Perfect Multi-User Distributed Computing

Ali Khalesi and Petros Elia

***Abstract*—In this paper, we investigate the problem of multi-user linearly decomposable function computation, where N servers help compute functions for K users, and where each such function can be expressed as a linear combination of L basis subfunctions. The process begins with each server computing some of the subfunctions, then broadcasting a linear combination of its computed outputs to a selected group of users, and finally having each user linearly combine its received data to recover its function. As it has become recently known, this problem can be translated into a matrix decomposition problem $\mathbf{F} = \mathbf{DE}$, where $\mathbf{F} \in \mathbf{GF}(q)^{K\times L}$ describes the coefficients that define the users' demands, where $\mathbf{E} \in \mathbf{GF}(q)^{N\times L}$ describes which subfunction each server computes and how it combines the computed outputs, and where $\mathbf{D} \in \mathbf{GF}(q)^{K\times N}$ describes which servers each user receives data from and how it combines this data.**

Our interest here is in reducing the total number of subfunction computations across the servers (cumulative computational cost), as well as the worst-case load which can be a measure of computational delay. Our contribution consists of novel bounds on the two computing costs, where these bounds are linked here to the covering and packing radius of classical codes. One of our findings is that in certain cases, our distributed computing problem — and by extension our matrix decomposition problem — is treated optimally when $\mathbf{F}$ is decomposed into a parity check matrix $\mathbf{D}$ of a perfect code, and a matrix $\mathbf{E}$ which has columns as the coset leaders of this same code.

***Keywords*—Distributed computing, Linearly decomposable functions, Perfect codes, Distributed gradient coding, Coded distributed computing.**

I. Introduction

Distributed computing systems such as MapReduce [1] and Spark [2] form the backbone for processing computationally hard functions in a variety of real-life applications such as scientific simulations, gaming, as well as training large-scale machine learning algorithms and deep neural networks with high data complexity (cf. [3]). For many such applications, distributed parallel processing techniques become crucial for offloading computations to a group of distributed servers in order to reduce the computation time.

It is the case though that this desired parallelization to the distributed servers, constantly presents new challenges that inspire various works, within the disciplines of information theory and coding theory, seeking to design algorithms and fundamental performance limits. Such research can be found of computational accuracy [4]–[7], latency and straggler mitigation [8]–[11], scalability [12]–[15], security and privacy [16]–[20], as well as in the context of communication and computation complexity [21]–[30].

Motivated by the same need to efficiently parallelize multiple computational tasks, our work here studies the known multi-user linearly decomposable scenario introduced in [32], [33], which can be seen as a multi-user extension to [34], and which entails a master node that acts as a total trusted authority in managing N server nodes, serving K users that each demand their function to be computed. Under the linearly-decomposable assumption, where each function is a linear combination of L basis subfunctions, it was recently shown (cf. [32], [33]) that the multi-user distributed computing problem is mathematically equivalent to a sparse matrix factorization problem of the form $\mathbf{F} = \mathbf{DE}$, where $\mathbf{F} \in \mathbf{GF}(q)^{K\times L}$ describes the linear coefficients of the demands of the users, where the so-called computation and encoding matrix $\mathbf{E} \in \mathbf{GF}(q)^{N\times L}$ describes which subfunctions each server evaluates and how each server combines the subfunctions' outputs, while the so-called communication and decoding matrix $\mathbf{D} \in \mathbf{GF}(q)^{K\times N}$ describes the server-to-user activated connectivity and the manner with which each user combines its received data.

This work here studies the above scenario by jointly considering both the cumulative computational cost across the servers, as well as (under a uniformity assumption in the computational delay of evaluating each subfunction) the worst-case computational load which, as we will discuss later, captures the concept of computational delay. Our main contribution is to connect the above two metrics of our distributed computing problem, to the coding-theoretic concepts of the covering radius and the packing radius respectively. Then, in terms of the construction of schemes that efficiently resolve our distributed computing problem, our work provides a never-before-seen connection between distributed computing and the powerful structure of perfect codes. Deviating from the approach in [32] which uses covering codes to reduce the computation cost in asymptotic settings, we here show how perfect codes — which optimize both the covering radius and packing density of codes — yield an improved solution to our distributed computing problem, both in terms of cumulative as well as worst-case costs, and do so for finite dimensions. To the best of our understanding, this is the first time that perfect codes (and the closely related quasi-perfect codes) have been associated with distributed computing and the equivalent problem of matrix factorization. The derived novel bounds on the cumulative

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).

computational cost as well as on the computational delay of a multi-user linearly-decomposable system capture the importance of the packing density as well as the packing and covering radius (defined in [35]) of a code whose parity-check matrix is our communication-and-computing matrix $\mathbf{D}$. .

Paper Organization and Notation: Section II introduces the system model, Section III formulates the problem, Section IV presents the main results, and finally, Section V concludes. In terms of notation, for some $n \in \mathbb{N}$, we define $[n] \triangleq \{1, 2, \ldots, n\}$, while for some vector $\mathbf{x}$ or matrix $\mathbf{X}$, we will use $\omega(\mathbf{x})$ (resp. $\omega(\mathbf{X})$) to denote the corresponding Hamming weight. Furthermore, we will use the short-hand notation $\mathbb{F}$ to represent a finite field $\mathbf{GF}(q)$. For some vector $\mathbf{x} \in \mathbb{F}^n$ and some code $\mathcal{C} \subseteq \mathbb{F}^n$, we will use $d(\mathbf{x}, \mathcal{C})$ to represent the Hamming distance of $\mathbf{x}$ to the nearest codeword in $\mathcal{C}$. For some matrix $\mathbf{H}$, we will use $\mathcal{C}_{\mathbf{H}}$ to represent the linear code whose parity-check matrix is $\mathbf{H}$. Similarly, when we write $\mathbf{H}_{\mathcal{C}}$, we will refer to the parity-check matrix of a linear code $\mathcal{C}$. For some $k \leq n$, $k, n \in \mathbb{N}$, we will also use $\mathcal{C}(k, n)$ to represent a linear code of message length k and codeword length n. Finally, we will use $\text{supp}(\mathbf{x})$ to represent the support of some vector $\mathbf{x} \in \mathbb{F}^n$, corresponding to the set of indices of non-zero elements of that vector.

II. System Model

We consider the multi-user linearly-decomposable distributed computation setting (cf. Fig. 1), which consists of K users/clients, N active (non-idle) servers, and a master node that coordinates servers and users. The process starts with each user asking for a (generally non-linear) function from a space of linearly decomposable functions, where each such function takes several subfunctions as input[1]. In particular, each desired function can be decomposed into a different linear combination of individual subfunctions $f_l(.)$, and thus the demanded function F_k of each user $k \in [K]$, takes the general linearly-decomposable form

$$\begin{aligned} F_k(.) &\triangleq f_{k,1} f_1(.) + f_{k,2} f_2(.) + \ldots + f_{k,L} f_L(.), \;\; k \in [K] \\ &= f_{k,1} W_1 + f_{k,2} W_2 + \ldots + f_{k,L} W_L, \;\; k \in [K] \end{aligned} \tag{1}$$

where $W_l = f_l(.) \in \mathbb{F}$, $l \in [L]$ is a so-called 'file' output, and $f_{k,l} \in \mathbb{F}$, $k \in [K], l \in [L]$ are the corresponding linear combination coefficients.

A. Three Phases of the Distributed Computing Problem

The model involves three phases, with the first being the *demand phase*, followed by the *computation and encoding phase*, and then the *communication and decoding phase*. In the demand phase, each user $k \in [K]$ sends the information of its desired function $F_k(.)$ to the master node, which then evaluates the linear decomposition of each function as in (1). Then based on these K desired functions, during the computation and encoding phase, the master assigns some of the subfunctions

[1]We here implicitly emphasize on scenarios where the computation of these subfunctions far outweighs and subsequent encoding and decoding operations

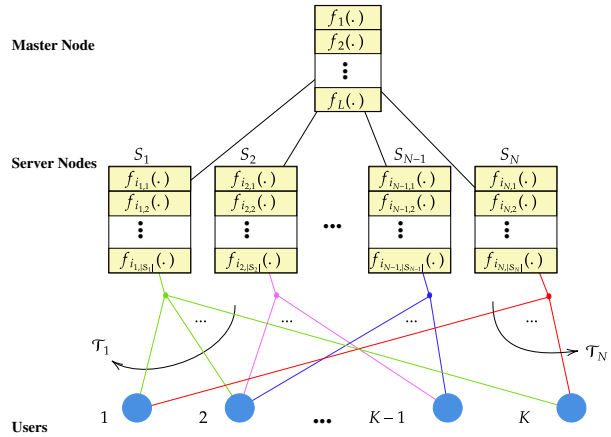

Fig. 1. The K-user, N-server, linearly-decomposable computation setting. After each user informs the master of its desired function $F_k(.)$, each server $n \in [N]$ computes a subfunction $W_l = f_l(.)$ in $\mathcal{S}_n \subset [L]$. Afterwards, server n broadcasts a linear combination z_n (of the locally available computed files) to all users in $\mathcal{T}_n \subseteq [K]$. This combination is defined by the coefficients $e_{n,l}$. Finally, based on some decoding-coefficient vectors (based on some decoding-coefficient vectors, to be described later on), each user $k \in [K]$ linearly combines (based on decoding vectors $\mathbf{d}_k$) all the received signals. Decoding must produce for each user its desired function $F_k(.)$.

to each server, who then proceeds to compute these and produce the corresponding files $W_l = f_l(.)$. In particular, each subfunction $f_l(.), l \in \mathcal{S}_n \subseteq [L]$ will be assigned to server n to be computed, for some carefully selected subset $\mathcal{S}_n \subseteq [L]$.

During the communication and decoding phase, each server $n \in [N]$ broadcasts in one shot, its own linear combination of the locally computed output files, to a specific[2] subset of users $\mathcal{T}_n$. In particular, each server n transmits

$$z_n \triangleq \sum_{l \in [L]} e_{n,l} W_l, \;\; n \in [N] \tag{2}$$

where the so-called encoding coefficients $e_{n,l} \in \mathbb{F}$ are determined by the master. Finally during the decoding part, each user k linearly combines the received signals as follows

$$F'_k \triangleq \sum_{n \in [N]} d_{k,n} z_n \tag{3}$$

for some decoding coefficients $d_{k,n} \in \mathbb{F}, n \in [N]$, determined by the master node. Naturally $d_{k,n} = 0, \forall k \notin \mathcal{T}_n$. The functions are successfully evaluated when $F'_k = F_k, \forall k \in [K]$.

B. Cumulative Computation Cost and Computational Delay

Recalling that $|\mathcal{S}_n|$ indicates the number of subfunctions that server n computes, we consider the *Cumulative Computation Cost* to take the form

$$\Gamma \triangleq \sum_{n=1}^{N} |\mathcal{S}_n| \tag{4}$$

[2]We here emphasize that the activated topology is not fixed before the demand phase, and that indeed the choice of receiving users for each server a function of the demanded jobs.

representing the cumulative number of sub-function computations across all servers.

Furthermore, assuming that the jobs at each server are computed sequentially — and under a uniformity assumption that evaluating each file $W_l, l \in [L]$ from the subfunction $f_l(.)$ requires a normalized unit of time — we may consider a server's computational delay $T_n = |\mathcal{S}_n|$

and thus, under some basic synchronization assumptions, we may consider an overall computational delay that takes the form

$$\Lambda \triangleq \max |\mathcal{S}_n|. \tag{5}$$

We wish to provide schemes that correctly compute the desired functions, with reduced costs Γ and Λ.

III. Problem Formulation: One-Shot Setting

To formulate the problem, we use

$$\mathbf{f} \triangleq [F_1, F_2, \dots, F_K]^\intercal, \tag{6}$$
$$\mathbf{f}_k \triangleq [f_{k,1}, f_{k,2}, \dots, f_{k,L}]^\intercal,\ k \in [K], \tag{7}$$
$$\mathbf{w} \triangleq [W_1, W_2, \dots, W_L]^\intercal \tag{8}$$

where $\mathbf{f}$ represents the vector of the output demanded functions (cf. (1)), $\mathbf{f}_k$ the vector of function coefficients for user k (cf. (1)), and $\mathbf{w}$ the vector of output files. We also have

$$\mathbf{e}_n \triangleq [e_{n,1}, e_{n,2}, \dots, e_{n,L}]^\intercal,\ n \in [N], \tag{9}$$
$$\mathbf{z} \triangleq [z_1, z_2, \dots, z_N]^\intercal \tag{10}$$

respectively representing the encoding vector at server n, and the overall transmitted vector across all the servers (cf. (2)). Furthermore, we have

$$\mathbf{d}_k \triangleq [d_{k,1}, d_{k,2}, \dots, d_{k,N}]^\intercal,\ k \in [K], \tag{11}$$
$$\mathbf{f}' \triangleq [F'_1, F'_2, \dots, F'_K]^\intercal \tag{12}$$

respectively representing the decoding vector at user k, and the vector of the decoded functions across all the users. In addition, we have

$$\mathbf{F} \triangleq [\mathbf{f}_1, \mathbf{f}_2, \dots, \mathbf{f}_K]^\intercal \in \mathbb{F}^{K \times L}, \tag{13}$$
$$\mathbf{E} \triangleq [\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_N]^\intercal \in \mathbb{F}^{N \times L}, \tag{14}$$
$$\mathbf{D} \triangleq [\mathbf{d}_1, \mathbf{d}_2, \dots, \mathbf{d}_K]^\intercal \in \mathbb{F}^{K \times N} \tag{15}$$

where $\mathbf{F}$ represents the $K \times L$ matrix of all function coefficients across all the users, where $\mathbf{E}$ represents the $N \times L$ *computation and encoding matrix* across all the servers, and where $\mathbf{D}$ represents the $K \times N$ *communication and decoding matrix* across all the users. We will henceforth assume that the columns of $\mathbf{F}$ are different from each other.

Directly from (1), we have that

$$\mathbf{f} = [\mathbf{f}_1, \mathbf{f}_2, \dots, \mathbf{f}_K]^\intercal \mathbf{w} \tag{16}$$

and from (2) we have the overall transmitted vector to be

$$\mathbf{z} = [\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_N]^\intercal \mathbf{w} = \mathbf{E}\mathbf{w}. \tag{17}$$

Furthermore, directly from (3) we have that

$$F'_k = \mathbf{d}_k^T \mathbf{z} \tag{18}$$

and thus we have

$$\mathbf{f}' = [\mathbf{d}_1, \mathbf{d}_2, \dots, \mathbf{d}_K]^\intercal \mathbf{z} = \mathbf{D}\mathbf{z}. \tag{19}$$

Recall that we must guarantee that

$$\mathbf{f}' = \mathbf{f}. \tag{20}$$

After substituting (16), (17) and (19) into (20), we see that the feasibility condition in (20) is satisfied if and only if

$$\mathbf{D}\mathbf{E}\mathbf{w} = \mathbf{F}\mathbf{w}. \tag{21}$$

For this to hold for any $\mathbf{w}$, we must thus guarantee

$$\mathbf{D}\mathbf{E} = \mathbf{F}. \tag{22}$$

At this point, we note that $\mathcal{S}_n = \text{supp}(\mathbf{E}(n,:))$ and $|\mathcal{S}_n| = \omega(\mathbf{E}(n,:))$, which gives

$$\Lambda = \max_{n \in [N]} |\mathcal{S}_n| = \max_{n \in [N]} \omega(\mathbf{E}(n,:)) \tag{23}$$

revealing how the computational delay Λ of our system is captured by the maximum number of non-zero elements of any row of $\mathbf{E}$. This is one of the two sparsity constraints that interest us. The other sparsity constraint can be seen by recalling that $|\mathcal{S}_n| = \omega(\mathbf{E}(n,:))$ which gives

$$\Gamma = \sum_{n=1}^{N} |\mathcal{S}_n| = \omega(\mathbf{E}) \tag{24}$$

representing the total number of non-zero elements of $\mathbf{E}$, and which relates naturally to the cumulative computational cost across all servers. Thus, being able to decompose $\mathbf{F}$ as $\mathbf{F} = \mathbf{D}\mathbf{E}$ where $\mathbf{E}$ corresponds to a reduced Λ and Γ, allows us indeed to reduce the two computation costs. To do this, we will turn to perfect codes, as well as to quasi-perfect codes.

Note that whenever we say the multi-user linearly-decomposable is implemented based on the decomposition $\mathbf{D}\mathbf{E} = \mathbf{F}$, it means that based on the dimensionality K, N, we pick a code and choose its parity-check matrix as our $\mathbf{D}$, and then perform a syndrome decoding algorithm by regarding each column of $\mathbf{F}$ as a syndrome and each column of $\mathbf{E}$ as its corresponding error vector.

Before doing so, let us here provide a simple example to help clarify the setting and the notation.

A. Example

We consider the example of a system with a master node, $N = 11$ servers, $K = 5$ users, $L = 12$ subfunctions, and a field of size $q = 3$. In our example, the jobs are defined (cf. (13)) by a demand matrix that here takes the form

$$\mathbf{F} = \begin{bmatrix} 2 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 2 & 1 & 2 & 0 \\ 1 & 0 & 0 & 2 & 2 & 2 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 2 & 1 & 0 & 1 & 0 & 2 & 1 & 2 & 0 & 1 & 1 \\ 0 & 2 & 0 & 2 & 0 & 1 & 2 & 1 & 0 & 1 & 2 & 1 \\ 0 & 0 & 0 & 1 & 1 & 2 & 2 & 0 & 1 & 1 & 1 & 2 \end{bmatrix}.$$

In the computation and encoding phase, the master allocates the computation of the different subfunctions $f_1(.), f_2(.), \ldots, f_{12}(.)$ across the 11 servers according to

$$\mathcal{S}_1 = \{1, 8, 10\},\ \mathcal{S}_2 = \{2, 6\},\ \mathcal{S}_3 = \{5, 9\},$$
$$\mathcal{S}_4 = \{4\},\ \mathcal{S}_5 = \{7, 11\},\ \mathcal{S}_6 = \{1, 6, 12\},\ \mathcal{S}_7 = \{3\},$$
$$\mathcal{S}_8 = \{2\},\ \mathcal{S}_9 = \{3, 10\},\ \mathcal{S}_{10} = \{7, 12\},\ \mathcal{S}_{11} = \{5, 8\}$$

forcing the two costs to be $\Gamma = \sum_{n=1}^{N} |\mathcal{S}_n| = 21$ and $\Lambda = \max_{n\in[N]} |\mathcal{S}_n| = 3$.

After computing their designated output files, each server n transmits z_n corresponding to a computation and encoding matrix (cf. (17)) which here takes the value

$$\mathbf{E} = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \end{bmatrix}$$

and which indeed abides by the constraint $\max_{n\in[N]} \omega(\mathbf{E}(n,:)) = \Lambda = 3$ (cf. (23)) and the constraint $\omega(\mathbf{E}) = \Gamma = 21$ (cf. (24)).

Subsequently, the master asks each server n to send its generated z_n to its designated receiving users, where for each server these user-sets are

$$\mathcal{T}_1 = \{1, 2, \ldots, 5\},\ \mathcal{T}_2 = \{1, 2, \ldots, 4\},$$
$$\mathcal{T}_3 = \{1, 2, 3, 5\},\ \mathcal{T}_4 = \{1, 2, 4, 5\},$$
$$\mathcal{T}_5 = \{1, 3, 4, 5\},\ \mathcal{T}_6 = \{2, 3, 4, 5\},\ \mathcal{T}_7 = \{1\}, \mathcal{T}_8 = \{2\},$$
$$\mathcal{T}_9 = \{3\},\ \mathcal{T}_{10} = \{4\}, \mathcal{T}_{11} = \{5\}$$

which simply says that, for example, server 2 will broadcast z_2 to users $1, 2, 3, 4$. Subsequently, the decoding procedure is executed, adhering to a decoding matrix which takes the value

$$\mathbf{D} = \begin{bmatrix} 1 & 1 & 1 & 2 & 2 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 2 & 1 & 0 & 2 & 0 & 1 & 0 & 0 & 0 \\ 1 & 2 & 1 & 0 & 1 & 2 & 0 & 0 & 1 & 0 & 0 \\ 1 & 2 & 0 & 1 & 2 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 2 & 2 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \quad (25)$$

In the next section, we will show that the decoding matrix $\mathbf{D}$, is drawn from a class of perfect codes whose properties — as we show below — allow for an improved performance.

IV. Main Results

We proceed with the main results of our work.

Theorem 1. *The optimal computational delay Λ of the (K, N) multi-user linearly decomposable problem implemented based on the decomposition $\mathbf{DE} = \mathbf{F}$, is bounded as*

$$\Lambda \leq \min\{L, \sum_{i=1}^{\tau} \binom{N-1}{i-1}(q-1)^i + (1-\mu_\tau)q^K\} \quad (26)$$

where τ and μ_τ are respectively the packing radius and the corresponding packing density of $\mathcal{C}_{\mathbf{D}}$.

Proof. Let us rewrite (22) as

$$\mathbf{DE}(:, l) = \mathbf{F}(:, l),\ \forall l \in [L] \quad (27)$$

and let us note that given a certain $\mathbf{D}$, we ask that for every column $\mathbf{F}(:, l) \in \mathbb{F}^K$, the resulting $\mathbf{E}(:, l)$ has a reduced number of non-zero elements. To achieve this, after regarding $\mathbf{D}$ to be a parity-check matrix of a linear code, we employ a syndrome decoder, rendering the 'error pattern' $\mathbf{E}(:, l)$ to be a coset leader associated to syndrome $\mathbf{F}(:, l)$. Now let's define the following set

$$\mathcal{S} \triangleq \{l : \omega(\mathbf{E}(:, l)) \leq \tau\} \quad (28)$$

which represents the syndromes $\mathbf{F}(:, l)$ for which the weight of their associated coset leader is below the packing radius τ. From the basic error-correcting argument, we know that all the error patterns having no more than than τ non-zero elements are indeed present in $\mathcal{S}$, and thus we have that $|\mathcal{S}| \leq \sum_{i=1}^{\tau} \binom{N}{i}(q-1)^i$. Let us now focus on a row subvector $\mathbf{E}(n, \mathcal{S})$, and let us count the number of its non-zero elements. Since we know that for all $l \in \mathcal{S}$ it is indeed the case that $\omega(\mathbf{E}(:, l)) \leq \tau$, and since the collection of all such columns $\mathbf{E}(:, l)$ contains all possible error patterns of weight up to τ, we can conclude that

$$\omega(\mathbf{E}(n, \mathcal{S})) \leq \sum_{i=1}^{\tau} \binom{N-1}{i-1}(q-1)^i, \quad \forall n \in [N] \quad (29)$$

because i) each element $\mathbf{E}(n, l) \neq 0, l \in \mathcal{S}$ can take any one of $(q-1)$ possible values, because ii) such a column vector $\mathbf{E}(:, l)$ (where again $l \in \mathcal{S}$) has at most τ such non-zero elements, and because iii) there exist at most $\binom{N-1}{i-1}(q-1)^{i-1}$ vectors $\mathbf{E}(:, l) \neq 0, l \in \mathcal{S}$ whose nth entry $\mathbf{E}(n, l)$ is non-zero and whose total weight is $i \leq \tau$. Summing across all $i = 1, 2, ..., \tau$ yields the first term in our bound in (29). To conclude the proof, we proceed to count the number of syndromes (columns $\mathbf{F}(:, l)$ of $\mathbf{F}$) whose coset leaders (referring to the corresponding column $\mathbf{E}(:, l)$ of $\mathbf{E}$) have weight strictly bigger than τ. To do so, we first note that the number of points that are not covered by the ball $\mathcal{B}(\mathbf{c}, \tau)$, $\mathbf{c} \in \mathcal{C}_{\mathbf{D}}$, is $(1-\mu_\tau)q^N$. Recalling that each coset corresponds to q^{N-K} points (vectors) in $\mathbf{GF}(q)^N$, we can conclude that the number of cosets in question is $(1-\mu_\tau)q^K$. These are the cosets corresponding to syndromes with indices from $\mathcal{S}' \triangleq \{l : \omega(\mathbf{E}(:, l)) > \tau\}$. Thus, for the worst-case scenario of interest, we can consider that for all l for which $\omega(\mathbf{E}(:, l)) > \tau$, it is the case that we will encounter a non-zero $\mathbf{E}(n, l)$ for any $n \in [N]$, which is reflected in the addition of the second term $(1 - \mu_\tau)q^K$ in our bound. Naturally, L is a trivial upper bound on Λ. This concludes the proof of the theorem. □

We now proceed with the second result, this time regarding the cumulative computation cost.

Theorem 2. *The optimal cumulative computation cost Γ of the (K, N) multi-user linearly decomposable problem implemented based on the decomposition $\mathbf{DE} = \mathbf{F}$, is bounded as*

$$\Gamma \leq \min\{NL, \sum_{i=1}^{\tau} \binom{N}{i}(q-1)^i i + (1-\mu_\tau)q^K \rho\} \quad (30)$$

where τ, ρ and μ_τ are respectively the packing radius, covering radius, and packing density of $\mathcal{C}_{\mathbf{D}}$.

Proof. The proof follows the steps of the proof of Theorem 1, up until the definition in (28), where similarly we now know that all the error patterns having no more than τ non-zero elements are present in the set of all possible $\mathbf{F}(:, l),\ l \in \mathcal{S}$, and thus we know that the sum of the Hamming weights of the coset leaders corresponding to the syndromes $\mathbf{F}(:, l),\ l \in \mathcal{S}$, takes the form $\sum_{i=1}^{\tau} \binom{N}{i} i(q-1)^i$ which matches the first term of our bound. Regarding the second term, we know that for any $l \notin \mathcal{S}$, the number of non-zero elements in $\mathbf{E}(:, l)$ is — by definition of the covering radius — no greater than ρ. We also see, following the ball arguments in the proof of the previous theorem, that the set $\mathcal{S}' \triangleq \{l : \omega(\mathbf{E}(:, l)) > \tau\}$ has size $|\mathcal{S}'| = (1-\mu_\tau)q^K$. Hence, we can now conclude that the sum of the weights of the coset leaders of the vectors (seen as syndromes) in $\mathcal{S}'$, is upper bounded by $\rho|\mathcal{S}'|$. □

A. The Connection to Perfect and Quasi-Perfect Codes

At this point, we note that the above bound on Γ is reduced when the covering radius ρ is reduced, while the bound on Λ is reduced when, for any given τ, the packing density μ_τ is increased. For this reason we are looking for codes (whose parity check matrix will be used as the decoding matrix $\mathbf{D}$) that indeed minimize ρ and increase μ_τ for a fixed τ. This special and rare property sought in $\mathcal{C}_{\mathbf{D}}$ is indeed attributed to the well-known class of perfect codes [35].

It is though known (cf. [36]) that there exist few such perfect codes, for a few select dimensions. Thus, for other dimensions, we will resort to the use of quasi-perfect codes which indeed ensure a similar performance, by guaranteeing optimal ρ and near-optimal μ_τ. We elaborate on this later on.

Remark 1. Looking back to our previous example corresponding to $N = 11, K = 5, L = 12$ and $q = 3$, the distributed computing solutions employed an $\mathbf{E}$ matrix that resulted from an optimal syndrome decoding process of a code whose parity check matrix $\mathbf{D}$ is indeed that of a ternary Golay code, which is a known perfect code.

B. The Special Case of Maximal Basis Set

We here consider also the case where $L = q^K$. This particular case of having a maximum number of basis subfunctions presents interesting advantages. In essence, under the same assumptions as before, we are now able to set the decoding matrix $\mathbf{D}$ once, well before the desired functions are declared. This allows us to have a fixed network connectivity, where — under the assumption of knowing in advance K, N — we can fix $\mathbf{D}$, and we can then account for each $\mathbf{F}$ simply by altering $\mathbf{E}$. For this setting, we have the following propositions.

Proposition 1. *The optimal computational delay Λ and cumulative computation cost Γ of the (K, N) multi-user linearly decomposable problem with maximal basis, are lower bounded as*

$$\Lambda \geq \sum_{i=1}^{\tau} \binom{N-1}{i-1}(q-1)^i, \quad \Gamma \geq \sum_{i=1}^{\tau} \binom{N}{i}(q-1)^i i \quad (31)$$

where τ is the packing radius of $\mathcal{C}_{\mathbf{D}}$.

Proof. Regarding the bound on Λ, we follow all the steps of the proof in Theorem 1, with the only differences being firstly that the used inequality $|\mathcal{S}| \leq \sum_{i=1}^{\tau} \binom{N}{i}(q-1)^i$ is now automatically forced into an equality, and secondly that we consider, by choice, a lower bound $|\mathcal{S}'| \geq 0$. Exactly the same approach applies for the bound on Γ. □

Regarding optimality, we have the following proposition.

Proposition 2. *The optimal computational costs Λ and Γ for the cases (K, N) for which a perfect code exists, take the form*

$$\Lambda = \sum_{i=1}^{\tau} \binom{N-1}{i-1}(q-1)^i, \quad \Gamma = \sum_{i=1}^{\tau} \binom{N}{i}(q-1)^i i \quad (32)$$

where τ is the packing radius of the used perfect code $\mathcal{C}_{\mathbf{D}}$.

Proof. For the case where (K, N) accepts a perfect code, we know (cf. [35]) that $\mu_\tau = 1$ for which the upper bounds in (26) and (30) match the corresponding upper bounds in (31). □

We also have the following proposition for a much broader range of dimensionalities.

Proposition 3. *For all cases (K, N) for which there exists a quasi-perfect code $\mathcal{C}_{\mathbf{D}}$, the optimal performance is upper bounded as $\Lambda < \sum_{i=1}^{\tau} \binom{N-1}{i-1}(q-1)^i + \binom{N}{\tau+1}(q-1)^{\tau+1}$ and $\Gamma < \sum_{i=1}^{\tau+1} \binom{N}{i}(q-1)^i i$, where τ is the packing radius of the corresponding quasi-perfect code $\mathcal{C}_{\mathbf{D}}$.*

Proof. The proof is direct by noting that quasi-perfect codes have $\mu_\tau > 1 - \binom{N}{\tau+1}(q-1)^{\tau+1}q^{-K}$ and $\tau = \rho - 1$ [35]. □

Remark 2. Given the non-existence results in [36], it is in fact not difficult to show that quasi-perfect codes minimize the gaps between the upper bounds in (26) and (30) and the corresponding lower bounds in (31).

V. Conclusion

We have explored the computational cost of the multi-user distributed computing setting of linearly decomposable functions, which nicely captures various problems such as the distributed gradient coding problem [11], the distributed linear transform problem [37], the distributed matrix multiplication problem, and the distributed multivariate polynomial computation problems [38], [39], among others. The work established various upper and lower bounds on the computational delay Λ and the cumulative computation cost $\Gamma \in [0, NL]$, and revealed new connections to the packing and covering capabilities of codes thus revealing for the first time powerful connections with the class of perfect codes.

REFERENCES

[1] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[2] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets," in *2nd USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 10)*, 2010.

[3] 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.

[4] T. Jahani-Nezhad and M. A. Maddah-Ali, "Codedsketch: A coding scheme for distributed computation of approximated matrix multiplication," *IEEE Transactions on Information Theory*, vol. 67, no. 6, pp. 4185–4196, 2021.

[5] J. Wang, Z. Jia, and S. A. Jafar, "Price of precision in coded distributed matrix multiplication: A dimensional analysis," in *2021 IEEE Information Theory Workshop (ITW)*, pp. 1–6, IEEE, 2021.

[6] E. Ozfatura, S. Ulukus, and D. Gündüz, "Coded distributed computing with partial recovery," *IEEE Transactions on Information Theory*, vol. 68, no. 3, pp. 1945–1959, 2022.

[7] K. Wan, H. Sun, M. Ji, D. Tuninetti, and G. Caire, "Cache-aided matrix multiplication retrieval," *IEEE Transactions on Information Theory*, vol. 68, no. 7, pp. 4301–4319, 2022.

[8] 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.

[9] A. Reisizadeh, S. Prakash, R. Pedarsani, and A. S. Avestimehr, "Tree gradient coding," in *2019 IEEE International Symposium on Information Theory (ISIT)*, pp. 2808–2812, IEEE, 2019.

[10] 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.

[11] 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.

[12] S. Li, S. M. M. Kalan, A. S. Avestimehr, and M. Soltanolkotabi, "Near-optimal straggler mitigation for distributed gradient methods," in *2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pp. 857–866, IEEE, 2018.

[13] 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.

[14] E. Parrinello, E. Lampiris, and P. Elia, "Coded distributed computing with node cooperation substantially increases speedup factors," in *2018 IEEE International Symposium on Information Theory (ISIT)*, pp. 1291–1295, 2018.

[15] S. Li, M. A. Maddah-Ali, Q. Yu, and A. S. Avestimehr, "A fundamental tradeoff between computation and communication in distributed computing," *IEEE Transactions on Information Theory*, vol. 64, no. 1, pp. 109–128, 2018.

[16] M. Soleymani and H. Mahdavifar, "Distributed multi-user secret sharing," *IEEE Transactions on Information Theory*, vol. 67, no. 1, pp. 164–178, 2020.

[17] A. Khalesi, M. Mirmohseni, and M. A. Maddah-Ali, "The capacity region of distributed multi-user secret sharing," *IEEE Journal on Selected Areas in Information Theory*, vol. 2, no. 3, pp. 1057–1071, 2021.

[18] H. Akbari-Nodehi and M. A. Maddah-Ali, "Secure coded multi-party computation for massive matrix operations," *IEEE Transactions on Information Theory*, vol. 67, no. 4, pp. 2379–2398, 2021.

[19] H. A. Nodehi and M. A. Maddah-Ali, "Limited-sharing multi-party computation for massive matrix operations," in *2018 IEEE International Symposium on Information Theory (ISIT)*, pp. 1231–1235, IEEE, 2018.

[20] 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.

[21] A. R. Elkordy, S. Li, M. A. Maddah-Ali, and A. S. Avestimehr, "Compressed coded distributed computing," *IEEE Transactions on Communications*, vol. 69, no. 5, pp. 2773–2783, 2021.

[22] Q. Yu, S. Li, M. A. Maddah-Ali, and A. S. Avestimehr, "How to optimally allocate resources for coded distributed computing?," in *2017 IEEE International Conference on Communications (ICC)*, pp. 1–7, IEEE, 2017.

[23] Q. Yu, M. Maddah-Ali, and S. Avestimehr, "Polynomial codes: an optimal design for high-dimensional coded matrix multiplication," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[24] W. Li, Z. Chen, Z. Wang, S. A. Jafar, and H. Jafarkhani, "Flexible distributed matrix multiplication," *IEEE Transactions on Information Theory*, vol. 68, no. 11, pp. 7500–7514, 2022.

[25] M. V. Jamali, M. Soleymani, and H. Mahdavifar, "Coded distributed computing: Performance limits and code designs," in *2019 IEEE Information Theory Workshop (ITW)*, pp. 1–5, IEEE, 2019.

[26] G. Suh, K. Lee, and C. Suh, "Matrix sparsification for coded matrix multiplication," in *2017 55th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 1271–1278, IEEE, 2017.

[27] 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.

[28] F. Brunero, K. Wan, G. Caire, and P. Elia, "Coded distributed computing for sparse functions with structured support," in *2023 IEEE Information Theory Workshop (ITW)*, pp. 474–479, 2023.

[29] E. Parrinello, E. Lampiris, and P. Elia, "Coded distributed computing with node cooperation substantially increases speedup factors," in *2018 IEEE International Symposium on Information Theory (ISIT)*, pp. 1291–1295, 2018.

[30] F. Brunero and P. Elia, "Multi-access distributed computing," *arXiv preprint arXiv:2206.12851*, 2022.

[31] S. Ulukus, S. Avestimehr, M. Gastpar, S. A. Jafar, R. Tandon, and C. Tian, "Private retrieval, computing, and learning: Recent progress and future challenges," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 3, pp. 729–748, 2022.

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

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

[34] 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.

[35] T. Etzion, *Perfect Codes and Related Structures*. WORLD SCIENTIFIC, 2022.

[36] A. Tietäväinen, "On the nonexistence of perfect codes over finite fields," *SIAM Journal on Applied Mathematics*, vol. 24, no. 1, pp. 88–96, 1973.

[37] S. Dutta, V. Cadambe, and P. Grover, ""short-dot": Computing large linear transforms distributedly using coded short dot products," *IEEE Transactions on Information Theory*, vol. 65, no. 10, pp. 6171–6193, 2019.

[38] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Straggler mitigation in distributed matrix multiplication: Fundamental limits and optimal coding," *IEEE Transactions on Information Theory*, vol. 66, no. 3, pp. 1920–1933, 2020.

[39] 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.