a)}{=} \min(\text{rank}(\mathbf{D}_{\mathcal{P}}), \text{rank}(\mathbf{E}_{\mathcal{P}})) \\ &\overset{(b)}{\geq} \text{rank}(\mathbf{F}_{\mathcal{P}}) \\ &\overset{(c)}{=} \min(|\mathcal{R}_{\mathcal{P}}|, |\mathcal{C}_{\mathcal{P}}|) = r_{\mathcal{P}} \end{aligned}$$

where (a) follows from Definition 1 and Lemma 5 which tells us that in the context of lossless schemes then each representative support is disjoint which in turn tells us that (48) applies in which case we have $\mathbf{D}(\mathcal{R}_{\mathcal{P}}, \mathcal{P}) = \mathbf{D}_{\mathcal{P}}$ and $\mathbf{E}(\mathcal{P}, \mathcal{C}_{\mathcal{P}}) = \mathbf{E}_{\mathcal{P}}$. Subsequently (b) follows from the fact that

$\mathbf{D}_{\mathcal{P}}\mathbf{E}_{\mathcal{P}} = \mathbf{F}_{\mathcal{P}}$, (c) follows from the dimensionality of $\mathbf{F}_{\mathcal{P}}$, and the last equality holds from the definition of $r_{\mathcal{P}}$.

□

We now proceed with a lemma that lower bounds the minimum number of horizontal or vertical sub-tiles needed[1] to cover $\mathbf{F} \in \mathbb{R}^{K\times L}$.

Lemma 7. For any single-shot lossless function reconstruction scheme, and for the corresponding $\mathbf{DE} = \mathbf{F}$ decomposition, the minimum number of sub-tiles needed to cover $\mathbf{F}$ is at least $\frac{KL}{\max(\Delta,\Gamma)}$.

Proof. Suppose first that $\Gamma \geq \Delta$ and consider a scheme that covers the entire $\mathbf{F}$, with m_1 horizontal sub-tiles and m_2 vertical sub-tiles. We wish to show that $m_1 + m_2 \geq \frac{KL}{\max(\Delta,\Gamma)}$. To see this, we first note that since there is no intersection between each of the tiles (cf. Definition 1 and Lemma 5), and since there is no intersection between each sub-tile inside a tile (Definition 8), then we can conclude that for any $\mathcal{P}, \mathcal{P}' \in \mathcal{C}$ and any $\mathcal{S} \in \Phi(\mathbf{S}_{\mathcal{P}}), \mathcal{S}' \in \Phi(\mathbf{S}_{\mathcal{P}'})$, we must have $\mathcal{S} \cap \mathcal{S}' = \emptyset$. This in turn implies that the sub-tiles can now cover at most $m_1\Gamma + m_2\Delta$ elements of $\mathbf{F}$, which in turn means that $m_1\Gamma + m_2\Delta \geq KL$, which means that $\frac{KL}{\Gamma} - m_2\frac{\Delta}{\Gamma} \leq m_1$, which means that $\frac{KL}{\Gamma} + (1-\frac{\Delta}{\Gamma})m_2 \leq m_1 + m_2$. Since $\frac{\Delta}{\Gamma} \leq 1$, we have that $\frac{KL}{\Gamma} \leq \frac{KL}{\Gamma} + (1-\frac{\Delta}{\Gamma})m_2$, which directly tells us that $m_1 + m_2 \geq \frac{KL}{\Gamma}$. This concludes the proof for the case of $\Gamma \geq \Delta$. The same process follows directly also for the case of $\Gamma \leq \Delta$, thus concluding the proof. □

At this point we can combine our results. We know from Definitions 7,8 and from Lemma 7 that $\sum_{\mathcal{P}\in\mathcal{C}} |\Phi(\mathbf{S}_{\mathcal{P}})| > \frac{KL}{\max(\Gamma,\Delta))}$, while we know from Lemma 8 that $\sum_{\mathcal{P}\in\mathcal{C}} |\mathcal{P}| \geq \sum_{\mathcal{P}\in\mathcal{C}} |\Phi(\mathbf{S}_{\mathcal{P}})|$. Now by recalling that $\sum_{\mathcal{P}\in\mathcal{C}} |\mathcal{P}|$ is the number of rank-one contribution supports, and by recalling from Lemma 2 that each rank-one contribution support corresponds to a server in any lossless scheme, we can use the lower bound on the number of sub-tiles in Lemma 7 as a lower bound on the number of servers, allowing us to thus conclude that $N_{opt} \geq \frac{KL}{\max(\Delta,\Gamma)}$, thus concluding the proof of our converse for the single-shot case.

D. Example

Example 1. Consider a single-shot scenario with $K = 7$ users, $L = 11$ subfunctions, $\Delta = 3$ and $\Gamma = 5$, where naturally $\mathbf{F} \in \mathbb{R}^{7\times 11}$. Let us go through the steps described in Subsection B3.

1) *Sizes and positions of tiles of* $\mathbf{DE}$*:* To assign the sizes and locations of the tiles (i.e., of the families of the equivalence classes of rank-one support) of $\mathbf{DE}$, we follow Equations (41)–(44), which yield the tessellation pattern shown in Figure 4. This pattern, which we show to be optimal, entails four tile families $\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3, \mathcal{C}_4$, of respective sizes $3\times5, 3\times1, 1\times5$ and 1×1. Each family has the following number of tiles $|\mathcal{C}_1| = \lfloor\frac{K}{\Delta}\rfloor\lfloor\frac{L}{\Gamma}\rfloor = 2\times 2 = 4, |\mathcal{C}_2| = \lfloor\frac{K}{\Delta}\rfloor = 2, |\mathcal{C}_3| = \lfloor\frac{L}{\Gamma}\rfloor = 2, |\mathcal{C}_4| = 1$ and the tiles have a maximum rank (Cf. Definition 6) equal to $r_{\mathcal{P}} = 3$ for $\mathcal{P} \in \mathcal{C}_1$, $r_{\mathcal{P}} = 1$ for $\mathcal{P} \in \mathcal{C}_2$, $r_{\mathcal{P}} = 1$ for $\mathcal{P} \in \mathcal{C}_3$, and $r_{\mathcal{P}} = 1$ for $\mathcal{P} \in \mathcal{C}_4$. We note that the pattern successfully covers $\mathbf{F}$, which is naturally a necessary condition.
2) *Creating and filling the non-zero tiles of in* $\mathbf{D}$ *and* $\mathbf{E}$*:* The master node extracts the submatrices corresponding to each of the tiles as described in (47), and proceeds to perform 4 complete SVD decompositions to get $\mathbf{D}_{\mathcal{P}}$ and $\mathbf{E}_{\mathcal{P}}$, $\forall\mathcal{P} \in \mathcal{C}_1$, as described in (48).
3) *Placing the filled cropped tiles* $\mathbf{D}_{\mathcal{P}}$ *and* $\mathbf{E}_{\mathcal{P}}$ *in* $\mathbf{D}$ *and* $\mathbf{E}$*:* Finally we place the tiles of $\mathbf{D}$ and $\mathbf{E}$, by following exactly the coordinates described in (52) and (53), as we see in Figure 5.

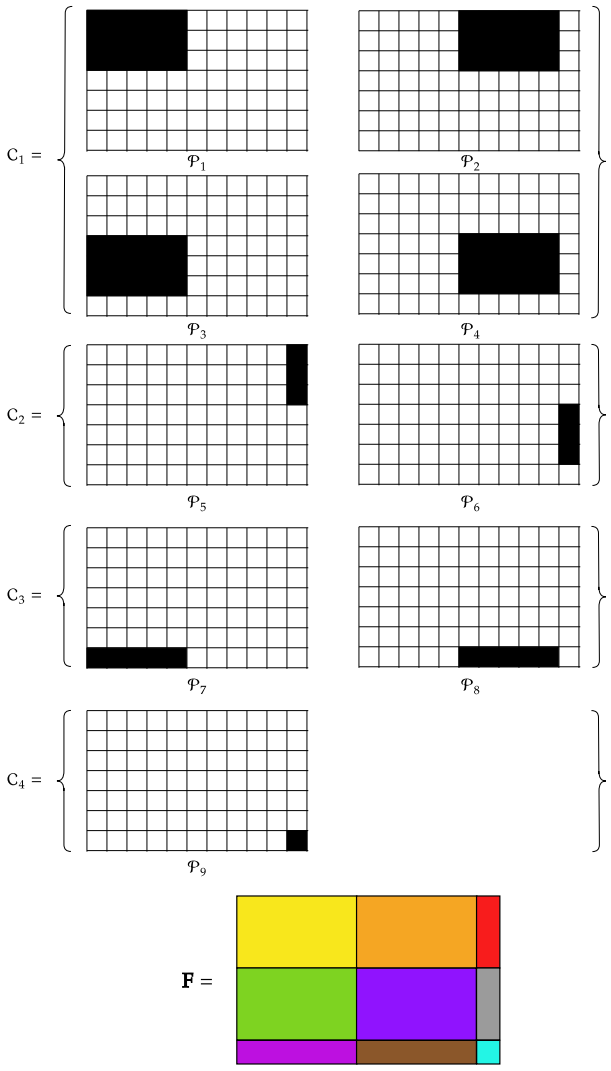

Fig. 4. Corresponding to Example 1, the figure (left) represents in black the families of the equivalent classes from (41)–(44). The 9 equivalent classes (right) cover the entire $\mathbf{F}$.

[1]To "cover" in this context means that every element of $\mathbf{F}$ has to be in at least one representative support.

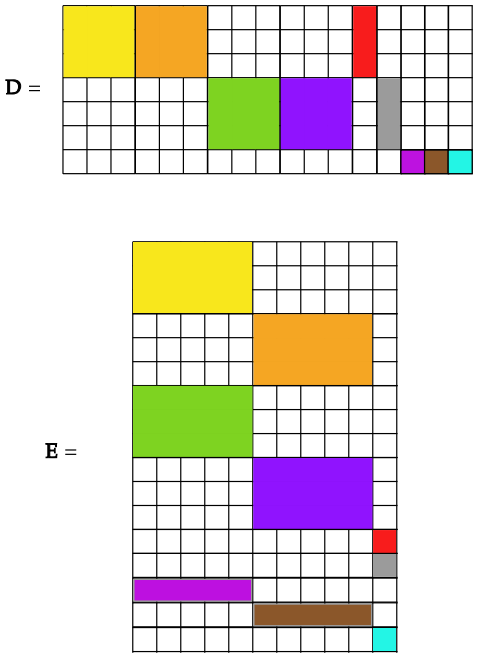

Fig. 5. Creating our communication and computing matrices $\mathbf{D}$ and $\mathbf{E}$, at the exact coordinates from (50)–(53).