Order Optimal Task Allocation in Distributed Computing via Interweaved Cliques

Javad Maheri, K. K. Krishnan Namboodiri, and Petros Elia
EURECOM, Sophia Antipolis, France
Email: {maheri, karakkad, elia}@eurecom.fr

***Abstract*—THIS PAPER IS ELIGIBLE FOR THE STUDENT PAPER AWARD. We consider a distributed computing system in which a master node coordinates N workers to evaluate a function over n input files, where this function accepts general decomposition. In particular, we focus on the general case where the requested function admits a d-uniform decomposition, meaning that it can be decomposed into a set of subfunctions that each depends on a unique d-tuple of the n files. Our objective is to design file and task allocations that minimize the worst-case communication from the master to any worker and the worst-case computational load across workers. We first show that the optimal file and task allocation with minimum communication and computation costs admits a natural characterization within combinatorial design theory: it corresponds to a Steiner system $S(t, k, v)$ with $t = d$, $v = n$, and $k \approx \frac{n}{N^{1/d}}$. However, Steiner systems are known to exist only for very restricted parameter regimes. To overcome this limitation, we propose the information-theoretic-inspired *Interweaved Clique (IC) design*, a universal and deterministic allocation framework that relaxes the strict structure of Steiner systems by allowing slight variations in worker file loads. Although slightly suboptimal, the IC design achieves a communication cost within a constant factor $4e$ from our converse, while also maintaining an order-optimal computation cost, thus allowing this work to derive the fundamental scaling laws of this general distributed computing problem for a large range of parameters.**

***Index Terms*—Distributed Computing, Coded Distributed Computing, Coded Caching, Distributed Learning, Communication Optimization, Map Reduce, Hypergraph Partitioning, Combinatorial Designs, t-Designs, Steiner Systems.**

I. Introduction

The efficient allocation of computational tasks and data is a cornerstone of modern distributed computing, caching, and distributed learning frameworks [1]–[5]. Across applications such as large-scale machine learning, covariance matrix estimation, and scientific simulation, system performance is often constrained by the volume of communication required during distributed execution and by the associated computational burden. This challenge has motivated extensive recent work on communication- and computation-efficient distributed function evaluation. A prominent line of research studies coded distributed computing frameworks, beginning with Coded MapReduce [6] and extending to variants that address stragglers, heterogeneity, and network topology [7]–[14], demonstrating that structured data placement and task allocation can substantially reduce communication through coded exchanges. Another set of works focuses on linearly separable functions and straggler resilience [15], [16], multi-user architectures and task assignment using covering and tiling constructions [17], [18], and worst-case communication minimization under task constraints [19]. All the above lines of research share the goal of designing, under various settings and assumptions, task and data assignment methods that reduce communication and computation costs in distributed computing, and in certain settings, highlight inherent interactions between these two resources.

Motivated by the same need for efficient task and data allocation, we here consider a general coded distributed computing framework for computing decomposable functions in distributed systems. In our framework, the function is decomposable into multiple subfunctions, each taking as input a different d-tuple of files, allowing the master node to assign collections of subfunctions (each represented here by a d-tuple) to multiple workers and communicates the necessary file inputs so as to enable local computation. Naturally, this setting entails a communication cost (as servers need to be communicated the necessary files), and a computation cost (which scales with the number of subfunctions each server must compute). The central design problem is to jointly determine the task assignment and file placement strategies that minimize communication while maintaining balanced computational loads across workers. Unlike formulations tailored to specific computation pipelines, our framework models the distributed evaluation of general d-tuple decomposable functions by explicitly characterizing the interaction between task partitioning and file placement. The resulting problem is inherently combinatorial in nature. Accordingly, we seek solutions based on structured combinatorial constructions that minimize both communication and computation costs.

Indeed, combinatorial designs have been widely used in prior coded distributed computing frameworks, particularly in MapReduce-based models, where clique covers based on t-designs and related combinatorial structures guide task and data assignment [20]–[24]. While these approaches have been effective in reducing communication, they are largely specialized to MapReduce-style computation pipelines. In contrast, our work develops combinatorial constructions for a more

This work was supported by the Huawei France-funded Chair towards Future Wireless Networks, by the French government under the France 2030 ANR program "PEPR Networks of the Future" (ref. ANR-22-PEFT-0010), and by European Research Council ERC-StG Project SENSIBILITÉ under Grant 101077361.

general d-tuple subfunction model, enabling distributed computation beyond the MapReduce paradigm.

Notations: We represent d-tuples using bold lowercase letters, such as $\mathbf{a} = \{a_1, a_2, \ldots, a_d\}$. Sets of d-tuples are denoted by bold uppercase letters, such as $\mathbf{\Phi}$. We use $[n]$ to denote the set $\{1, 2, \ldots, n\}$. For any set $\mathcal{S}$, $\binom{\mathcal{S}}{d}$ denotes the set of all d-sized subsets of $\mathcal{S}$. We use $\mathbf{A}_{n,d}$ to denote $\binom{[n]}{d}$. For positive integers a and b, $a \mid b$ indicates that a divides b. Finally, we write $f(n) \asymp g(n)$ if there exist constants c_1, c_2, and n_0 such that for all $n \geq n_0$, $c_1 g(n) \leq f(n) \leq c_2 g(n)$.

II. System Model and Problem Statement

We consider a distributed computing system consisting of N worker nodes (servers) and a master node that coordinates the computation of a desired function of n input files (as shown in Fig. 1). The master has access to a library of files $\mathcal{W} = \{W_1, \ldots, W_n\}$, where each file $W_j \in \mathbb{F}^B$ contains B symbols over a field $\mathbb{F}$. The desired function $F : (\mathbb{F}^B)^n \to \mathbb{F}^L$ is assumed to be decomposable into $\binom{n}{d}$ subfunctions, each depending on a unique subset of d files. Accordingly, any such decomposition can be written as

$$\Psi\big(\{\zeta_{\mathcal{T}}(\mathcal{W}_{\mathcal{T}}) : \mathcal{T} \in \mathbf{A}_{n,d}\}\big) : (\mathbb{F}^T)^{\binom{n}{d}} \to \mathbb{F}^L \tag{1}$$

where Ψ is an aggregation function, $\mathbf{A}_{n,d} = \binom{[n]}{d}$, and each subfunction $\zeta_{\mathcal{T}} : (\mathbb{F}^B)^d \to \mathbb{F}^T$ operates on the set of files $\mathcal{W}_{\mathcal{T}} = \{W_j : j \in \mathcal{T}\}$. The parameter d is referred to as the *subfunction file degree*.

Task and File Allocation: The master assigns to each worker $b \in [N]$ a set of subfunctions $\mathbf{\Phi}_b \subseteq \mathbf{A}_{n,d}$ to compute. The collection $\mathbb{P} \triangleq \{\mathbf{\Phi}_1, \ldots, \mathbf{\Phi}_N\}$ forms a partition of $\mathbf{A}_{n,d}$,

$$\bigcup_{b=1}^{N} \mathbf{\Phi}_b = \mathbf{A}_{n,d}, \qquad \mathbf{\Phi}_b \cap \mathbf{\Phi}_{b'} = \emptyset \ \text{ for } b \neq b'. \tag{2}$$

To compute the subfunctions in $\mathbf{\Phi}_b$, worker b must receive all files indexed by the union of the d-tuples in $\mathbf{\Phi}_b$. Let

$$\alpha(\mathbf{\Phi}_b) = \bigcup\nolimits_{\mathcal{T} \in \mathbf{\Phi}_b} \mathcal{T} \subseteq [n] \tag{3}$$

denote the set of required file indices. The master sends the set of files $\mathcal{W}^{(b)} = \alpha(\mathbf{\Phi}_b)$ to worker b. The communication cost is defined as

$$\pi = \max_{b \in [N]} |\alpha(\mathbf{\Phi}_b)| \tag{4}$$

which captures the bottleneck communication load across workers.

Computing Phase: During the computing phase, each worker b computes all subfunctions $\zeta_{\mathcal{T}}(\mathcal{W}_{\mathcal{T}})$ for $\mathcal{T} \in \mathbf{\Phi}_b$. Assuming identical computational capability across workers and equal cost per subfunction, the computation time is proportional to the maximum number of subfunctions assigned to any worker. Accordingly, the computation cost is defined as

$$\delta = \frac{\max_{b \in [N]} |\mathbf{\Phi}_b|}{\lceil |\binom{n}{d}|/N \rceil} \tag{5}$$

where the denominator corresponds to the ideal uniform assignment achieving the minimum possible computation delay.

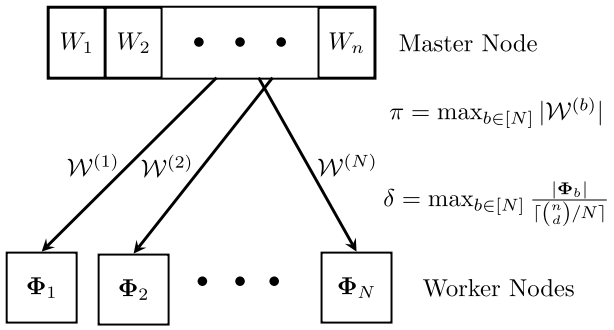

Fig. 1. Distributed computing model. The set of files communicated to worker b is denoted with $\mathcal{W}^{(b)} = \alpha(\mathbf{\Phi}_b)$, while π denotes the maximum communication cost across the master–worker links under the assumption of parallel, uniform-capacity links. Similarly, the set of subfunctions assigned for computation to worker b is denoted by $\mathbf{\Phi}_b$, while δ denotes the computational delay normalized by the minimum possible computational delay, assuming homogeneous workers.

We define $\pi^\star$ as the minimum communication cost over all valid allocation schemes. Our objective is to design a partition $\mathbb{P} = \{\mathbf{\Phi}_b\}_{b=1}^N$ that minimizes the communication cost π while ensuring that the computation cost δ remains close to unity, for given n, d, and N.

Our approach, and specifically the d-tuple based decomposability model we proposed, is motivated by the many practically relevant functions that require aggregating interactions over a large fraction of subfunctions - and which thus entail interactions the d-tuples in $\mathbf{A}_{n,d} = \binom{[n]}{d}$, the collection of all subsets of $[n]$ of size d. Canonical examples include covariance and correlation computations involving all pairwise dependencies ($d = 2$) [25], higher-order cumulant estimation with $d > 2$ interactions [26], and kernel matrix construction in kernel methods, which requires evaluating similarities over all pairs of data points [27]. Similar dense interaction patterns arise in particle and molecular dynamics simulations, where forces are computed over all particle pairs or higher-order groups [28], [29], as well as in exhaustive SNP–SNP interaction analysis in genomics [30]. In all these settings, the final output depends on a very large collection of d-tuple subfunctions, making scalable distributed computation a central challenge.

A. The Combinatorial Perspective: Steiner Systems

The set of all d-tuples (the subfunctions) can be represented as the edge set of a complete d-uniform hypergraph on vertex set $[n]$. Consequently, assigning subfunctions to N servers is equivalent to partitioning the hyperedges of this complete hypergraph into N groups. From combinatorial design theory, one natural solution to this hyperedge partitioning problem is a *Steiner system*, denoted by $S(t, k, v)$.

Definition 1 (Steiner System): A Steiner system $S(t, k, v)$ consists of a set of v points and a collection of blocks, each of size k, such that every subset of t points is contained in exactly one block.

Directly from the definition, the following lemma follows.

Lemma 1: Let $S(t, k, v)$ be a Steiner system. Consider a distributed computing setting with $n = v$ files and subfunction

degree $d = t$, and let $N = \binom{n}{d}/\binom{k}{d}$. Then there exists an assignment of files and subfunctions to N workers such that each worker is communicated with k files and is assigned exactly $\binom{k}{d}$ subfunctions. Consequently, the communication cost $\pi = k$ and the computation cost $\delta = 1$ are achievable.

Proof: Let $\mathcal{S}_1, \ldots, \mathcal{S}_N$ denote the blocks of $S(d, k, n)$. Assign worker b the subfunction set $\mathbf{\Phi}_b = \binom{\mathcal{S}_b}{d}$, and communicate to it the k files indexed by $\mathcal{S}_b$. Since every d-subset of $[n]$ is contained in exactly one block, the sets $\{\mathbf{\Phi}_b\}_{b=1}^{N}$ form a partition of $\mathbf{A}_{n,d}$, and each worker is assigned $\binom{k}{d}$ subfunctions. Hence, we have $\pi = k$ and $\delta = 1$. ∎

However, the applicability of Steiner systems is severely limited by their stringent existence requirements. A Steiner system $S(d, \pi, n)$ can exist only if the divisibility conditions that $\binom{n-i}{d-i}$ is divisible by $\binom{\pi-i}{d-i}$ for all $0 \leq i \leq d-1$ are satisfied, and even when these necessary conditions hold, existence is guaranteed only for restricted parameter regimes. Consequently, for most choices of n, d, and N, no Steiner system exists. This strong structural rigidity makes Steiner systems unsuitable as a general design tool for distributed computing systems in which n or N may vary freely.

III. Main Results

To overcome the non-existence of Steiner systems for most parameter regimes, we propose the *Interweaved Clique (IC) design*, a relaxed combinatorial framework that provides a constructive and broadly applicable solution for a wide range of system parameters (n, N, d). The relaxation allows a slight variation in the number of tasks assigned to each worker, while preserving a deterministic structure based on interweaved cliques. Unlike prior appearances of clique-based constructions in information-theoretic problems such as coded caching [1], where cliques represent user side-information structures, in this work cliques serve as the combinatorial seed for constructing the worker–file–task allocation. Specifically, the global allocation is built over $\mathbf{A}_{n,d}$ using a smaller seed structure $\mathbf{A}_{f,d}$, where f can be much smaller than n. Similar two-level complete-set formulations have also appeared in other coding-theoretic and information-theoretic contexts [11], [31]–[37].

The proposed IC design enjoys the following key properties:

1) ***Universality****:* A valid partition exists for a wide range of parameters (n, N, d).

2) ***Order-Optimality****:* The proposed scheme achieves $\pi \leq 4e\, n/N^{1/d}$ and therefore achieves a gain in communication cost that scales as $N^{1/d}$, where e denotes Euler's constant. Moreover, the scheme is order-optimal, as its communication cost is within a constant factor $4e$ of the information-theoretic lower bound obtained from a packing argument.

Let us now formally state the performance guarantees of the IC design, emphasizing the communication cost π.

Theorem 1 (Optimal Communication Cost): For a distributed computing system with n files and $N \leq (\frac{9}{10}\sqrt{\frac{n}{d}})^d$ workers, the IC design achieves a communication cost π satisfying

$$\pi \leq \frac{4e \cdot n}{N^{1/d}}.$$

Furthermore, the scaling law $\pi \asymp \frac{n}{N^{1/d}}$ is optimal.

Proof: The proof of Theorem 1 follows from the IC design presented in Section IV and a converse on $\pi^\star$ presented in Section V, and is provided in Section VI. ∎

In the end, let us here also note that in the extended version of our work in [38], we also show that this same IC design guarantees, in a very broad setting and with high probability, a near-optimal computation cost of $\delta \leq 4$.

IV. Achievable Scheme: Interweaved-Cliques Design

We now describe the Interweaved-Cliques (IC) design, which constructs a partition of $\mathbf{A}_{n,d}$ for any given tuple $(n, d, N \leq (\frac{9}{10}\sqrt{\frac{n}{d}})^d)$. The design leverages an intermediate parameter f to structure the file library into *families* and then assigns tasks based on the intersection of these families. The parameter f be defined as

$$f \triangleq \max\left\{r \in \mathbb{Z}^+ \mid \binom{r}{d} \leq N\right\}. \tag{6}$$

First, we design the partition of $\mathbf{A}_{n,d}$ for an intermediate number of groups N', defined as $N' \triangleq \binom{f}{d}$. Then, we extend the construction from N' to N groups. The construction of $\mathbf{\Phi}_1, \mathbf{\Phi}_2, \ldots, \mathbf{\Phi}_{N'}$ proceeds in two cases.

A. Case 1: Divisible Parameters ($f \mid n$)

Assume n is divisible by f, so $s \triangleq n/f$ is an integer.

1) File Families: We partition the n files into f disjoint sets called families, denoted $\mathcal{F}_1, \ldots, \mathcal{F}_f$, each containing s files. Specifically, $\mathcal{F}_i$ contains files with indices $\{(i-1)s + 1, \ldots, is\}$.

2) Group Identification: We create $N' = \binom{f}{d}$ base groups, indexed by the set of d-subsets of families, i.e., $\sigma \in \binom{[f]}{d}$. We denote the set of all base groups as

$$\mathbf{\Sigma} \triangleq \{\sigma \subseteq [f] : |\sigma| = d\}. \tag{7}$$

Each group $\sigma \in \mathbf{\Sigma}$ is allocated the union of files from the families in σ. The number of files allocated to each base group is thus

$$\pi = \sum_{i \in \sigma} |\mathcal{F}_i| = s \cdot d = \frac{n \cdot d}{f}. \tag{8}$$

3) Task Partitioning: We partition $\mathbf{A}_{n,d}$ by assigning each d-tuple $\mathbf{a} \in \mathbf{A}_{n,d}$ to one of the groups. We define the *support family* of a tuple $\mathbf{a}$ as $\mathcal{B}(\mathbf{a}) \triangleq \{j \in [f] \mid \mathbf{a} \cap \mathcal{F}_j \neq \emptyset\}$.

- **Full Support Tuples:** We let $\mathbf{A}_{\text{full}} \triangleq \{\mathbf{a} \in \mathbf{A}_{n,d} \mid |\mathcal{B}(\mathbf{a})| = d\}$ represent the so-called set of full support (maximal support) d-tuples. If $|\mathcal{B}(\mathbf{a})| = d$, the $\mathbf{a}$ intersects exactly one file from d distinct families. For a d-tuple $\mathbf{a} \in \mathbf{A}_{\text{full}}$, let $\mathcal{B}(\mathbf{a}) = \sigma$. We assign $\mathbf{a}$ to the group σ which belongs to $[N'] = \binom{[f]}{d}$. For each $\sigma \in \binom{[f]}{d}$, we form its full supports members as follows

$$\mathbf{\Phi}_\sigma^{(\text{full})} \triangleq \{\mathbf{a} \in \mathbf{A}_{n,d} \mid \mathcal{B}(\mathbf{a}) = \sigma\}. \tag{9}$$

  This forms the *clique* core of the design.
- **Complement Tuples:** We also let $\mathbf{A}_{\text{com}} \triangleq \mathbf{A}_{n,d} \backslash \mathbf{A}_{\text{full}}$. For each $\mathbf{a} \in \mathbf{A}_{\text{com}}$, $|\mathcal{B}(\mathbf{a})| < d$. This means, this d-tuple is

supported by a subset of families $\mathcal{I} \subset [f]$ with $|\mathcal{I}| < d$. The $\mathbf{a}$ is eligible to be assigned to all $\sigma \in \binom{[f]}{d}$ such that $\mathcal{I} \subset \sigma$. To design partition $\mathbf{A}_{\rm com}$ into N' groups, we consider a partition of $\mathbf{A}_{n,d}$, which classifies its d-tuples according to the size of their support family. This partition is as follows

$$\mathbf{A}_{n,d} = \{\mathbf{C}_{\lceil \frac{d}{s} \rceil}, \mathbf{C}_{\lceil \frac{d}{s} \rceil + 1}, \ldots, \mathbf{C}_d\} \tag{10}$$

where for each $\beta \in [\lceil \frac{d}{s} \rceil, d]$, the set $\mathbf{C}_\beta \triangleq \{\mathbf{a} \in \mathbf{A}_{n,d} \mid |\mathcal{B}(\mathbf{a})| = \beta\}$ represents the set of d-tuples $\mathbf{a} \in \mathbf{C}_\beta$ that each intersects exactly β families. Naturally, we have $\mathbf{C}_d = \mathbf{A}_{\rm full}$ and $\mathbf{A}_{\rm com} = \bigcup_{\beta=\lceil \frac{d}{s} \rceil}^{d-1} \mathbf{C}_\beta$. Let us fix a $\beta \in [\lceil \frac{d}{s} \rceil, d-1]$. Then, for each $\mathcal{I} \in \binom{[f]}{\beta}$, we define

$$\mathbf{C}_{\beta,\mathcal{I}} \triangleq \{\mathbf{a} \in \mathbf{C}_\beta \mid \mathcal{B}(\mathbf{a}) = \mathcal{I}\}. \tag{11}$$

We define $\mathbf{C}_{\beta,\mathcal{I},\sigma}$ as the subset of $\mathbf{C}_{\beta,\mathcal{I}}$ allocated to group σ. In Appendix B.C of [38], we describe a sequence of steps that leads to the construction of the sets $\mathbf{C}_{\beta,\mathcal{I},\sigma}$. For each $\sigma \in \binom{[f]}{d}$, there exist $\binom{d}{\beta}$ distinct $\mathcal{I} \in \binom{[f]}{\beta}$ such that $\mathcal{I} \subset \sigma$. Consequently, $\mathbf{\Phi}_\sigma^{(\rm com)} \triangleq \bigcup_{\beta=\lceil \frac{d}{s} \rceil}^{d-1} \bigcup_{\mathcal{I} \subset \sigma} \mathbf{C}_{\beta,\mathcal{I},\sigma}$. Then, the subfunctions (d-tuples) allocated to worker σ, where $\sigma \in \binom{[f]}{d}$, is

$$\mathbf{\Phi}_\sigma \triangleq \mathbf{\Phi}_\sigma^{(\rm full)} \cup \mathbf{\Phi}_\sigma^{(\rm com)}. \tag{12}$$

B. Case 2: General Parameters ($f \nmid n$)

In this case, we cannot create equal-sized families. We adapt the design by introducing *excluded* files. Let $s_0 \triangleq \lfloor \frac{n}{f+d} \rfloor + 1$ and define $g \triangleq n - f \cdot s_0$. We treat the last g files as an excluded set $\mathcal{E}$, i.e.,

$$\mathcal{E} \triangleq [n]/[n-g] = \{n, n-1, \ldots, n-g+1\}. \tag{13}$$

The remaining $n' \triangleq n - g$ files are partitioned into f families of size s_0. The partition of $\mathbf{A}_{n,d}$ is constructed by

1) Step 1: In this step, we apply the Case IV-A construction to the n' non-excluded files, i.e., $[n']$. Thus, each group $\sigma \in \binom{[f]}{d}$ receives from $\mathbf{A}_{n',d}$

$$\mathbf{\Phi}_\sigma^{(\rm full)} \cup \mathbf{\Phi}_\sigma^{(\rm com)}. \tag{14}$$

2) Step 2: The second step considers the excluded [1] d-tuples

$$\mathbf{A}_{\rm exc} \triangleq \mathbf{A}_{n,d} \setminus \mathbf{A}_{n',d}. \tag{15}$$

Thus, we aim to distribute the $\mathbf{A}_{\rm exc}$ that contain excluded files (from $\mathcal{E}$) into the groups $N' = \binom{f}{d}$ formed by their non-excluded elements (see Section IV-A). Any d-tuple $\mathbf{t} \in \mathbf{A}_{\rm exc}$ will have an arbitrary number $m_{\mathbf{t}} = |\mathbf{t} \cap \mathcal{E}|$ of components/elements from the excluded file-index set $\mathcal{E}$, and it will have $d - m_{\mathbf{t}} = |\mathbf{t} \cap [n']|$ elements from the rest. It is easy to see that $m_{\mathbf{t}} \in [1, \min\{d, g\}]$ and thus that $d - m_{\mathbf{t}} \in [\max\{d-g, 0\}, d-1]$. Whenever there is no ambiguity, we will henceforth revert to the simpler notation m instead of $m_{\mathbf{t}}$. For every $m \in [1, \min\{d, g\}]$, we define the set

$$\mathbf{R}_{m,\beta} \triangleq \{\mathbf{t} \in \mathbf{A}_{\rm exc} \mid |\mathcal{B}(\mathbf{t})| = \beta,\ |\mathbf{t} \cap \mathcal{E}| = m\} \tag{16}$$

which describes the d-tuples $\mathbf{t}$ that intersect exactly β families and contain m excluded elements from $\mathcal{E}$. Notice that β can take values in the range $\left[\lceil \frac{d-m}{s_0} \rceil, d-m\right]$. If $m = d \leq g$, then $\beta = 0$, which means that all the entries of $\mathbf{t}$ are from $\mathcal{E}$. Let us now partition $\mathbf{A}_{\rm exc}$ as follows $\mathbf{A}_{\rm exc} = \bigcup_{m=1}^{\min\{d,g\}} \bigcup_{\beta=\lceil \frac{d-m}{s_0} \rceil}^{d-m} \mathbf{R}_{m,\beta}$. For each $\mathcal{I} \in \binom{[f]}{\beta}$, let us now define

$$\mathbf{R}_{\beta,\mathcal{I}} \triangleq \left\{\mathbf{t} \in \mathbf{A}_{\rm exc} \mid \mathcal{B}(\mathbf{t}) = \mathcal{I},\ \mathbf{t} \in \bigcup_{m=1}^{\min\{d-\beta,g\}} \mathbf{R}_{m,\beta}\right\} \tag{17}$$

to be the set of all d-tuples $\mathbf{t} \in \mathbf{A}_{\rm exc}$ that intersect exactly all families in $\mathcal{I}$, where in the above, $\mathcal{B}(\mathbf{t})$ denotes the set of families that $\mathbf{t}$ intersects. Let us now also define $\mathbf{R}_\beta \triangleq \bigcup_{\mathcal{I} \in \binom{[f]}{\beta}} \mathbf{R}_{\beta,\mathcal{I}} \subset \mathbf{A}_{\rm exc}$ to be the set of all excluded d-tuples that meet exactly β families. Furthermore, directly by applying the established ranges of parameters m and k, we can conclude that the range of $\beta \in [\beta_{\min}, \beta_{\max}]$, is defined by $\beta_{\min} \triangleq \lceil \frac{d - \min\{d,g\}}{s_0} \rceil = \lceil \frac{\max\{0, d-g\}}{s_0} \rceil$ and $\beta_{\max} \triangleq d-1$.

Our next step involves going through the range of β. For each $\beta \in [\beta_{\min}, \beta_{\max}]$, we partition each time the set $\mathbf{R}_\beta$ into $N' = \binom{f}{d}$ groups. This partitioning is described in detail in Appendix B.E of [38]. In particular, let us first recall that each group is labeled by a $\sigma \in \binom{[f]}{d}$. For each such σ, there exist $\binom{d}{\beta}$ different subsets $\mathcal{I} \subset \sigma$ with cardinality β. For each $\mathcal{I} \subset \sigma$, the set $\mathbf{R}_{\beta,\mathcal{I},\sigma}$ collects all d-tuples in $\mathbf{R}_{\beta,\mathcal{I}}$ associated to group σ. We then form the union and define

$$\mathbf{\Phi}_\sigma^{(\rm exc)} \triangleq \bigcup_{\beta=\beta_{\min}}^{\beta_{\max}} \bigcup_{\mathcal{I} \subset \sigma} \mathbf{R}_{\beta,\mathcal{I},\sigma}. \tag{18}$$

Combining (14) and (18), we get the subfunctions (d-tuples) allocated to worker σ, where $\sigma \in \binom{[f]}{d}$, as follows.

$$\mathbf{\Phi}_\sigma \triangleq \mathbf{\Phi}_\sigma^{(\rm full)} \cup \mathbf{\Phi}_\sigma^{(\rm com)} \cup \mathbf{\Phi}_\sigma^{(\rm exc)}. \tag{19}$$

Finally, the partition of $\mathbf{A}_{n,d}$ into N' groups is described by

$$\mathbf{A}_{n,d} = \bigcup_{\sigma \in \binom{[f]}{d}} \mathbf{\Phi}_\sigma. \tag{20}$$

We continue with the following lemma.

Lemma 2: For n, d, the IC design in Case 2 (Section IV-B) achieves

$$\pi \leq s_0 \cdot d + g.$$

Proof: The proof is direct by noting that worker σ receives all files in families σ plus, at worst case, the entire set $\mathcal{E}$. ∎

C. Extension of the Partition from N' Groups to N Groups

Recall (cf. (20)) that we have already partitioned $\mathbf{A}_{n,d}$ into $N' = \binom{f}{d}$ disjoint groups $\mathbf{\Phi}_{\sigma_1}, \ldots, \mathbf{\Phi}_{\sigma_{N'}}$, $\sigma_1, \ldots, \sigma_{N'} \in \binom{[f]}{d}$. We will now redistribute the d-tuples of these N' groups across all existing N groups. Towards this, let us assume that the indices $\sigma_1, \ldots, \sigma_{N'}$ are in lexicographic order and, in order to ease notation, let us rename the corresponding N' groups by their lexicographic position, as follows $\mathbf{\Phi}_1, \ldots, \mathbf{\Phi}_{N'}$ where in particular, $\mathbf{\Phi}_b = \mathbf{\Phi}_{\sigma_b}$ for $b \in [N']$. Recalling that there are $N \geq N'$ actual groups, let us first define the variables

$$q \triangleq \lfloor \frac{N}{N'} \rfloor, \qquad p \triangleq \lceil \frac{N}{N'} \rceil, \qquad r \triangleq N \bmod N' \tag{21}$$

[1] For example, for $n = 5, n' = 4, d = 2$, we have that $\mathbf{A}_{\rm exc} = \{15, 25, 35, 45\}$.

thus noting that $N = qN' + r$, where $p = q$ if $r = 0$, and $p = q+1$ if $r > 0$. At this point, we proceed with the first step of dividing the d-tuple set of each of the first N' groups into different parts, and then with the second step of redistributing some of these parts to fill up the empty $N - N'$ groups.

Step 1 – Dividing the d-tuples of each of the first N' groups: For each $b \in [N']$, we define the number of parts

$$s_b \triangleq \begin{cases} p & \text{if } 1 \le b \le r, \\ q & \text{if } r < b \le N' \end{cases} \tag{22}$$

and we split each $\mathbf{\Phi}_b$ into s_b disjoint sub-parts using lexicographic ordering that yields slicing of equal sizes, plus or minus 1, where we naturally keep track of the exact size of each sub-part. We denote these sub-parts by $\mathbf{\Phi}_b^{(0)}, \mathbf{\Phi}_b^{(1)}, \ldots, \mathbf{\Phi}_b^{(s_b-1)}$, where $\mathbf{\Phi}_b = \bigcup_{b'=0}^{s_b-1} \mathbf{\Phi}_b^{(b')}$.

Step 2 – Extending to N groups: We then relabel these sub-parts to obtain the desired N groups. We define the new N groups $\mathbf{\Phi}_1, \ldots, \mathbf{\Phi}_N$ by the indexing rule

$$\mathbf{\Phi}_{b+b'N'} \triangleq \mathbf{\Phi}_b^{(b')}, \text{ for } b \in [N'],\ b' \in \{0, \ldots, s_b - 1\}. \tag{23}$$

We conclude this section with the following lemma.

Lemma 3: For any n, d, N, the IC design uses $N' = \binom{f}{d}$, where $f = \max\left\{r \in \mathbb{Z}^+ \,\middle|\, \binom{r}{d} \le N\right\}$ (cf. (6)), and guarantees that

$$\frac{N}{N'} < d + 1 \le 2^d.$$

Proof: Directly from the definitions of f and N', we note that $N < \binom{f+1}{d}$ and $N' = \binom{f}{d}$, and thus $\frac{N}{N'} < \frac{\binom{f+1}{d}}{\binom{f}{d}} = \frac{f+1}{f+1-d}$. Since the function $\frac{x}{x-d}$ is decreasing on $[d+1, \infty)$, we conclude that $\frac{N}{N'} \le \max_{f \ge d} \frac{f+1}{f+1-d} \le d + 1 \le 2^d$. ∎

V. A Lower Bound on $\pi^\star$

A converse bound on the communication cost can be derived by observing that a worker with π files can compute at most $\binom{\pi}{d}$ subfunctions. To cover all $\binom{n}{d}$ tasks with N workers, we must have

$$\binom{n}{d} \le \sum_{b=1}^{N} \binom{|\alpha(\mathbf{\Phi}_b)|}{d} \le N \binom{\pi}{d} \tag{24}$$

since $\pi = \max_{b \in [N]} |\alpha(\mathbf{\Phi}_b)|$.

Using the inequality $\frac{\pi - i}{n - i} \le \frac{\pi}{n}$ for $1 \le i \le d-1$ in (24), we obtain the following lower bound on π

$$\pi \ge \frac{n}{N^{1/d}}. \tag{25}$$

For any possible π, the (25) holds. Consequently, for the optimal π, denoted by $\pi^\star$, we have

$$\pi^\star \ge \frac{n}{N^{1/d}}. \tag{26}$$

This lower bound represents the *packing radius* of the hypergraph. While Steiner systems achieve this bound with equality (where π is exactly the block size), their non-existence for most N forces us to seek approximate designs that still respect this $N^{-1/d}$ scaling (please see Appendix B.A in [38] for more details).

VI. Proof of Theorem 1

From the achievable scheme discussed in Section IV-A, we have $\pi = s \cdot d = \frac{n}{f} \cdot d$. Using the simple bound $N' = \binom{f}{d} \le (\frac{e \cdot f}{d})^d$, we can conclude that $f \ge \frac{d}{e} N'^{1/d}$, which directly yields

$$\pi \le \frac{nd}{\frac{d}{e} N'^{1/d}} = \frac{N^{1/d}}{N'^{1/d}} \cdot \frac{nd}{\frac{d}{e} N^{1/d}} = \frac{2e \cdot n}{N^{1/d}} \tag{27}$$

where the last step follows from Lemma 3. Then from Lemma 2, we conclude that $\pi = s_0 \cdot d + g$. Similarly, we can show that

$$s_0 \cdot d \le \frac{2e \cdot n}{N^{1/d}}. \tag{28}$$

This, combined with $g = n - s_0 \cdot f$, directly yields

$$g = n - \left(\left\lfloor \frac{n}{f+d} \right\rfloor + 1 \right) \cdot f \le n - \left\lceil \frac{n}{f+d} \right\rceil \cdot f \tag{29}$$

$$\le n - \frac{n}{f+d} \cdot f = \frac{n \cdot d}{f+d} \tag{30}$$

and since $\frac{n}{f+d} < \left\lfloor \frac{n}{f+d} \right\rfloor + 1 = s_0$, we can directly conclude that $g \le \frac{n \cdot d}{f+d} \le s_0 \cdot d$. Combining this with (28), we get $\pi \le \frac{4e \cdot n}{N^{1/d}}$. Finally, applying the converse in (26) shows that for any $(n, d, N \le (\frac{9}{10}\sqrt{\frac{n}{d}})^d)$, we have $\pi/\pi^\star \le 4e$, and hence the scaling law $\pi \asymp n/N^{1/d}$ is optimal.

VII. Comparison with Steiner Systems

The advantage of the IC design over Steiner systems lies in its flexibility with respect to column sizes. A Steiner system $S(d, \pi, n)$ requires the number of blocks $\binom{n}{d}/\binom{\pi}{d}$ to be exactly equal to N, which severely restricts its applicability. In contrast, the IC design fixes N and n, and then effectively determines the optimal clique size f (and hence π) compatible with the available workers. By allowing file assignments to overlap according to a family-based interlaced clique structure, rather than a rigid block structure, the IC design guarantees the existence of a valid partition for any $N \le (\frac{9}{10}\sqrt{\frac{n}{d}})^d$. As we have discussed, while a Steiner system would achieve $\pi \asymp n/N^{1/d}$ with $\delta = 1$, it may not exist for a given N. The IC design attains the same communication scaling with $\delta \le 4$, offering a practical trade-off that represents a controlled increase in computation imbalance in exchange for universal applicability.

VIII. Conclusion

This paper addressed a fundamental file and task allocation problem in distributed computing. We highlighted the theoretical optimality of Steiner systems while exposing their practical limitations due to sparsity. The proposed Interweaved-Cliques (IC) design was shown to bridge this gap, offering a deterministic and universally applicable allocation scheme. By achieving order-optimal communication cost $\pi \asymp n/N^{1/d}$ and bounded computation balance, the IC design provides a robust solution for deploying large-scale distributed function evaluations without the rigid constraints of classical combinatorial designs.

REFERENCES

[1] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *IEEE Trans. Inf. Theory*, vol. 60, no. 5, pp. 2856–2867, 2014.

[2] K. Wan, D. Tuninetti, and P. Piantanida, "An index coding approach to caching with uncoded cache placement," *IEEE Trans. Inf. Theory*, vol. 66, no. 3, pp. 1318–1332, 2020.

[3] R. Bitar, M. Wootters, and S. El Rouayheb, "Stochastic gradient coding for straggler mitigation in distributed learning," *IEEE J. Sel. Areas Inf. Theory*, vol. 1, no. 1, pp. 277–291, 2020.

[4] R. Bitar, P. Parag, and S. El Rouayheb, "Minimizing latency for secure distributed computing," in *2017 IEEE Int. Symp. Inf. Theory (ISIT)*, 2017, pp. 2900–2904.

[5] V. K. Vavilapalli, A. C. Murthy, C. Douglas, S. Agarwal, M. Konar, R. Evans, T. Graves, J. Lowe, H. Shah, S. Seth *et al.*, "Apache Hadoop yarn: Yet another resource negotiator," in *Proc. of the 4th annu. Symp. Cloud Comput.*, 2013, pp. 1–16.

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

[7] H. Chen, M. Cheng, and Y. Wu, "On the fundamental limits of decentralized linearly separable computation under cyclic assignment," *IEEE Trans. Commun.*, pp. 1–1, 2025.

[8] Y. Wang and Y. Wu, "Coded distributed computing with pre-set data placement and output functions assignment," *IEEE Trans. Inf. Theory*, vol. 71, no. 3, pp. 2195–2217, 2025.

[9] Y. Bi, M. Wigger, and Y. Wu, "Normalized delivery time of wireless MapReduce," *IEEE Trans. Inf. Theory*, vol. 70, no. 10, pp. 7005–7022, 2024.

[10] E. Peter, K. K. K. Namboodiri, and B. S. Rajan, "Wireless MapReduce arrays for coded distributed computing," in *Proc. IEEE Inf. Theory Workshop (ITW)*, 2024, pp. 163–168.

[11] F. Brunero and P. Elia, "Multi-access distributed computing," *IEEE Trans. Inf. Theory*, vol. 70, no. 5, pp. 3385–3398, 2024.

[12] K. Wan, H. Sun, M. Ji, D. Tuninetti, and G. Caire, "On the optimal load-memory tradeoff of cache-aided scalar linear function retrieval," *IEEE Trans. Inf. Theory*, vol. 67, no. 6, pp. 4001–4018, 2021.

[13] Y. Yao and S. A. Jafar, "The capacity of 3 user linear computation broadcast," *IEEE Trans. Inf. Theory*, vol. 70, no. 6, pp. 4414–4438, 2024.

[14] Y. Ma and D. Tuninetti, "An achievable scheme for the k-user linear computation broadcast channel," *arXiv 2501.12322*, 2025.

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

[16] ——, "On the tradeoff between computation and communication costs for distributed linearly separable computation," *IEEE Trans. Commun.*, vol. 69, no. 11, pp. 7390–7405, 2021.

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

[18] ——, "Tessellated distributed computing," *IEEE Trans. Inf. Theory*, vol. 71, no. 6, pp. 4754–4784, 2025.

[19] K. K. K. Namboodiri, E. Peter, D. Malak, and P. Elia, "Fundamental limits of distributed computing for linearly separable functions," *arXiv 2509.23447*, 2025.

[20] M. Cheng, Y. Wu, X. Li, and D. Wu, "Asymptotically optimal coded distributed computing via combinatorial designs," *IEEE/ACM Trans. Networking*, vol. 32, no. 4, pp. 3018–3033, 2024.

[21] J. Jiang, W. Wang, and L. Zhou, "Cascaded coded distributed computing schemes based on symmetric designs," *IEEE Trans. Commun.*, vol. 70, no. 11, pp. 7179–7190, 2022.

[22] S. Agrawal and P. Krishnan, "Low complexity distributed computing via binary matrices with extension to stragglers," in *2020 IEEE Int. Symp. Inf. Theory (ISIT)*, 2020, pp. 162–167.

[23] S. Agrawal, K. V. S. Sree, P. Krishnan, A. Vaishya, and S. Kale, "Cache-aided communication schemes via combinatorial designs and their q-analogs," *IEEE J. Sel. Areas Inf. Theory*, vol. 4, pp. 551–568, 2023.

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

[25] O. Ledoit and M. Wolf, "A well-conditioned estimator for large-dimensional covariance matrices," *J. Multivariate Analysis*, vol. 88, no. 2, pp. 365–411, 2004.

[26] P. Comon, "Independent component analysis, a new concept?" *Signal processing*, vol. 36, no. 3, pp. 287–314, 1994.

[27] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural computation*, vol. 10, no. 5, pp. 1299–1319, 1998.

[28] G. Dhaliwal, P. B. Nair, and C. V. Singh, "Machine learned interatomic potentials using random features," *npj Computational Materials*, vol. 8, no. 1, p. 7, 2022.

[29] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," *Advances neural inf. process. syst.*, vol. 20, p. 1177–1184, 2007.

[30] P. Li, M. Guo, C. Wang, X. Liu, and Q. Zou, "An overview of SNP interactions in genome-wide association studies," *Briefings in functional genomics*, vol. 14, no. 2, pp. 143–155, 2015.

[31] P. N. Muralidhar, D. Katyal, and B. S. Rajan, "Maddah-Ali-Niesen scheme for multi-access coded caching," in *2021 IEEE Inf. Theory Workshop (ITW)*, 2021, pp. 1–6.

[32] F. Brunero and P. Elia, "Fundamental limits of combinatorial multi-access caching," *IEEE Trans. Inf. Theory*, vol. 69, no. 2, pp. 1037–1056, 2023.

[33] E. Parrinello, E. Lampiris, and P. Elia, "Coded distributed computing with node cooperation substantially increases speedup factors," in *2018 IEEE Int. Symp. Inf. Theory (ISIT)*, 2018, pp. 1291–1295.

[34] K. K. K. Namboodiri and B. S. Rajan, "Combinatorial multi-access coded caching: Improved rate-memory trade-off with coded placement," *IEEE Trans. Inf. Theory*, vol. 70, no. 3, pp. 1787–1805, 2024.

[35] E. Peter, K. K. K. Namboodiri, and B. S. Rajan, "Coded caching with shared caches and private caches," *IEEE Trans. Commun.*, vol. 72, no. 8, pp. 4857–4872, 2024.

[36] F. Engelmann and P. Elia, "A content-delivery protocol, exploiting the privacy benefits of coded caching," in *2017 15th Int. Symp. Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, 2017, pp. 1–6.

[37] H. Zhao, A. Bazco-Nogueras, and P. Elia, "Vector coded caching multiplicatively increases the throughput of realistic downlink systems," *IEEE Trans. Wireless Commun.*, vol. 22, no. 4, pp. 2683–2698, 2023.

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