A Linear Approach for \((k + 1, k, k)\) Exact-Regeneration Distributed Storage Codes

Soheil Mohajer, Member, IEEE, Mehran Elyasi, and Ravi Tandon, Member, IEEE

Abstract

Characterizing the exact repair storage-vs-repair bandwidth tradeoff for distributed storage systems remains an open problem for more than five storage nodes. Motivated by the prevalence and practical applicability of linear codes, the exact repair problem when restricted to linear codes is considered. The main result of this paper is a new approach to develop bounds for exact repair distributed storage systems with linear codes (LDSS). Using this approach, the exact repair region for the \((k + 1, k, k)\) LDSS is completely characterized. The new approach utilizes the properties of linear codes in conjunction with the constraints arising from exact repair requirements. These constraints are formally captured through an optimization problem with a recursive structure, and its solution finally yields the new bounds for the LDSS. Moreover, analysis of the structure of the optimum point of the optimization problem, reveals several important properties that an optimal code must posses to achieve such performance. A class of codes are designed by exploiting such properties. The new approach to upper bound the optimum trade-off together with obtained code construction characterizes the exact repair region for \((k + 1, k, k)\) LDSS.

I. INTRODUCTION

Distributed storage systems (DSS) are increasingly being employed by various technologies. While the size of data, number of storage components, and number of users connecting to these servers are dramatically growing, efficiency of the system, in the sense of a fundamental trade-off between the overhead penalty paid to provide robustness and the cost of system maintenance, is becoming the key factor to determine their performance.

In this paper, we focus on distributed storage systems, which can be described via the parameters, \((n, k, d)\) and \((\alpha, \beta)\). The total data must be stored on a total of \(n\) distributed nodes; and must be recoverable by accessing any \(k \leq n\) nodes (also known as the data recovery property of the DSS). Each node can store at most \(\alpha\) units of data (i.e., \(\alpha\) is the per-node storage capacity). As nodes can fail over time (which often happens in practice due to a variety of factors such as hardware failures, disk corruption etc), it is also desirable to design storage systems with...
a self-recovery or regenerating capability. In particular, if a node fails, it should be possible to replace the failed node by a new node by contacting any \( d \) nodes (where \( k \leq d \leq n-1 \)) and downloading a total of \( d\beta \) units of data. Hence, \( d\beta \) refers to the total repair bandwidth (or \( \beta \) refers to the per-node repair bandwidth). Upon regeneration of a failed node, the new system of \( n \) nodes must satisfy the data recovery property along with the regenerating capability. These requirements lead to a fundamental tradeoff between \( \alpha \) (per-node storage) and \( \beta \) (per node repair bandwidth), i.e., repair bandwidth can be reduced at the cost of higher storage and vice-versa. With regards to the regeneration of a failed node, one possibility is to replace it with a new node (albeit with possibly different content than the original failed node), while meeting data recovery and regeneration constraints. Such a repair process is known as functional repair. In a seminal work, Dimakis et.al [2] formulated the above functional repair problem and fully characterized the optimal trade-off between storage and repair bandwidth for any set of \((n, k, d)\) parameters. It was also shown that the functional repair problem is intimately related to the celebrated network coding problem [3].

From a practical perspective however, exact repair of a failed node is highly desirable as it reduces the computational complexity of encoding/decoding the data as well as the maintenance of the storage system. Since the work of Dimakis et.al [2], which established the functional repair tradeoff, there have been several works on exact repair code constructions such as [4]–[7] (also see the references therein), and non-achievability of some points on the functional repair tradeoff [8]. However, the fundamental question regarding the gap between functional and exact repair tradeoff remained open until recently shown by Tian in [9] where it was shown that for the \((n, k, d) = (4, 3, 3)\), functional and exact repair tradeoffs are different through a novel computer-aided proof. Despite its significant originality, the solution involved an optimization problem over a large number of variables, therefore it is unclear if this approach is scalable to larger system parameters. Moreover, such a computer-aided approach does not necessarily lead to intuition and insights which are very valuable for system design (also see [10], [11] for more recent results). Despite the above recent progress on the exact repair problem, characterizing the fundamental exact repair trade-off for general \((n, k, d)\) parameters remains a challenging open problem.

In this paper, we focus on another key practical aspect of codes for DSS, i.e., linearity, in addition to the existing constraints of exact repair and data recovery. To the best of our knowledge, all practically deployed DSS employ linear codes, such as codes from the RAID family, variations of Reed-Solomon codes, and a variety of erasure codes (e.g., see [12]–[14]). Motivated by the popularity and practical relevance of linear codes, in this paper, we focus on linear codes with the exact repair property. Henceforth, we refer LDSS as distributed storage systems which employ linear codes.

In this paper, we make new progress on the exact repair problem by characterizing the optimal exact repair tradeoff of the \((n, k, d)\) LDSS when \(k = d = n - 1\). As mentioned earlier, characterizing the corresponding tradeoff for a general DSS (not necessarily restricted to be linear) remains open even when \(k = d = n - 1\) (barring the exception of Tian’s result [9] for \(k = 3\)).

Our key approach is to use the underlying algebraic structure of linear codes to provide bounds on the performance of LDSS. In particular, we exploit the duality between the multi-dimensional subspaces over finite fields and linear
Moreover, $S$ denote that $x$ to denote the largest integer number not exceeding $x$. For a random variable $X$, $H(X)$ denotes its entropy. Finally, for two vector spaces $U$ and $V$, we use $U \subseteq V$ to denote that $U$ is a subspace of $V$. We will introduce more notations after formally defining the problem.

II. Model and Problem Statement

**Notation:** Throughout this paper we use $[i : j] = \{i, i + 1, \ldots, j\}$ to denote the set of positive integers between (and including) $i$ and $j$. If $i = 1$, we drop it, and simply use $[j]$ to denote the set $\{1, 2, \ldots, j\}$. We also use $[x]$ to denote the largest integer number not exceeding $x$, and $\lfloor x \rfloor$ to denote the smallest integer number not less than $x$. For a random variable $X$, $H(X)$ denotes its entropy. Finally, for two vector spaces $U$ and $V$, we use $U \subseteq V$ to denote that $U$ is a subspace of $V$. We will introduce more notations after formally defining the problem.

A. Exact Repairable Regeneration Codes for Distributed Storage Systems

We first describe the exact repair problem for distributed storage systems. An exact repair DSS with parameters $(n, k, d) = (k+1, k, k)$ and $(\alpha, \beta)$ consists of $n = k+1$ storage devices, each with storage capacity $\alpha$. The entire data (Data) is encoded and stored over $n$ nodes in distributed manner. We use $W_i$ to denote the random variable representing the content stored in node $i$, and extend this definition to $W_A = \{W_i; i \in A\}$ for any $A \subseteq [k+1]$. Moreover, $S_{i \rightarrow j}$ denotes the random variable representing the repair data sent from node $i$ to repair node $j$. Note that since $n = d + 1$, there is a unique way of choosing $d$ helper nodes to repair any failed node within $[n]$. Therefore, the dependence of $S_{i \rightarrow j}$ on the remaining $(d-1)$ helper nodes, that is $[n] \setminus \{i, j\}$, is clear due to their uniqueness and is hence dropped from the notation for simplicity. We also set $S_{i \rightarrow i}$, to be a dummy random variable with zero entropy, for consistency. Moreover, $S_{A \rightarrow B} = \{S_{i \rightarrow j} : i \in A, j \in B\}$ denotes the total data sent by nodes in set $A$ for the repair of nodes in set $B$.

**Definition 1.** A code $C = \{W_i, i \in [k+1]; S_{i \rightarrow j}, i, j \in [k+1]\}$ is an exact-regeneration code for a $(k+1, k, k)$-DSS with parameters $(\alpha, \beta)$ if it satisfies the following properties:

- **Data size:** The node content and repair data sizes satisfy $H(W_i) \leq \alpha$ for $i \in [k+1]$, and $H(S_{i \rightarrow j}) \leq \beta$ for $i, j \in [k+1]$. Moreover, the repair data sent by a node is a function of the content stored on that node, that is $H(S_{i \rightarrow j}|W_i) = 0$ for $i, j \in [k+1]$.

- **Data Recovery:** The entire file can be recovered from the content of any $k$ nodes: $H(\text{Data}|W_A) = 0$ for any $A \subseteq [k+1]$ satisfying $|A| \geq k$. 

It is the goal of this paper to establish a characterization of the set of codes $C$ that are read optimal.
• **Failure Node (Exact) Repairability**: The content of any failed node can be exactly recovered (repaired) by receiving no more than $\beta$ units of repair data from the other $d = k$ nodes, that is, $H(W_i|S_{A \rightarrow i}) = 0$ for any $A = [k + 1] \setminus \{i\}$.

The next definition specifies achievable triples $(\alpha, \beta, F)$.

**Definition 2.** A triple $(\alpha, \beta, F)$ is called achievable for a $(k + 1, k, k)$-DSS if there exists an exact-regeneration code with parameters $(\alpha, \beta)$ that can store a file $\text{Data}$ of size $H(\text{Data}) = F$.

Our ultimate goal is to characterize optimum triples $(\alpha, \beta, F)$ for exact-regeneration codes.

**Definition 3.** An achievable operation point $(\alpha, \beta, F)$ is called pareto-optimum (or simply optimum) if any other achievable point $(\alpha', \beta', F')$ satisfies either with $\alpha'/F' > \alpha/F$ or $\beta'/F' > \beta/F$.

It turns out that for any given $F$, there is range for pairs of $(\alpha, \beta)$ that triples $(\alpha, \beta, F)$ is achievable [2]. More precisely, a trade-off between per-node-storage capacity ($\alpha$) and per-node repair bandwidth ($\beta$). Our focus is on the storage-repair bandwidth trade-off of the pareto-optimum regeneration codes.

We adopt the formal definition of symmetry for regeneration codes from [9] as follows.

**Definition 4.** A regeneration code $C = \{W_i, i \in [k + 1]; S_{i \rightarrow j}, i, j \in [k + 1]\}$ is called symmetric if all the information-theoretic quantities are preserved under any relabeling of the nodes. More precisely, it should satisfy

$$H(W_A, S_{B \rightarrow C}) = H(W_{\rho(A)}, S_{\rho(B) \rightarrow \rho(C)})$$

for every permutation $\rho : [k + 1] \rightarrow [k + 1]$.

Without loss of generality, we can restrict our attention to homogeneous and symmetric regeneration codes. The symmetrization lemma shows an arbitrary code can be used to form a symmetric one.

**Lemma 1 (Symmetrization Lemma).** Let $\mathcal{C} = \{W_i, i \in [k + 1]; S_{i \rightarrow j}, i, j \in [k + 1]\}$ be an arbitrary exact-regeneration code, with $\alpha_i = H(W_i)$ and $\beta_{ij} = H(S_{i \rightarrow j})$. Then one can build a code $\tilde{C}$ with parameters

$$(\tilde{\alpha}, \tilde{\beta}, \tilde{F}) = \left( k! \sum_{i=1}^{k+1} \alpha_i, (k - 1)! \sum_{i,j=1}^{k+1} \beta_{ij}, (k + 1)!F \right),$$

and hence, normalized triple $(\sum_i \alpha_i/(k + 1), \sum_{i,j} \beta_{ij}/k(k + 1), F)$ is achievable.

**Proof.** We start with $(k + 1)!$ copies of $\mathcal{C}$, each with separate and mutually independent $F$ data symbols, and form all possible permutations $\sigma$ of node labels. Then we augment them to each other, to obtain a new (and bigger) super-code $\tilde{C} = \{\tilde{W}_i; \tilde{S}_{i \rightarrow j}\}$, where

$$\tilde{W}_i = \{W_{\sigma(i)} : \forall \sigma : [k + 1] \rightarrow [k + 1]\} \quad \text{and} \quad \tilde{S}_{i \rightarrow j} = \{S_{\sigma(i) \rightarrow \sigma(j)} : \forall \sigma : [k + 1] \rightarrow [k + 1]\}.$$
Here $\sigma$ takes all possible permutations on $[k+1]$. Since each copy is exactly repairable, hence the super-code $\tilde{C}$ is also exactly repairable. Moreover, since the random variables involved in each copy is independent from the rest, for arbitrary subsets $(A, B, C)$ of nodes and any fixed permutation $\rho$ we have

\[
H(\tilde{W}_{\rho(A)}, \tilde{S}_{\rho(B) \rightarrow \rho(C)}) = \sum_{\sigma} H(\tilde{W}_{\sigma(\rho(A))}, \tilde{S}_{\sigma(\rho(B)) \rightarrow \sigma(\rho(C))})
= \sum_{\sigma'} H(\tilde{W}_{\sigma'(A)}, \tilde{S}_{\sigma'(B) \rightarrow \sigma'(C)}) = H(\tilde{W}_A, \tilde{S}_{B \rightarrow C}),
\]

where $\sigma'(i) = \sigma(\rho(i))$ also goes over all possible permutations. This shows the super-code is symmetric. In particular, we have

\[
H(\tilde{W}_i) = \sum_{\sigma} H(W_{\sigma(t)}) = \sum_{\sigma: \sigma(t) = i} \sum_{i=1}^{k+1} H(W_j) = k! \sum_{i=1}^{k+1} \alpha_i,
\]

\[
H(\tilde{S}_{s \rightarrow t}) = \sum_{\sigma} H(S_{\sigma(s) \rightarrow \sigma(t)}) = \sum_{\sigma: \sigma(s, t) = (i, j)} \sum_{i,j=1}^{k+1} H(S_{i \rightarrow j}) = (k-1)! \sum_{i,j=1}^{k+1} \beta_{ij},
\]

\[
H(\tilde{Data}) = \sum_{\sigma} H(Data_\sigma) = (k+1)! F.
\]

This completes the proof of the Lemma.

Symmetrization Lemma shows that focusing on symmetric regeneration codes leads to no loss in generality. Having any asymmetric code, one can from a symmetric super-code with scaled parameters, and since we are only interested in the trade-off between normalized parameters, i.e. $(\alpha/F, \beta/F)$, the symmetric code is as good as the original asymmetric one. Therefore, without loss of generality, we can restrict our attention to symmetric codes throughout this paper.

As a consequence, one can show that in an optimum regeneration code we have $H(W_i) = \alpha$ and $H(S_{i \rightarrow j}) = \beta$ for every $i, j \in \{1, \ldots, k+1\}$, otherwise one can build a new super-code with strictly smaller parameters.

We next tailor these definitions for the $(k+1, k, k)$ linear distributed storage systems (LDSS). Before that, we briefly review some standard definitions from linear algebra and finite field vector spaces.

**B. A Review on Vector Spaces over Finite Fields**

We first review some basic properties of vector spaces defined over finite fields. Most of the following definitions and properties would not be needed until we present the proofs of the theorems in the upcoming sections.

**Definition 5.** Let $\mathcal{F}$ be a vector space defined over some finite field $\mathbb{F}$. For two subspaces $\mathcal{V}, \mathcal{U} \subseteq \mathcal{F}$ over $\mathbb{F}$, we define the following relationships and operations as follows:

i) $\mathcal{V}$ is a subspace of $\mathcal{U}$ and denoted by $\mathcal{V} \subseteq \mathcal{U}$ if for every vector $v \in \mathcal{V}$ we have $v \in \mathcal{U}$.

ii) Intersection of $\mathcal{V}$ and $\mathcal{U}$ is defined as

\[
\mathcal{V} \cap \mathcal{U} = \{ v : v \in \mathcal{V}, v \in \mathcal{U} \}.
\]

The trivial intersection of any two vector spaces is the all-zero vector, denoted by $\mathbf{0}$. 
iii) Summation of $\mathcal{U}$ and $\mathcal{V}$ is defined as
\[
\mathcal{V} + \mathcal{U} = \{ v + u : v \in \mathcal{V}, u \in \mathcal{U} \}.
\]

If $\mathcal{V} \cap \mathcal{U} = \{0\}$, the summation of two subspaces is called as the direct sum and denoted by $\mathcal{V} \oplus \mathcal{U}$. Note that the intersection and summation are also vector spaces.

It is worth noting that unlike vector spaces defined over $\mathbb{R}$ (or $\mathbb{C}$) in which two orthogonal subspaces are always disjoint (except their trivial intersection at $\{0\}$), in a finite field vector space, a vector can be orthogonal to itself (e.g. $[1, 1] \cdot [1, 1]^T = 0$ in $\mathbb{F}_2$). Hence, we need an alternative approach to define the complement of a subspace. This is formally defined as follows.

**Definition 6.** Consider two vector spaces $\mathcal{U}$ and $\mathcal{V}$. We define the modulo space $\mathcal{U}_{\text{mod}(\mathcal{V})}$ ($\mathcal{U}$ after nulling $\mathcal{V}$) as
\[
\mathcal{U}_{\text{mod}(\mathcal{V})} \triangleq \text{span}\{ u \in \mathcal{U} : v = 0, \ \forall v \in \mathcal{V} \},
\]
that is the vector space spanned by vectors in $\mathcal{U}$ where every vector in $\mathcal{V}$ is collapsed to zero. In other words, $\mathcal{U}_{\text{mod}(\mathcal{V})}$ can be considered as the complement of $\mathcal{U}$ with respect to $\mathcal{V}$.

The following lemma characterizes some of the basic properties of this operator. The proof of this lemma follows from basic concepts of linear algebra [15], and hence omitted.

**Lemma 2.** The modulo operation satisfies the following properties:

(P.1) $\mathcal{U}_{\text{mod}(\mathcal{V})}$ preserves subspace relationship, i.e., if $\mathcal{U}_1 \subseteq \mathcal{U}_2$, then $\mathcal{U}_{1_{\text{mod}(\mathcal{V})}} \subseteq \mathcal{U}_{2_{\text{mod}(\mathcal{V})}}$.

(P.2) $\mathcal{U}_{\text{mod}(\mathcal{V})}$ is distributive over summation, that is,
\[
\mathcal{U}_{1 + \mathcal{U}_2_{\text{mod}(\mathcal{V})}} = \mathcal{U}_{1_{\text{mod}(\mathcal{V})}} + \mathcal{U}_{2_{\text{mod}(\mathcal{V})}},
\]

(P.3) $\mathcal{U}_{\text{mod}(\mathcal{V})}$ is an associative operation, i.e.,
\[
\left[ \mathcal{U}_{\text{mod}(\mathcal{V})} \right]_{\text{mod}(\mathcal{W})} = \left[ \mathcal{U}_{\text{mod}(\mathcal{V})} \right]_{\text{mod}(\mathcal{W})} = \left[ \mathcal{U}_{\text{mod}(\mathcal{V} + \mathcal{W})} \right]_{\text{mod}(\mathcal{V})},
\]

(P.4) $\dim(\mathcal{U}_{\text{mod}(\mathcal{V})}) = \dim(\mathcal{U}) - \dim(\mathcal{U} \cap \mathcal{V})$.

(P.5) If $\mathcal{V} \subseteq \mathcal{U}$, then $\mathcal{U}_{\text{mod}(\mathcal{V})} + \mathcal{V} = \mathcal{U}$, and in particular, $\mathcal{U}_{\text{mod}(\mathcal{V})} \subseteq \mathcal{U}$.

C. Linear Exact-Repair Regeneration Codes

In a linear regeneration code the original file of size $F$ is divided into $F$ sub-packets. These sub-packets form a basis for an $F$-dimensional vector space over a finite field $\mathbb{F}_q$, which we denote by $\mathcal{F}$. This vector space is indeed equivalent to the entire file. Therefore, each unit of stored data can be expressed as a linear combination of the $F$
sub-packets, which is indeed a vector in $\mathcal{F}$. Thus, the content of each disk is equivalent to a subspace of $\mathcal{F}$. We denote by $\mathcal{W}_i$, the subspace spanned by the vectors stored on the $i$-th node. Moreover, $S_{i \rightarrow j}$ denotes the subspace spanned by the vectors sent from node $i$ to node $j$ in order to repair $j$. Similar to the previous section, we also define

$$W_A = \sum_{i \in A} W_i \quad \text{and} \quad S_{A \rightarrow B} = \sum_{i \in A, j \in B} S_{i \rightarrow j}. \tag{2}$$

In this new framework, the concept of exact-repair regeneration code can be redefined as the following.

**Definition 7.** Let $\mathcal{F}$ be a vector space defined over a finite field $\mathbb{F}_q$ with $\dim(\mathcal{F}) = F$. An exact-repair regeneration code to store $\mathcal{F}$ in a $(k+1, k, k)$ DSS with parameters $(\alpha, \beta)$ is defined as $C = \{ W_i, i \in [k+1]; S_{i \rightarrow j}, i, j \in [k+1] \}$, where

i) each node $i$ stores a subspace $W_i \subseteq \mathcal{F}$ with $\dim(W_i) = \alpha$;

ii) the repair data from node $i$ to $j$ is a subspace $S_{i \rightarrow j} \subseteq W_i$ with $\dim(S_{i \rightarrow j}) = \beta$;

iii) the summation of the subspaces of every $k$ node span the entire space, that is that is, $\mathcal{F} = \sum_{i \in A} W_i$ for every $A \subset [k+1]$ with $|A| = k$;

iv) the subspace stored in each node can be spanned by the summation of the received repair subspaces from all other nodes. In other words, $W_j \subseteq \sum_{i \in [k+1] \backslash \{j\}} S_{i \rightarrow j}$.

A triple $(\alpha, \beta, F)$ is called linearly achievable for $(k+1, k, k)$-DSS if there exists such an exact-repair regeneration code.

Note the minor difference between this definition and Definition 1, that is, here we set $\dim(W_i) = \alpha$ (rather $\dim(W_i) \leq \alpha$) and $\dim(S_{i \rightarrow j}) = \beta$ (see the discussion at the end of Section II-A). This assumption is made because we are interested in optimum codes, and if dimension constraints do not match with equality, we can shrink at least one of the subspaces, and symmetrize the code using Lemma 1 to obtain a super-code with a lower (scaled) trade-off.

**Definition 8.** The linear capacity of a $(k+1, k, k)$-DSS is defined as the supremum of all values $F$ such that triple $(\alpha, \beta, F)$ is linearly achievable, and is denoted by $F_k(\alpha, \beta)$.

We next explain our approach to analyze the relationship between subspaces defined above, and derive bounds on the optimum trade-off of the LDSS.

### III. MAIN RESULTS

We will present an outer bound together with an inner bound for the optimum trade-off of exact repair regeneration codes. Both bounds are presented in terms of linear optimization problems, which can be analytically solved. It turns out that optimum solution of both bounds match, which yields in a complete characterization for the trade-off of interest.
Theorem 1. The exact-repair linear capacity of a \((k + 1, k, k)\)-DSS with per-node storage \(\alpha\) and total node repair-bandwidth \(k\beta\) is upper bounded by

\[
F_k(\alpha, \beta) \leq \max_{(\pi_1, \pi_2, \ldots, \pi_k) \in \mathcal{P}_k(\alpha, \beta)} \sum_{i=0}^{k-1} (\alpha - \pi_i),
\]

(OPT - OB)

where \(\pi_0 \equiv 0\), and the maximum is taken over all \((\pi_1, \pi_2, \ldots, \pi_k)\) in the set \(\mathcal{P}_k(\alpha, \beta)\), which is defined as

\[
\mathcal{P}_k(\alpha, \beta) = \left\{ (\pi_1, \pi_2, \ldots, \pi_k) : 0 \leq \pi_1 \leq \cdots \leq \pi_k = \alpha, \pi_i \geq \alpha - (k - i)\beta, i = 1, \ldots, k, \right\}
\]

The proof of this theorem is presented in Section IV.

Remark 1. Recall the optimum tradeoff of DSS with functional repair which is given by [2]

\[
F \leq \sum_{i=0}^{k-1} \min(\alpha, (k - i)\beta).
\]

Comparison the functional repair bound with the new bound for exact repair, using (OPT – OB), we find out that

\[
F \leq \max_{i=0}^{k-1} (\alpha - \pi_i) = \max_{i=0}^{k-1} \min(\alpha - \pi_i, (k - i)\beta),
\]

highlights the penalty we need to pay in terms of the total storage capacity of the system, in order to provide exact repair property.
The next theorem provides a lower bound for the linear capacity of a \((k+1, k, k)\)-DSS.

**Theorem 2.** The storage capacity of a distributed storage system with \((k+1, k, k)\) with per-node capacity \(\alpha\) and total repair bandwidth \(d\beta\), is at least

\[
F_k(\alpha, \beta) \geq \max_{(\theta_1, \ldots, \theta_k) \in Q_k(\alpha, \beta)} \sum_{i=1}^{k} i \binom{k+1}{i+1} \theta_i
\]

where maximization is performed over all non-negative \(q\)-vectors in \(Q_k(\alpha, \beta)\) defined as

\[
Q_k(\alpha, \beta) = \left\{ (\theta_1, \theta_2, \ldots, \theta_k) : \begin{array}{l}
    \theta_i \geq 0 \quad i = 1, \ldots, k \\
    \sum_{j=1}^{k} \binom{k}{j} \theta_j \leq \alpha \\
    \sum_{j=1}^{k} \binom{k-1}{j-1} \theta_j \leq \beta.
\end{array} \right\}
\]

The proof of Theorem 2 is presented in Section V.

We will propose a recursive code construction for a \((k+1, k, k)\)-DSS, and show that the obtained code is linear and can store up to \(\Psi_k(\theta_1, \theta_2, \ldots, \theta_k) \triangleq \sum_{i=1}^{k} i \binom{k+1}{i+1} \theta_i\) units of data for any feasible sequence \((\theta_1, \ldots, \theta_k)\). Next, we will show that the optimum solution of \((\text{OPT} - \text{OB})\) and \((\text{OPT} - \text{IB})\) match, and by finding such optimum solution, we can conclude the following theorem, which is a complete characterization for the linear capacity of an exact-repair distributed storage system.

**Theorem 3.** The linear capacity of a \((k+1, k, k)\)-DSS with per-node capacity \(\alpha\), and total repair bandwidth \(k\beta\) is given by

\[
F_k(\alpha, \beta) = \frac{k+1}{m+1} \left( k\alpha + \frac{k}{m+1} \beta \right),
\]

where \(m = \lfloor \frac{k\beta}{\alpha} \rfloor\). Moreover, the optimum trade-off of an exact-repair \((k+1, k, k)\)-DSS is given by the convex-hull of

\[
\left\{ (\alpha_i, \beta_i) = \left( \frac{i+1}{i(k+1)} F, \frac{i+1}{k(k+1)} F \right) : i = 1, 2, \ldots, k \right\}.
\]

The proof of Theorem 3 is presented in Section VI-C.

**IV. PROOF OF THEOREM 1: A LINEAR PROGRAMMING PERSPECTIVE**

This section is dedicated to proof of the outer bound. Before presenting the formal proof of Theorem 1, we need a few definitions and some primary results, which will be reviewed in the following.

The relationship between the subspaces stored on different storage nodes can be formally characterized using the following definition.
Definition 9. Consider a symmetric and linear regeneration code $C = \{W_i, i \in [k+1]; S_i \rightarrow j, i, j \in [k+1]\}$ for a $(k+1,k,k)$-DSS. We associate a characteristic vector $\pi = (\pi_1, \pi_2, \ldots, \pi_k)$ to $C$ where
\[
\pi_{|A|} = \dim (W_i \cap W_A) = \dim \left(W_i \cap \sum_{j \in A} W_j \right), \quad i \in [k+1], A \subseteq [k+1] \setminus \{i\},
\]
for $|A| = 1, \ldots, k$. We also set $\pi_0 = 0$ for consistency.

Moreover, a sequence $\pi = (\pi_1, \pi_2, \ldots, \pi_k)$ is called feasible if there exists a symmetric and linear regeneration code for $(k+1,k,k)$-DSS with associated vector $\pi$.

The characteristic vector together with $(\alpha, \beta)$ completely determine the storage capacity of a code, as formally stated in the following lemma.

Lemma 3. The storage capacity of a $(k+1,k,k)$-DSS operating at $(\alpha, \beta)$ and employs a linear code with characteristic vector $(\pi_1, \pi_2, \ldots, \pi_k)$ is given by
\[
F \leq \sum_{i=1}^{k} (\alpha - \pi_i).
\]

Proof of Lemma 3. First note that for an arbitrary set $A$, we have
\[
\dim (W_A) \leq \sum_{i=0}^{\lfloor A \rfloor - 1} (\alpha - \pi_i).
\]
This can be shown using an induction argument: for $|A| = 1$, this clear, since $\dim (W_i) \leq \alpha$. Now, assume the claim holds for some $A$. For set $B = A \cup \{j\}$ with $|B| = |A| + 1$, we have
\[
\dim (W_B) = \dim (W_A + W_j) = \dim (W_A) + \dim (W_j) - \dim (W_A \cap W_j)
\]
\[
\leq \dim (W_A) + \alpha - \pi_{|A|}
\]
\[
\leq \sum_{i=0}^{\lfloor A \rfloor - 1} (\alpha - \pi_i) + (\alpha - \pi_{|A|})
\]
\[
= \sum_{i=0}^{\lfloor A \rfloor} (\alpha - \pi_i) - \sum_{i=0}^{\lfloor B \rfloor - 1} (\alpha - \pi_i).
\]
Note that $(a)$ follows from the definition of $\pi_{|A|}$ in Definition 9, and in $(b)$ we used the assumption of induction for $A$.

Now, the proof of lemma is straight-forward, by recalling that the entire vector space is recoverable from any $k$ node subspaces. Hence, evaluating $(4)$ for some $A$ with $|A| = k$, we get
\[
F = \dim (\mathcal{F}) = \dim (W_A) \leq \sum_{i=0}^{k-1} (\alpha - \pi_i).
\]

\[Note that the definition of symmetry in Definition 4 implies \dim (W_i \cap W_A) does not depend on the choice of $i$, and $A$, except for $|A|$.\]
Recall that symmetric regeneration codes achieve the optimum capacity of DSS, and any symmetric linear code is associated with some feasible \( \pi \). Therefore, from Definition 9, the following lemma is immediate.

**Lemma 4.** The exact repair capacity of a \((k+1, k, k)\) LDSS with per node storage \( \alpha \) and per node repair bandwidth \( \beta \) is upper bounded by

\[
F_k(\alpha, \beta) \leq \max_{(\pi_1, \pi_2, \ldots, \pi_k)-\text{feasible}} \sum_{i=0}^{k-1} (\alpha - \pi_i).
\]

Note that lemma 4 is essentially an optimization problem over the feasible set for \((\pi_1, \pi_2, \ldots, \pi_k)\), which needs to be determined. The rest of this section is dedicated to formal characterization of this set.

### A. Set of Feasible \( \pi \)-Sequences

The sequence \( \pi = (\pi_1, \ldots, \pi_k) \) is defined in Definition 9 as a joint parameter of a family of subspaces. Hence, its entries are not isolated (independent) from each other, and not every sequence is feasible. In other words, a sequence \( \pi \) is feasible only if there exists an exact-regeneration code associated with it. Instead of characterizing the feasible sequences (which is perhaps not easier than solving the original problem), we introduce a set of necessary constraints that any feasible sequence should satisfy, and hence determine a super-set for the feasible set. We also characterize a set of constraints that the \( \pi \) sequence associated to any pareto-optimal LDSS should satisfy.

The following lemma states a necessary conditions on feasible sequences.

**Lemma 5.** Any feasible sequence \( \pi = (\pi_1, \ldots, \pi_k) \) satisfies

\begin{enumerate}
\item[(C1)] \( 0 \leq \pi_1 \leq \pi_2 \leq \cdots \leq \pi_k = \alpha \).
\item[(C2)] \( \pi_i \geq \alpha - (k - i)\beta \).
\end{enumerate}

**Proof.** The proof of the above lemma is based on the relationship between the vector spaces stored on nodes. Let \( \pi \) be a feasible sequence, associated with some exact-repair regeneration code \( C = \{\mathcal{W}_i; \mathcal{S}_{i \rightarrow j}\} \).

First note that all the \( \pi_i \)'s are non-negative integers since they are dimensions of some vector spaces. Moreover, for any pair of sets \( A \subseteq B \) we have

\[
\mathcal{W}_A = \sum_{j \in A} \mathcal{W}_j \subseteq \sum_{j \in A} \mathcal{W}_j + \sum_{j \in B \setminus A} \mathcal{W}_j = \mathcal{W}_B.
\]

Therefore, \( (\mathcal{W}_i \cap \mathcal{W}_A) \subseteq (\mathcal{W}_i \cap \mathcal{W}_B) \). This implies

\[
\pi_{|A|} = \dim (\mathcal{W}_i \cap \mathcal{W}_A) \leq \dim (\mathcal{W}_i \cap \mathcal{W}_B) = \pi_{|B|},
\]

which shows \( \pi \) is a non-decreasing sequence. Next, since \( C \) is exact-repairable, \( \mathcal{W}_{k+1} \) can be repaired by all other nodes, that is \( \mathcal{W}_{k+1} \subseteq \mathcal{S}_{[k] \rightarrow k+1} \subseteq \mathcal{W}_{[k]} \). Hence,

\[
\pi_k = \dim (\mathcal{W}_{k+1} \cap \mathcal{W}_{[k]}) = \dim (\mathcal{W}_{k+1}) = \alpha.
\]
This completes the proof of (C1).

In order to show (C2), we again start from $W_{k+1} \subseteq S_{[k] \rightarrow k+1}$. Let $A \subseteq [k]$ be a subset of nodes. We have

$$W_{k+1} \subseteq S_{[k] \rightarrow k+1} = S_{A \rightarrow k+1} + S_{[k] \setminus A \rightarrow k+1} \subseteq W_A + S_{[k] \setminus A \rightarrow k+1}.\]$$

Therefore,

$$\alpha = \dim(W_k) = \dim((W_{k+1} \cap (W_A + S_{[k] \setminus A \rightarrow k+1}))) \leq \dim(W_{k+1} \cap W_A) + \dim(S_{[k] \setminus A \rightarrow k+1}) \leq \dim(W_{k+1} \cap W_A) + \sum_{i \in [k] \setminus A} \dim(S_{i \rightarrow k+1}) \leq \pi_{|A|} + (k - |A|) \beta.$$

This implies $\pi_i \geq \alpha - (k - i) \beta$ for $i = 1, \ldots, k$. $\square$

Conditions (C1) and (C2) in Lemma 5 are universally necessary for every feasible characteristic vector $\pi$. In the following we present one more condition, which does not necessarily hold for every feasible $\pi$. However, we will show that any feasible $\pi$ that maximizes the outer bound in Theorem 1 should satisfy this last condition.

**Theorem 4.** Let $C$ be a symmetric and linear exact-regeneration code that operates at a pareto-optimal point $(\alpha, \beta, F)$. Then its characteristic vector $\pi$ satisfies

$$\sum_{j=1}^{i} (-1)^{i-j} \binom{i}{j} \pi_j \geq 0 \quad i = 1, \ldots, k. \quad (C3)$$

This Theorem will be proved in the rest of this section. We need some more definitions, before proving this Theorem.

**Definition 10.** Let $C = (W_1, \ldots, W_{k+1})$ be an exact-regeneration code for a $(k+1, k, k)$ LDSS, and $v \in F$ be an arbitrary vector stored in this system. A subset of nodes $A \subseteq [k+1]$ is called a representation for $v$ if $v \in W_A = \sum_{i \in A} W_i$. A representation $A$ is called minimal if no proper (strict) subset of $A$ includes $v$, that is, $v \notin W_B$ for every $B \subseteq A$.

The following propositions present two fundamental properties for optimum exact-regeneration codes.

**Proposition 1.** Let $C = (W_1, \ldots, W_{k+1})$ be a linear exact-regeneration code for a $(k+1, k, k)$-DSS. If $C$ is pareto-optimum, then any vector $v \in W_i$ in any node $i \in \{1, \ldots, k+1\}$ has exactly two minimal representations.

This proposition not only plays a major role in the proof of the outer bound, but also provides a guideline towards designing optimum codes for $(k+1, k, k)$ distributed storage systems. The proof of the proposition is presented in Appendix A.
Next proposition holds for exact-regeneration codes that satisfy the two-minimal-representation property. It will be proved in Appendix B.

**Proposition 2.** Let $C$ be a linear exact-regeneration code for a $(k+1,k,k)$-DSS, such that every vector $v \in \mathcal{W}_i$ has exactly two minimal representations for every $i \in [k+1]$. The characteristic vector $\pi$ of $C$ satisfies

$$\sum_{j=1}^{i} (-1)^{i-j} \binom{i}{j} \pi_j \geq 0 \quad i = 1, \ldots, k.$$ 

Now, Theorem 4 can be immediately implied from the two propositions.

**Proof of Theorem 4.** Note that code $C$ is pareto-optimum by assumption, and hence Propositions 1 implies that every vector $v \in \mathcal{W}_i$ in each node $i = 1, \ldots, k+1$ has exactly two minimal representations. Therefore the assumption of Proposition 2 holds, and we can conclude the inequality (C3). This completes the proof of the Theorem. \hfill $\square$

**B. Proof of Theorem 1**

We have developed all the required tools for the proof of Theorem 1. Note that Lemma 3 already provides an upper bound for the storage capacity of a DSS with exact-regeneration code.

The set of feasible $\pi$ vectors is shown in orange in Figure 2. Lemma 5 provides us with two necessary conditions for feasible characteristic vectors, i.e., the set of $\pi$ vectors satisfying $(C1) - (C2)$ (the yellow region in the figure) is a superset of feasible set. Furthermore, Theorem 4 provides a new condition for $\pi$ vectors that can maximize the objective function of Lemma 3. Hence, in order to search for the maximizing $\pi$ in the entire feasible set, we can restrict our attention to search among feasible vectors that satisfy $(C3)$. This set is illustrated in magenta in Figure 2. Instead we search for the optimum $\pi$ in a larger set, that is $P_k(\alpha, \beta)$, the intersection of $(C1) - (C2)$ and $(C3)$. This is the region within the dashed line in Figure 2.
It is clear that by expanding the search space for the optimum \( \pi \), we only upper bound the maximum of the objective function. More formally, we have

\[
\max_{\pi \text{ feasible}} k-1 \sum_{i=0}^{k-1} (\alpha - \pi_i) \leq \max_{\pi \in \mathcal{P}_k(\alpha, \beta)} k-1 \sum_{i=0}^{k-1} (\alpha - \pi_i).
\]

This together with Lemma 4 completes the proof of Theorem 1.

V. INNER BOUND: CANONICAL CODES

In the following we propose a new code construction for \((k+1, k, k)\) distributed storage system, and derive the performance of this code for a given \(k\) and pair of parameters \((\alpha, \beta)\). This performance analysis essentially implies the claim of Theorem 2.

The proposed code construction is based on a layered structure, which we will explain in detail in Section V-A. We then demonstrate an example of this layered code construction in the next section, which is helpful in understanding the general concept of canonical codes and to follow the mathematical analysis.

A. Layered Structure

A canonical code is formed by \(k\) layers. Each layer \(\ell\) encodes a certain set of information symbols \(V^{(\ell)}\), to generate corresponding parity symbols \(U^{(\ell)}\), and allocates them on the storage devices in a distributed manner.

The information symbols and the parity symbols in each layer are independent from those of all other layers. Layer \(\ell\) only contributes to the content of the first \((\ell+1)\) nodes. More precisely, layer \(\ell\) consists of \((\ell+1)\) sections, namely \(\{V_1^{(\ell)}, V_2^{(\ell)}, \ldots, V_\ell^{(\ell)}, U_{\ell+1}^{(\ell)}\}\), where \(V_i^{(\ell)}\)'s include information symbols of the \(\ell\)-th layer and will be stored in nodes indexed by \(i = 1, 2, \ldots, \ell\), and \(U_{\ell+1}^{(\ell)}\) includes the parity symbols of this layer and will be allocated on the \((\ell+1)\)-th node, in a systematic fashion.

The information symbols in layer \(\ell\) can be split into \(\ell\) disjoint sets

\[
V^{(\ell)} = V_1^{(\ell)} \cup V_2^{(\ell)} \cup \ldots \cup V_\ell^{(\ell)},
\]

where \(V_i^{(\ell)}\) denotes the set of information symbols stored in node \(i\). We use notation \(v_i^{(\ell)}[A; t] \in \mathbb{F}_q\) to enumerate the set of information symbols stored in node \(i\) by layer \(\ell\) of the code. The set of symbols stored in node \(i\) can be partitioned into symbols of type \(v_i^{(\ell)}[A; \cdot]\) for each subset \(A\) of nodes which include \(i\). Each of these types include \(\theta_{|A|}\) symbols, enumerated by a counter \(t\), where \(\theta_{|A|}\) are non-negative integers which will be determined later to satisfy the per node storage and bandwidth constraints. More precisely,

\[
V_i^{(\ell)} \triangleq \left\{ v_i^{(\ell)}[A; t] : A \subseteq [\ell], i \in A, t = 1, 2, \ldots, \theta_{|A|} \right\}. \tag{6}
\]

The parity symbols for layer \(\ell\) will be generated as linear combinations of the information symbols. In particular, symbols indicated by the same pair of \((A; t)\) from different nodes will be added to generate a parity symbol. More formally, the set of all parity symbols obtained by

\[
U_{\ell+1}^{(\ell)} \triangleq \left\{ u_{\ell+1}^{(\ell)}[A; t] = \sum_{j \in A} v_j^{(\ell)}[A; t] : A \subseteq [\ell], t = 1, 2, \ldots, \theta_{|A|} \right\}. \tag{7}
\]
will be stored on the node with label \((\ell + 1)\).

Next, we can find the number of symbols stored in each section corresponding to layer \(\ell\) of the code.

**Proposition 3.** The number of information symbols stored on each node \(i = 1, 2, \ldots, \ell\) using the \(\ell\)-th layer is given by

\[
\left| V_i^{(\ell)} \right| = \sum_{h=1}^{\ell} \binom{\ell - 1}{h - 1} \theta_h,
\]

and the number of parity symbols stored on node \((\ell + 1)\) is given by

\[
\left| U_{\ell+1}^{(\ell)} \right| = \sum_{h=1}^{\ell} \binom{\ell}{h} \theta_h.
\]

**Proof.** Recall the definition of \(V_i^{(\ell)}\) in (6) for \(i = 1, 2, \ldots, \ell\). Thus, we have

\[
\left| V_i^{(\ell)} \right| = \sum_{A \subseteq [\ell]} \sum_{t \in A} 1 = \sum_{A \subseteq [\ell]} \sum_{t \in A} \theta_{|A|} = \sum_{h=1}^{\ell} \sum_{A \subseteq [\ell]} \sum_{|A| = h - 1} \theta_h = \sum_{h=1}^{\ell} \binom{\ell - 1}{h - 1} \theta_h.
\]

The parity section of the \(\ell\)-th layer is defined in (7), which implies

\[
\left| U_{\ell+1}^{(\ell)} \right| = \sum_{A \subseteq [\ell]} \sum_{t \in A} 1 = \sum_{A \subseteq [\ell]} \sum_{t \in A} \theta_{|A|} = \sum_{h=1}^{\ell} \sum_{A \subseteq [\ell]} \sum_{|A| = h} \theta_h = \sum_{h=1}^{\ell} \binom{\ell}{h} \theta_h.
\]

\(\square\)

**Remark 2.** Note that the number of information symbols stored in the first \(\ell\) layers is not necessarily the same as the number of parity symbols stored in node \((\ell + 1)\). However, as we will describe later, the layers will be augmented in a nested manner, and form an overall homogeneous regeneration code, i.e., a code with identical number of symbols stored per node and identical number of repair symbols sent from one node to another.

Next, we will show two fundamental properties of this layer, namely, data recovery and node repair.

**Proposition 4** (Information Symbols Recovery Within A Layer). All the information symbols encoded by layer \(\ell\), namely, \(V^{(\ell)}\) can be recovered from any subset of \(B \subseteq [\ell + 1]\) with \(|B| = \ell\).

**Proof.** The claim is trivially true if \(B\) consists of the first \(\ell\) nodes indexed by \(i = 1, \ldots, \ell\).

Now, assume \(B = \{1, 2, \ldots, i-1, i+1, \ldots, \ell, \ell + 1\}\). It is easy to see that \(B\) contains all information symbols except \(V_i^{(\ell)}\). We will show that \(V_i^{(\ell)}\) can be recovered from the content of the elements of \(B\).

Consider an element \(v_i^{(\ell)}[A; t] \in V_i^{(\ell)}\), for some \(A \subseteq [\ell]\) with \(i \in A\) and \(t \in [\theta_{|A|}]\). We have

\[
v_i^{(\ell)}[A; t] = \sum_{j \in A} v_j^{(\ell)}[A; t] - \sum_{j \in A \setminus \{i\}} v_j^{(\ell)}[A; t]
= u_i^{(\ell)}[A; t] - \sum_{j \in A \setminus \{i\}} v_j^{(\ell)}[A; t].
\]

(8)
Therefore, $v_i^{(\ell)}[A; t]$ can be recovered from the symbols available in $B$.

Next we will show the repair property of the layer.

**Proposition 5 (Section Repairability Within A Layer).** Each missing section of a layer can be reconstructed using a subset of the symbols downloaded from the remaining $\ell$ sections.

We will determine the exact subset of symbols required for section recovery throughout the proof.

**Proof.** We can identify the following cases.

Case I: Reconstruction of $U_{\ell+1}^{(\ell)}$ from $B = \{V_1^{(1)}, V_2^{(1)}, \ldots, V_\ell^{(1)}\}$

Recall the general form of the parity symbols in (7). It is clear that $U_{\ell+1}^{(\ell)}$ can be reconstructed from all the information symbols, by setting

$$S_{i\rightarrow \ell+1}^{(\ell)} \triangleq V_i^{(\ell)}.$$  \hspace{1cm} (9)

We refer to this repair process as *parity-recovery*.

Case II: Reconstruction of $V_i^{(\ell)}$ from $B = \{V_1^{(\ell)}, \ldots, V_{i-1}^{(\ell)}, V_{i+1}^{(\ell)}, \ldots, V_\ell^{(\ell)}, U_{\ell+1}^{(\ell)}\}$

Now, let the $i$-th section is failed, and we wish to reconstruct it using the other sections. Consider an information symbol $v_i^{(\ell)}[A; t] \in V_i^{(\ell)}$ with $i \in A$, and recall from (8) that

$$v_i^{(\ell)}[A; t] = v_{A}^{(i)}[A; t] - \sum_{j \in A \setminus \{i\}} v_j^{(\ell)}[A; t].$$ \hspace{1cm} (10)

This equation shows that $v_i^{(\ell)}[A; t]$ can be reconstructed using a linear combination of $v_{A}^{(i)}[A; t] \in U_{\ell+1}^{(\ell)}$ and $v_j^{(\ell)}[A; t] \in V_i^{(\ell)}$. Hence, we can define

$$S_{\ell+1\rightarrow i}^{(\ell)} \triangleq \left\{ v_{\ell+1}^{(\ell)}[A; t] : A \subseteq [\ell], i \in A, t = 1, 2, \ldots, \theta_{[A]} \right\}$$ \hspace{1cm} (11)

and

$$S_{j \rightarrow i}^{(\ell)} \triangleq \left\{ v_j^{(\ell)}[A; t] : A \subseteq [\ell], i, j \in A, t = 1, 2, \ldots, \theta_{[A]} \right\} \hspace{1cm} (j \in [\ell], j \neq i) \hspace{1cm} (12)$$

to include all the required repair data from every section in $B$.

Note that $u_{A}^{(i)}[A; t]$ is a linear combination of $v_i^{(\ell)}[A; t]$ together with some other symbols which act as interference. In order to repair $v_i^{(\ell)}[A; t]$, we download $u_{A}^{(i)}[A; t]$, and then cancel the interference by the help of the other sections. So, we refer to this repair process as *interference cancellation*.

The number of symbols used for the repair process in layer $\ell$ is given by the following proposition.

**Proposition 6.** The size of the repair data required to repair a missing section within a layer $\ell$ is given by

$$\left| S_{1\rightarrow j}^{(\ell)} \right| = \begin{cases} \sum_{h=1}^{\ell} \binom{\ell}{h-1} \theta_h & \text{if } i \in [\ell] \text{ and } j = \ell + 1 \\ \sum_{h=1}^{\ell} \binom{\ell}{h-1} \theta_h & \text{if } i = \ell + 1 \text{ and } j \in [\ell] \\ \sum_{h=2}^{\ell} \binom{\ell-2}{h-2} \theta_h & \text{if } i, j \in [\ell] \end{cases}$$ \hspace{1cm} (13)
Proof. First note that from (9) and Proposition 3, we have

$$|S_{i \rightarrow \ell + 1}^{(\ell)}| = |V_i^{(\ell)}| = \sum_{h=1}^{\ell-1} \binom{\ell-1}{h-1} \theta_h.$$  

Next, the repair data sent from section $\ell + 1$ to a section $j$ is given in (11). Hence,

$$|S_{\ell + 1 \rightarrow j}^{(\ell)}| = \sum_{A \subseteq [\ell]} \sum_{t=1}^{\ell} 1 = \sum_{A \subseteq [\ell]} \sum_{j \in A} \theta_{|A|} = \sum_{h=1}^{\ell} \binom{\ell-1}{h-1} \theta_h.$$  

Finally, equation (12) provides the repair data needed to be sent from section $i$ in order to repair section $j$ in the $\ell$-th layer. Hence, for $i, j \in [\ell]$ we have

$$|S_{i \rightarrow j}^{(\ell)}| = \sum_{A \subseteq [\ell]} \sum_{t=1}^{\ell} 1 = \sum_{A \subseteq [\ell]} \sum_{i, j \in A} \theta_{|A|} = \sum_{h=2}^{\ell} \binom{\ell-2}{h-2} \theta_h.$$  

This completes the proof of the proposition. \hfill \qed

B. Canonical Code for $k = 4$

Before presenting the general notion of canonical codes, we illustrate the main idea through an example. Consider a distributed storage system with $n = k + 1 = 5$ nodes. We pick a vector $\underline{\theta} = (\theta_1, \theta_2, \theta_3, \theta_4) = (0, 2, 1, 0)$, and construct the canonical code for this vector.

The code is obtained by nested concatenation of four layers, namely, $\ell = 1, 2, 3, 4$, as illustrated in Figure 3. Each layer $\ell$ is demonstrated by a distinct color, and highlighted by an $L$-shape background.

Note that the first layer corresponding to $\ell = 1$ is empty since it only depends on $\theta_1 = 0$.

The second layer (shown in green), touches three nodes $i = 1, 2, 3$. Node $i = 1$ includes information symbols of form $v_1^{(2)}[A; t]$ for $A \subseteq \{1, 2\}$ with $1 \in A$. There are only two of such subsets, namely $\{1\}$ and $\{1, 2\}$, with $|\{1\}| = 1$ and $|\{1, 2\}| = 2$. Recall that $\theta_1 = 0$ and $\theta_1 = 2$. Hence, there is no symbol of the form $v_1^{(2)}[\{1\}; t]$, while we have symbols $v_1^{(2)}[\{1, 2\}; 1] = v_1^{(2)}[\{1, 2\}; 2]$ stored in section 1 on node 1. The information symbols of node $i = 2$ can be found in a similar way. Node $i = 3$ only includes the (parity) $u$-symbols of this layer.

There is a similar interpretation for the contents of the nodes in layers $\ell = 3$ and $\ell = 4$.

Note that the overall code consists of $F = 52$ information symbols, which are distributedly stored on disks with storage capacity $\alpha = 6$ each. It can be shown that each missing node can be repaired by receiving $\beta = 9$ symbols from each of the remaining nodes.

Remark 3. It is worth mentioning that this code satisfies the repair-by-transfer property [8] (also referred to as uncoded-repair in [16]), i.e., helper nodes simply collect the repair data needed for each repair process from their stored symbols without any computation. In order to repair a node $j$, each helper node $i$ should send (information and parity) symbols indexed by sets $A$ such that $i, j \in A$. On the other hand the replacement process at the failed node can be simply performed by summation of certain received symbols, and no other sophisticated arithmetic
Fig. 3. Canonical Code for $\theta_k = (0, 2, 1, 0)$.

operation is needed. This property significantly simplifies system implementation, and therefore is of practical interest. This also holds for the general canonical codes presented in this work.

C. Canonical Regeneration Codes

We introduced the notion of layer and analyzed its properties for data recovery and section repair in Section V-A. Here we will present the construction of a homogeneous regeneration code from multiple layers, by augmenting them in a nested manner.

A canonical regeneration code for a $(k+1, k, k)$-DSS can be constructed for any feasible characteristic vector $\theta_k = (\theta_1, \theta_2, \ldots, \theta_k)$. We will later explicitly define the set of feasible characteristic vectors, but we can simply consider any integer valued vectors $\bar{\theta}$ for now.
Recall the layer construction in Section V-A for \( \ell = 1, 2, \ldots, k \) using the first \( \ell \) entries of \( \theta \). Hence, we have a family of information and parity sections

\[
\left\{ V_1^{(i)}, V_2^{(i)}, \ldots, V_{\ell}^{(i)}, U_{\ell+1}^{(i)} \right\}_{\ell=1}^{k}
\]

where \( V_i^{(\ell)} \) is the \( i \)-th information section of the layer \( \ell \), and \( U_{\ell+1}^{(i)} \) is the parity section of the layer.

Then a canonical code for the system can be formally defined as follow.

**Definition 11.** A canonical code associated to a characteristic function \( \theta_k = (\theta_1, \theta_2, \ldots, \theta_k) \) is defined as \( C = \{ W_i, i \in [k+1]; S_{i \rightarrow j}, i, j \in [k+1] \} \), where

\[
W_i \triangleq \left( V_i^{(i)} \cup V_i^{(i+1)} \cup \cdots \cup V_i^{(k)} \right) \cup U_i^{(i-1)} \quad i = 1, 2, \ldots, k + 1.
\] (14)

Here \( V_i^{(\ell)} \) and \( U_i^{(i-1)} \) are defined in (6) and (7), and we define layer \( \ell = 0 \) to be a null layer for consistency.

Moreover, the repair data sent by node \( i \) to repair node \( j \) is defined as

\[
S_{i \rightarrow j} = \bigcup_{\ell = \max(i, j) - 1}^{k} S_{i \rightarrow j}^{(\ell)} = \left( S_{i \rightarrow j}^{(\max(i, j) - 1)} \cup S_{i \rightarrow j}^{(\max(i, j))} \cup S_{i \rightarrow j}^{(\max(i, j) + 1)} \cup \cdots \cup S_{i \rightarrow j}^{(k)} \right) \] (15)

for \( i, j \in [k + 1] \).

Note that this code can store

\[
\text{Data} = \bigcup_{\ell = 1}^{k} V_i^{(\ell)} = \bigcup_{\ell = 1}^{k} \bigcup_{i=1}^{\ell} \left( V_i^{(\ell)} \cup U_i^{(\ell+1)} \cup \cdots \cup V_i^{(k)} \right) \cup U_i^{(i-1)}
\] (16)

in a distributed manner. Next we will show that the constructed code maintains the two fundamental properties of regeneration codes, namely, data recovery and exact repair for any failed node, for a \( (n, k, d) = (k + 1, k, k) \)-distributed storage system.

The following proposition show the data recovery property of the constructed code.

**Proposition 7** (Data Recovery). For any subset of nodes \( B \subset [k+1] \) of size \( |B| = k \), all the (information) symbols in \( \text{Data} \) (defined in (16)) can be recovered from the content of the nodes in \( B \), that is, \( W_B = \bigcup_{i \in B} W_i \).

**Proof.** Assume \( B = [k+1] \setminus \{i\} \), i.e., \( B \) consists of all the nodes except node \( i \). Hence, \( W_B \) includes everything, except

\[
W_i = \left( V_i^{(i)} \cup V_i^{(i+1)} \cup \cdots \cup V_i^{(k)} \right) \cup U_i^{(i-1)}.
\]

Note that \( U_i^{(i-1)} \) is just the set parity symbols. We will show that all the missing information symbols in \( W_i \) can be recovered from \( W_B \).

Consider \( V_i^{(\ell)} \) for some \( \ell \in i, i + 1, \ldots, k \). We have already shown in Proposition 4 that within each layer, each missing section can be recovered from the remaining sections. More precisely, for each layer \( \ell \), the information symbols \( V_i^{(\ell)} \) can be recovered from

\[
\left( \bigcup_{j \in [\ell] \setminus \{i\}} V_j^{(\ell)} \right) \cup U_{\ell+1}^{(\ell)} \subseteq \left( \bigcup_{j \in [\ell] \setminus \{i\}} W_j \right) \cup W_{\ell+1} \subseteq W_B.
\]
Using this argument for \( \ell = i, i + 1, \ldots, k \), we can recover all the information symbols in \( W_i \), which shows the entire file can be recovered from \( W_B \). 

The next proposition below is a formal statement of the exact repair property of the proposed code.

**Proposition 8 (Repairability).** The content of any node \( i \), can be exactly reconstructed from the union of the repair data \( S_{i \rightarrow j} \) (defined in (15)) sent by nodes \( j \in B \), where \( B = [k + 1] \setminus \{i\} \).

**Proof.** This proof is based on the fact that symbols in each layer are independent from all others, and use the repair process within the layers. Recall the content of \( W \), defined in (14). We will prove repairability of sections \( V_i^{(\ell)} \) (for \( \ell = i, i + 1, \ldots, k \)) and \( U_i^{(i-1)} \), separately.

First, consider a section of information symbols \( V_i^{(\ell)} \) for some \( \ell = i, i + 1, \ldots, k \), and recall the argument in the proof of Proposition 5, which shows \( V_i^{(\ell)} \) can be recovered (exactly repaired) from the union of all \( S_{j \rightarrow i}^{(\ell)} \) (for \( j \in [\ell] \setminus \{i\} \)) and \( S_{\ell+1 \rightarrow i}^{(\ell)} \). More formally,

\[
V_i^{(\ell)} \leftarrow \left( \bigcup_{j \in [\ell] \setminus \{i\}} S_{j \rightarrow i}^{(\ell)} \right) \cup S_{\ell+1 \rightarrow i}^{(\ell)} = \bigcup_{j \in [\ell+1] \setminus \{i\}} S_{j \rightarrow i}^{(\ell)}.
\]

Next note that for every \( j \in [\ell + 1] \setminus \{i\} \) and \( \ell = i, i + 1, \ldots, k \), we have \( \max(i, j) \leq \ell + 1 \), and hence,

\[
S_{\ell+1 \rightarrow i}^{(\ell)} \subseteq \left( \bigcup_{\ell = \max(i, j) - 1} S_{j \rightarrow i}^{(\ell)} \right) = S_{j \rightarrow i}.
\]

Therefore,

\[
V_i^{(\ell)} \leftarrow \bigcup_{j \in [\ell+1] \setminus \{i\}} S_{j \rightarrow i} \quad \ell = i + 1, \ldots, k.
\]

Next, consider the parity section of \( W_i \), that is \( U_i^{(i-1)} \). By definition, every element in this set is just the parity of the information symbols of layer \( (i - 1) \), which is a linear combination of symbols in \( V_j^{(i-1)} \), for \( j = 1, 2, \ldots, i - 1 \). Thus,

\[
U_i^{(i-1)} \leftarrow \left( \bigcup_{j \in [i-1]} V_j^{(i-1)} \right) = \left( \bigcup_{j \in [i-1]} S_j^{(i-1)} \right) \subseteq \left( \bigcup_{j \in [i-1]} \bigcup_{\ell = \max(i, j) - 1} S_{j \rightarrow i}^{(\ell)} \right) = \left( \bigcup_{j \in [i-1]} S_{j \rightarrow i} \right)
\]

This completes the proof of node repairability. 

Propositions 7 and 8 show that the constructed code is indeed an exact-repair regeneration code. In the next section, we analyze the required per-node-capacity and repair-bandwidth of the proposed canonical code.

**D. Performance Analysis**

In this section we derive the total size of the file, the size of the content of each node \( W_i \), as well as the size of the repair data sent from each node \( i \) to repair node \( j \). In particular, we show that canonical code introduced in this section is homogeneous.

**Proposition 9.** The proposed canonical code with characteristic vector \( \theta_k = (\theta_1, \theta_2, \ldots, \theta_k) \) can store a total of

\[
\Psi_k(\theta_1, \theta_2, \ldots, \theta_k) \triangleq \sum_{h=1}^{k} \binom{k+1}{h+1} \theta_h
\]

(17)
symbols in \( n = k + 1 \) disks, each with storage capacity

\[
|W| = k \sum_{h=1}^{k} \binom{k}{h} \theta_h.
\]

Moreover, the content of each failed node can be (exactly) repaired by receiving

\[
|S_{i \rightarrow j}| = k \sum_{h=1}^{k} \binom{k-1}{h-1} \theta_h
\]

units of data from the remaining \( d = k \) disks.

**Proof.** Recall the definition of the entire file, consisting all information symbols in all layers, from (16). Using Proposition 3, we can write

\[
 F = \left| \bigcup_{\ell=1}^{k} \bigcup_{i=1}^{\ell} V_i^{(\ell)} \right| = \sum_{\ell=1}^{k} \sum_{i=1}^{\ell} |V_i^{(\ell)}| = \sum_{\ell=1}^{k} \sum_{i=1}^{\ell} \sum_{h=1}^{\ell} \binom{\ell-1}{h-1} \theta_h = \sum_{\ell=1}^{k} \sum_{h=1}^{\ell} \binom{\ell-1}{h-1} \theta_h
\]

\[
 = k \sum_{h=1}^{k} \binom{\ell}{h} \theta_h = k \sum_{h=1}^{k} \theta_h \sum_{\ell=1}^{k} \binom{\ell}{h} = \sum_{h=1}^{k} h \binom{\ell}{h} \theta_h,
\]

where in (\( \ast \)) we have used the combinatorial identity

\[
\sum_{j=1}^{n} \binom{j}{i} = \binom{j+1}{i+1} - \binom{j+1}{i}.
\]

The same identity will be used in the rest of this proof wherever indicated by (\( \ast \)). Note that the obtained quantity for \( F \) is a function of \( \theta_1, \theta_2, \ldots, \theta_k \), and we denote it by \( \Psi_k(\theta_1, \theta_2, \ldots, \theta_k) \).

Similarly, we can find the size of node \( i \) from (14) and Proposition 3:

\[
|W_i| = \sum_{\ell=1}^{k} |V_i^{(\ell)}| + |U_i^{(i-1)}| = \sum_{\ell=1}^{k} \sum_{h=1}^{\ell} \binom{\ell-1}{h-1} \theta_h + \sum_{h=1}^{i-1} \binom{i-1}{h} \theta_h
\]

\[
\overset{(a)}{=} \sum_{h=1}^{k} \sum_{\ell=1}^{k} \binom{\ell-1}{h-1} \theta_h + \sum_{h=1}^{i-1} \binom{i-1}{h} \theta_h
\]

\[
= \sum_{h=1}^{k} \theta_h \left[ \sum_{\ell=1}^{k} \binom{\ell-1}{h-1} \right] + \sum_{h=1}^{i-1} \binom{i-1}{h} \theta_h
\]

\[
\overset{(\ast)}{=} \sum_{h=1}^{k} \left[ \binom{k}{h} - \binom{i-1}{h} \right] \theta_h + \sum_{h=1}^{i-1} \binom{i-1}{h} \theta_h
\]

\[
\overset{(a)}{=} \sum_{h=1}^{k} \binom{k}{h} \theta_h
\]

where in (\( \ast \)) we used the fact that \( \binom{j}{i} = 0 \) when \( j < i \).

Finally, the size of the repair data can be found from (15) and Proposition 6. Let \( m \triangleq \max(i, j) - 1 \). Then, we
have

\[
|S_{i \rightarrow j}| = \sum_{\ell = m}^{k} |S_{i \rightarrow j}^{(\ell)}| = |S_{i \rightarrow j}^{(m)}| + \sum_{\ell = m+1}^{k} |S_{i \rightarrow j}^{(\ell)}|
\]

\[
= \sum_{h=1}^{m} \binom{m-1}{h-1} \theta_h + \sum_{\ell=m+1}^{k} \sum_{h=1}^{\ell} \binom{\ell-2}{h-2} \theta_h
\]

\[
= \sum_{h=1}^{m} \binom{m-1}{h-1} \theta_h + \sum_{h=1}^{k} \sum_{\ell=m+1}^{k} \binom{\ell-2}{h-2} \theta_h
\]

\[
= \sum_{h=1}^{m} \binom{m-1}{h-1} \theta_h + \sum_{h=1}^{k} \left[ \binom{k-1}{h-1} - \binom{m-1}{h-1} \right] \theta_h
\]

\[
= \sum_{h=1}^{k} \binom{k-1}{h-1} \theta_h
\]

Remark 4. Note that size of node content as well as repair data does not depend on the node labels. Hence, even though the layers form a non-homogeneous structure, the proposed canonical code is homogeneous.

Note that a $\theta$-canonical code can store up to $\Psi_k(\theta_1, \theta_2, \ldots, \theta_k)$ units of data. However, for a DSS with given $(\alpha, \beta)$, the only feasible $\theta$ vectors are those resulting in $|W_i| \leq \alpha$ and $|S_{i \rightarrow j}| \leq \beta$. Therefore, a canonical code for a $(k+1, k, k)$ DSS with per node storage $\alpha$ and repair bandwidth $\beta$, can store up to

\[
\max \Psi_k(\theta_1, \theta_2, \ldots, \theta_k) = \sum_{j=1}^{k} j \binom{k+1}{j+1} \theta_j
\]

units of data, where the maximization is taken over all non-negative integer-valued characteristic vectors $\theta = (\theta_1, \theta_2, \ldots, \theta_k)$ satisfying

\[
\sum_{j=1}^{k} \binom{k}{j} \theta_j \leq \alpha
\]

\[
\sum_{j=1}^{k} \binom{k-1}{j-1} \theta_j \leq \beta.
\]

This completes the proof of Theorem 2.

Remark 5. It is worth noting that the integral constraint in the optimization problem can be relaxed: Since all the coefficients in the constraints as well as the objective function in (OPT − IB) are integral, the coordinates of the optimum solution for the relaxed problem are always linear combinations of $\alpha$ and $\beta$ with rational coefficients. We can further scale up all the $\theta_i$'s by the least common multiple (l.c.m.) of all the denominators, which also results in scaling of $\alpha$, $\beta$, as well as the file size. Hence, the resulting solution is integer as long as $\alpha$ and $\beta$ are integers too.
We can arbitrarily approach this latter condition as long as we concern about normalized parameters \( \bar{\alpha} = \alpha / F \) and \( \bar{\beta} = \beta / F \).

**Remark 6.** Even though this canonical code is constructed through solving an optimization problem, it turns out that its structure is similar to that presented in [5]. However, the construction in [5] is specialized to the extreme points on the optimum trade-off, where the interior points can be achieved using space-sharing. The current construction is more general in the sense that it can be directly applied to any achievable pair \((\alpha, \beta)\).

**VI. EVALUATION OF OPTIMIZATION PROBLEMS**

In this section we analyze the two optimization problems in \((\text{OPT} - \text{OB})\) and \((\text{OPT} - \text{IB})\), and derive their optimum solutions. By showing that the optimum points of the two problems match, we provide the complete linear capacity characterization, which is indeed the proof of Theorem 3. We start with showing equivalence between the optimization problems in \((\text{OPT} - \text{OB})\) and \((\text{OPT} - \text{IB})\).

**A. Equivalence Between the Outer and Inner Bounds**

We rewrite the optimization problem in \((\text{OPT} - \text{OB})\) in a new format, to be able to closely compare it with the one in \((\text{OPT} - \text{IB})\). Recall parameters \( \theta_i \) for \( i = 1, \ldots, k \) used in the proof of Proposition 2 (see Definition 12 in Appendix B). We wish to rephrase the optimization problem in \((\text{OPT} - \text{OB})\) in terms of \( \theta_i \)'s, by setting \( \bar{\pi} = U\theta \) where \( U_{i,j} = \binom{i}{j} \).

First from (38), constraint (C3) on \( \bar{\pi} \) is equivalent to non-negativity of \( \theta_i \)'s, i.e., \( \theta \geq 0 \).

In order to rewrite (C1) for new variables, we have

\[
\begin{align*}
\pi_{i+1} - \pi_i &= \sum_{j=1}^{i+1} \binom{i+1}{j} \theta_j - \sum_{j=1}^{i} \binom{i}{j} \theta_j \\
&= \sum_{j=1}^{i+1} \left[ \binom{i+1}{j} - \binom{i}{j} \right] \theta_j = \sum_{j=1}^{i+1} \binom{i}{j-1} \theta_j \geq 0.
\end{align*}
\]

Therefore \( \pi_{i+1} - \pi_i \geq 0 \) is naturally guaranteed by \( \theta \geq 0 \). We also have

\[
\pi_k = \binom{k}{0} \binom{k}{1} \binom{k}{2} \cdots \binom{k}{k-1} \theta.
\]

Hence, \( \pi_k = \alpha \) will be translated to

\[
\sum_{j=1}^{k} \binom{k}{i} \theta_i = \alpha.
\]

The inequality in (C2) corresponding to \( i = k - 1 \) will be translated to

\[
\beta \geq \alpha - \pi_{k-1} = \pi_k - \pi_{k-1} = \sum_{j=1}^{k-1} \binom{k}{j-1} \theta_j.
\]
It can be shown that \((C2)\) for \(i < k - 1\) is naturally guaranteed by \((22)-(23)\). To this end, we can write

\[
\alpha - \pi_i = (\alpha - \pi_{k-1}) + \sum_{\ell=i}^{k-2} (\pi_{\ell+1} - \pi_\ell) = \sum_{j=1}^{k} \binom{k-1}{j-1} \theta_j + \sum_{\ell=i}^{k-2} \sum_{j=1}^{\ell} \binom{\ell}{j-1} \theta_j \\
\leq \sum_{j=1}^{k} \binom{k-1}{j-1} \theta_j + \sum_{\ell=i}^{k-2} \sum_{j=1}^{k} \binom{k-1}{j-1} \theta_j \\
= (k - i) \sum_{j=1}^{k} \binom{k-1}{j-1} \theta_j \leq (k - i) \beta
\]

Finally the objective function in \((\text{OPT} - \text{OB})\) can be rewritten as

\[
\sum_{i=0}^{k-1} (\alpha - \pi_i) = (k + 1) \alpha - 1^T \pi \\
= (k + 1) \alpha - 1^T U \tilde{\theta} \\
= \sum_{j=1}^{k} (k + 1) \binom{k}{j} \theta_j - \sum_{j=1}^{k} \left[ \theta_j \sum_{i=j}^{k} \binom{i}{j} \right] \\
\overset{(*)}{=} \sum_{j=1}^{k} \binom{j + 1}{j+1} \theta_j - \sum_{j=1}^{k} \binom{k+1}{j+1} \theta_j \\
= \sum_{j=1}^{k} \binom{j + 1}{j+1} \theta_j
\]

where in \((*)\) we used the identity in \((20)\). Now, writing the maximization problem for \((24)\) subject to \(\tilde{\theta} \geq 0\) and \((22)-(23)\), we reach to

\[
\max_{\theta_1, \ldots, \theta_k} \sum_{j=1}^{k} \binom{j + 1}{j+1} \theta_j \\
\text{subject to } \theta_j \geq 0 \quad i = 1, \ldots, k, \\
\sum_{j=1}^{k} \binom{k}{j} \theta_j = \alpha, \\
\sum_{j=1}^{k-1} \binom{k-1}{j-1} \theta_j \leq \beta.
\]

\((\text{OPT} - \text{OB}^+)\)

It is worth reminding that \((\text{OPT} - \text{OB}^+)\) is nothing but rephrasing \((\text{OPT} - \text{OB})\) in terms of new variables, and hence they share the same maximum value. Also note that \((\text{OPT} - \text{OB}^+)\) and \((\text{OPT} - \text{IB})\) are identical, except in one constraint (involving \(\alpha\)) where the equality in \((\text{OPT} - \text{OB}^+)\) is relaxed to an inequality in \((\text{OPT} - \text{IB})\). We will see next that the constraint in \((\text{OPT} - \text{IB})\) is indeed tight at the solution of the optimization problem.
**B. Evaluation of the Inner Bound**

This section is dedicated to analytically solve the optimization problem in \( \text{OPT} - \text{IB} \). We start with writing the dual of the problem as [17]

\[
\begin{align*}
\min & \quad \alpha r_1 + \beta r_2 \\
\text{subject to} & \quad \binom{k}{i} r_1 + \binom{k-1}{i-1} r_2 \geq i \binom{k+1}{i+1} \quad i = 1, 2, \ldots, k \\
& \quad r_1, r_2 \geq 0.
\end{align*}
\]

(Dual-OP)

Let \( F^* \) be the maximum value of the objective function in the primal problem, and \( G^* \) be the minimum solution of the dual problem corresponding to optimum values \( r^*_1 \) and \( r^*_2 \), i.e., \( G^* = \alpha r^*_1 + \beta r^*_2 \). The strong duality condition [17] for these linear optimization problems implies that \( F^* = G^* \).

To simplify the analysis we distinguish two cases based on whether \( k\beta/\alpha \) is an integer or not.

**Case I: If \( k\beta/\alpha \) is an integer:** The analysis in this case is fairly simple. The constraint at \( i = m \equiv k\beta/\alpha \) in (Dual-OP) can be simplified to

\[
kr^*_1 + mr^*_2 \geq \frac{m(k+1)}{m+1}
\]

which implies

\[
G^* = \alpha r^*_1 + \beta r^*_2 = \alpha \frac{m(k+1)}{m+1} \geq \alpha \frac{mk(k+1)}{m+1} = \frac{m(k+1)\alpha}{m+1}.
\]

On the other hand, it is easy to show that

\[
\tilde{r}_1 = \frac{(m-1)(k+1)}{m+1}, \quad \tilde{r}_2 = \frac{k(k+1)}{m(m+1)}
\]

satisfy all the constraints in (Dual-OP), and hence, \((\tilde{r}_1, \tilde{r}_2)\) is feasible. To this end, we can write

\[
\frac{\binom{k}{i}}{i\binom{k+1}{i+1}} \tilde{r}_1 + \frac{\binom{k-1}{i-1}}{i\binom{k+1}{i+1}} \tilde{r}_2 = \frac{i+1}{i(k+1)} \frac{(m-1)(k+1)}{m+1} + \frac{i+1}{i(k+1)} \frac{k(k+1)}{m(m+1)} = 1 + \frac{(i-m)^2 + (i-m)}{im(m+1)} \geq 1
\]

where in (27) we have used the fact that \( a^2 + a \geq 0 \) for every integer \( a \).

Therefore,

\[
G^* \leq \alpha \tilde{r}_1 + \beta \tilde{r}_2 = \alpha \frac{(m-1)(k+1)}{m+1} + \beta \frac{k(k+1)}{m(m+1)} = \frac{k+1}{(m+1)(m+2)} [m(m-1)\alpha + k\beta] = \frac{k+1}{(m+1)(m+2)} \frac{m(m-1)\alpha + m\alpha}{m+1}
\]

\[
= \frac{m(k+1)\alpha}{m+2}.
\]

(28)

Now, (26) together with (28) imply

\[
F^* = G^* = \frac{m(k+1)\alpha}{m+1} = \frac{k+1}{m+2} \left( m\alpha + \frac{k}{m+1} \beta \right).
\]
One can easily check that $F^*$ can be achieved by setting

$$
\theta_i^* = \begin{cases} 
\frac{\alpha}{\beta} & i = \frac{k \beta}{\alpha} \\
0 & \text{otherwise.}
\end{cases}
$$

(29)

**Case II:** If $k \beta / \alpha$ is not an integer: Let $m \triangleq \lfloor \frac{k \beta}{\alpha} \rfloor$. The constraints of the dual problem in (Dual-OP) for $i = m$ and $i = m + 1$ imply

$$
kr_1 + mr_2 \geq \frac{mk(k + 1)}{m + 1}
$$

(30)

$$
kr_2 + (m + 1)r_2 \geq \frac{(m + 1)k(k + 1)}{m + 2}.
$$

(31)

Hence, we have

$$
G^* = \alpha r_1^* + \beta r_2^* = \frac{(m + 1)\alpha - k \beta}{k} [kr_1^* + mr_2^*] + \frac{k \beta - m \alpha}{k} [kr_1^* + (m + 1)r_2^*]
$$

$$
\geq \frac{(m + 1)\alpha - k \beta}{k} \cdot \frac{mk(k + 1)}{m + 1} + \frac{k \beta - m \alpha}{k} \cdot \frac{(m + 1)k(k + 1)}{m + 2}
$$

(32)

$$
= \frac{k + 1}{m + 2} \left( m \alpha + \frac{k}{m + 1} \beta \right).
$$

(33)

where in (32) we lower bound the terms in square brackets using (30) and (31).

On the other hand, similar to (27), it is easy to see that

$$
\tilde{r}_1 = \frac{m(k + 1)}{m + 2} \quad \tilde{r}_2 = \frac{k(k + 1)}{(m + 1)(m + 2)}
$$

satisfy all the constraints in (Dual-OP). Hence, we have

$$
G^* = \min \alpha r_1 + \beta r_2 \leq \alpha \tilde{r}_1 + \beta \tilde{r}_2
$$

$$
= \frac{m(k + 1)}{m + 2} + \beta \frac{k(k + 1)}{(m + 1)(m + 2)}
$$

$$
= \frac{k + 1}{m + 2} \left( m \alpha + \frac{k}{m + 1} \beta \right).
$$

(34)

Comparing (33) and (34), we can conclude

$$
F^* = G^* = \frac{k + 1}{m + 2} \left( m \alpha + \frac{k}{m + 1} \beta \right).
$$

(35)

It is easy to see that $F^*$ can be achieved by setting

$$
\theta_i^* = \begin{cases} 
\frac{(i + 1)\alpha - k \beta}{\binom{\alpha}{i}} & i = \frac{k \beta}{\alpha} \\
\frac{k \beta - (i - 1)\alpha}{\binom{\alpha}{i}} & i = \frac{k \beta}{\alpha} \\
0 & \text{otherwise.}
\end{cases}
$$

(36)

Now we are ready to present the proof of Theorem 3.
C. Proof of Theorem 3

In Section VI-A we showed that optimization problems in $(\text{OPT} - \text{OB})$ is essentially equivalent to that in $(\text{OPT} - \text{OB}^*)$. On the other hand optimization problems in $(\text{OPT} - \text{OB}^*)$ and $(\text{OPT} - \text{IB})$ only differ in one constraint, where the equation $\sum_{j=1}^{k} \binom{k}{j} \theta_j = \alpha$ in $(\text{OPT} - \text{OB}^*)$ is relaxed to an inequality in $(\text{OPT} - \text{IB})$. However, the analysis in Section VI-B revealed the optimum choice of $\hat{\theta}^*$ as given in (29) and (36). It is easy to check that these optimum values satisfy the above-mentioned constraint with equality. This implies that the optimum solution of $(\text{OPT} - \text{OB}^*)$ (and therefore $(\text{OPT} - \text{OB})$) is identical to that of $(\text{OPT} - \text{IB})$, which is given in (35) as

$$F_k(\alpha, \beta) = \frac{k + 1}{m + 2} \left( m\alpha + \frac{k}{m + 1} \beta \right).$$

Hence, we have a pair of matching lower and upper bounds for the data size, which yield in a complete characterization for the linear capacity of a $(k + 1, k, k)$-DSS. This conclude the proof of Theorem 3.

VII. Conclusion

We studied the linear capacity of regeneration codes for a distributed storage systems with parameters $(n, k, d) = (k + 1, k, k)$, for any value of $k$. Linearity of the code allows us to represent all the random variables in terms of subspaces of a vector space generated by the data stored in the system. One wishes to maximize the dimension of such generic vector space in order to maximize the storage capacity of the system. As shown in this work, a set of constraints for the dimensions of the underlying subspaces can be obtained by exploiting the regeneration properties, as well optimality of the system. These convert the problem into an optimization problem, whose solution provides us with an upper bound for the storage capacity of the system.

Moreover, analysis of this optimization problem is insightful for code construction. Based on the necessary properties obtained for an optimum code, we proposed a code construction with guarantee for exact-regeneration. The proposed construction provides a whole family of codes for a system with given $(\alpha, \beta)$, among which we can pick the one with the maximum file size. This yields in another optimization problem over instances of the code construction, which provides a lower bound for the storage capacity. We subsequently showed that the two optimization problems have matching solutions, which characterizes the optimum trade-off of the system.

Exploring connections between vector spaces and linear codes, as well as identifying necessary conditions for optimality of a regeneration code resulted in promising results for a class of system parameters in this work. We expect the proposed approach to be further investigated in the near future, for distributed storage systems with arbitrary parameters. Another interesting direction for future work is to translate the linear algebraic analysis of this work into information-theoretic framework. This can potentially relax the linearity assumption of the code in this work, and provide the optimum trade-off for general codes.
**APPENDIX A**

**TWO MINIMAL REPRESENTATIONS**

We present the proof of Proposition 1 in this section. To this end, we first present a technique which allows us to reduce an exact-repair code, and obtain a code for a system with smaller number of nodes from an existing symmetric code for a \((k+1, k, k)\)-DSS.

**Proposition 10.** Let \(\mathcal{C} = \{\mathcal{W}_i; i \in [k+1]; S_{i\rightarrow j}, i, j \in [k+1]\}\) be an exact-regeneration code with characteristic vector \((\pi_1, \pi_2, \ldots, \pi_k)\) for a \((k+1, k, k)\)-LDSS, that operates at \((\alpha, \beta, F)\), and \(\mathcal{G} \subseteq \mathcal{F}\) be a vector space spanned by some vectors from the code. Then

\[
\mathcal{C}' = \{\mathcal{W}'_i, i \in [k+1]; S'_{i\rightarrow j}, i, j \in [k+1]\} = \left\{\mathcal{W}_i|_{\mathcal{mod}(\mathcal{G})}, i \in [k+1]; [S_{i\rightarrow j}]|_{\mathcal{mod}(\mathcal{G})}, i, j \in [k+1]\right\}
\]

is an (possibly heterogeneous and asymmetric) exact-regeneration code, that can store the vector space \(\mathcal{F}' = [\mathcal{F}]|_{\mathcal{mod}(\mathcal{G})}\) of dimension \(F' = F - \dim(G)\) in distributed manner. Moreover, the storage capacity required for node \(i\) is given by \(\alpha'_i = \alpha - \dim(W_i \cap G)\), and the repair bandwidth required from node \(i\) to node \(j\) is bounded by \(\beta'_{ij} = \beta - \dim(S_{i\rightarrow j} \cap G)\leq \beta\).

The following corollary is a special case of the proposition.

**Corollary 1.** Let \(\mathcal{C} = \{\mathcal{W}_i; i \in [k+1]; S_{i\rightarrow j}, i, j \in [k+1]\}\) be an exact-regeneration code for a \((k+1, k, k)\)-LDSS operating \((\alpha, \beta, F)\) which admits the characteristic vector \((\pi_1, \pi_2, \ldots, \pi_k)\). Then for any \(A \subseteq [k+1]\),

\[
\mathcal{C}' = \{\mathcal{W}'_i, i \in [k+1] \setminus A; S'_{i\rightarrow j}, i, j \in [k+1] \setminus A\} = \left\{\mathcal{W}_i|_{\mathcal{mod}(\mathcal{W}_A)}, i \in [k+1] \setminus A; [S_{i\rightarrow j}]|_{\mathcal{mod}(\mathcal{W}_A)}, i, j \in [k+1] \setminus A\right\}
\]

is a homogeneous and symmetric \((k - |A| + 1, k - |A|, k - |A|)\) exact-regeneration code, that can store \(\mathcal{F}' = [\mathcal{F}]|_{\mathcal{mod}(\mathcal{W}_A)}\) and operates at parameters \((\alpha', \beta', F')\), where \(\alpha' = \alpha - |A|\), \(\beta' \leq \beta\) and \(F' = F - \sum_{i=0}^{|A|-1}(\alpha - \pi_i)\).

Moreover, the characteristic vector of \(\mathcal{C}'\) is given by

\[
(\pi'_1, \pi'_2, \ldots, \pi'_{k-|A|}) = (\pi_{|A|+1} - \pi_{|A|}, \pi_{|A|+1} - \pi_{|A|}, \ldots, \pi_k - \pi_{|A|}).
\]

The proofs of the proposition and its corollary are resented in Appendix C.

A few more primary results will be used in the proof of the proposition. The following lemmas along with their proofs provide us with the required tools.

**Lemma 6.** Let \(\mathcal{C} = (\mathcal{W}_1, \ldots, \mathcal{W}_{k+1})\) be an exact-regeneration code with characteristic vector \((\pi_1, \ldots, \pi_k)\) for an \((n, k, d) = (k+1, k, k)\) system that operates at \((\alpha, \beta, F)\). Also assume all the nodes share a vector, i.e., that there exists some vector \(0 \neq v \in \mathcal{F}\) that \(v \in \mathcal{W}_i\) for \(i = 1, \ldots, k+1\). Then the code obtained by nulling \(v\) is an exact-regeneration code with parameters \((\alpha', \beta', F')\) where

(i) \(\alpha' = \alpha - 1\), \(F' = F - 1\), and \(\beta' = \beta - 1/k\),

(ii) \((\pi'_1, \ldots, \pi'_k) = (\pi_1 - 1, \ldots, \pi_k - 1)\)**
Proof of Lemma 6. Let \( v \in \mathcal{W}_i \) for \( i = 1, \ldots, k + 1 \), and define \( \mathcal{W}'_i = [\mathcal{W}_i]_{\text{mod}(v)} \). An argument similar to the proof of Proposition 10 can be used to show that the code obtained by nulling \( v \) is exact-repairable, which we skip for the sake of brevity. We have

\[
\alpha' = \dim(\mathcal{W}'_i) = \dim([\mathcal{W}_i]_{\text{mod}(v)}) = \dim(\mathcal{W}_i) - \dim(\mathcal{W}_i \cap v) \tag{a}
\]

\[
= \dim(\mathcal{W}_i) - \dim(v) = \alpha - 1, \quad i = 1, \ldots, k + 1,
\]

and

\[
F' = \dim(\mathcal{F}') = \dim([\mathcal{F}]_{\text{mod}(v)}) = \dim(\mathcal{F}) - \dim(\mathcal{F} \cap v) = \dim(\mathcal{F}) - \dim(v) = F - 1, \tag{37}
\]

where equalities indicated by (a) and (b) are due to the facts that \( v \in \mathcal{W}_i \) for \( i = 1, \ldots, k + 1 \), and \( v \in \mathcal{F} \), respectively.

In order to prove the last equality in (i), consider the repair process for a node, say \( \mathcal{W}_{k+1} \) in \( \mathcal{C} \). Since \( v \in \mathcal{W}_{k+1} \), it should be repaired by at least one vector received from helpers, which is not required to be sent in the reduced code \( \mathcal{C}' \). This can reduce the total repair bandwidth by one unit, which implies \( k\beta' = k\beta - 1 \). More precisely, \( S'_{i \rightarrow k+1} = [S_{i \rightarrow k+1}]_{\text{mod}(v)} \), the repair data sent to node \( i \) in the new system satisfies

\[
\dim \left( \sum_{i=1}^{k} S'_{i \rightarrow k+1} \right) = \dim \left( \sum_{i=1}^{k} [S_{i \rightarrow k+1}]_{\text{mod}(v)} \right) \tag{c}
\]

\[
\equiv \dim \left( \sum_{i=1}^{k} S_{i \rightarrow k+1} \right) - \dim \left( \left( \sum_{i=1}^{k} S_{i \rightarrow k+1} \right) \cap \{v\} \right) \tag{d}
\]

\[
= \dim \left( \sum_{i=1}^{k} S_{i \rightarrow k+1} \right) - \dim(v) \tag{e}
\]

\[
= k \dim(S_{i \rightarrow k+1}) - \dim(v) = k \beta - 1. \tag{f}
\]

Here (c) and (d) are followed from (P.2) and (P.4) in Lemma 2, respectively; in (e) we used the fact that since \( v \in \mathcal{W}_{k+1} \subseteq \sum_{i=1}^{k} S_{i \rightarrow k+1} \), and finally (f) holds since subspaces \( S_{i \rightarrow k+1} \) are all disjoint in an optimum system.

This equality implies that \( k\beta - 1 \) vectors suffice to span the repair vector space \( \sum_{i=1}^{k} S'_{i \rightarrow k+1} \), i.e., \( (k - 1) \) nodes should send \( \beta \) vectors, and one helper node sends only \( \beta - 1 \) repair vectors. After symmetrizing the system (see Lemma 1), we have \( \beta' = (k\beta - 1)/k = \beta - 1/k \).
In order to prove (ii), we can start with a node $\mathcal{W}_i$ and a set of nodes $A$ where $i \notin A$. We have

$$\pi'_{|A|} = \dim (\mathcal{W}_i' \cap \mathcal{W}_A') = \dim \left( [\mathcal{W}_i']_{\mod(v)} \cap [\mathcal{W}_A']_{\mod(v)} \right)$$

$$= \dim \left( [\mathcal{W}_i']_{\mod(v)} \right) + \dim \left( [\mathcal{W}_A']_{\mod(v)} \right) - \dim \left( [\mathcal{W}_i']_{\mod(v)} + [\mathcal{W}_A']_{\mod(v)} \right)$$

$$= \dim \left( [\mathcal{W}_i']_{\mod(v)} \right) + \dim \left( [\mathcal{W}_A']_{\mod(v)} \right) - \dim \left( [\mathcal{W}_i + \mathcal{W}_A]_{\mod(v)} \right)$$

$$(d) \quad = (\dim (\mathcal{W}_i) - \dim (v)) + (\dim (\mathcal{W}_A) - \dim (v)) - (\dim (\mathcal{W}_i + \mathcal{W}_A) - \dim (v))$$

$$= (\dim (\mathcal{W}_i) + \dim (\mathcal{W}_A) - \dim (\mathcal{W}_i + \mathcal{W}_A)) - \dim (v)$$

$$= \dim (\mathcal{W}_i \cap \mathcal{W}_A) - \dim (v)$$

$$= \pi_{|A|} - 1.$$  

Equality in (d) is due the fact that $c \in \mathcal{W}_i$ for $i = 1, \ldots, k + 1$. This completes the proof of the lemma. □

Next, we propose a reduction algorithm for a code including a vector with more than two minimal representations.
Lemma 7 summarizes the properties of this algorithm.

**Input:** $\mathcal{C} = (W_1, \ldots, W_{k+1})$, $v$, $A_1, \ldots, A_t$ (minimal representations for $v$)

**Output:** $\tilde{\mathcal{C}} = (\tilde{W}_1, \ldots, \tilde{W}_{m+1})$, $\tilde{v}$

**Initialization:**

for $i \leftarrow 1$ to $k + 1$ do

$W_i^{(0)} \leftarrow W_i$

end

$v^{(0)} \leftarrow v$

**Disjoint Representations:**

for $i \leftarrow 1$ to $k + 1$ do

$W_i^{(1)} \leftarrow \left[ W_i^{(0)} \right]_{\mod (\sum_{\ell \in A_2 \cap A_3} W_\ell^{(0)})}$

end

$v^{(1)} \leftarrow \left[ v^{(0)} \right]_{\mod (\sum_{\ell \in A_2 \cap A_3} W_\ell^{(0)})}$

$j \leftarrow 1$

**Removing Partial Contributors:**

while $\exists \ell$ such that $W_\ell^{(j)} \neq \{0\}$ and $v^{(j)} \notin W_\ell^{(j)}$ do

for $i \leftarrow 1$ to $k + 1$ do

$W_i^{(j+1)} \leftarrow \left[ W_i^{(j)} \right]_{\mod (W_\ell^{(j)})}$

end

$v^{(j+1)} \leftarrow \left[ v^{(j)} \right]_{\mod (W_\ell^{(j)})}$

$j \leftarrow j + 1$

end

**Relabeling:**

$m \leftarrow 1$

for $i \leftarrow 1$ to $k + 1$ do

if $W_i^{(j)} \neq \{0\}$ then

$\tilde{W}_m \leftarrow W_i^{(j)}$

$m \leftarrow m + 1$

end

end

$\tilde{v} \leftarrow v^{(j)}$

**Algorithm 1:** Code reduction for a code including a vector with more than two minimal representations.

Lemma 7. Let $\mathcal{C} = (W_1, \ldots, W_{k+1})$ be an exact-regeneration code with parameters $(\alpha, \beta, F)$, and (without loss of generality) $0 \neq v \in F$ be a vector in $W_1$ with $t \geq 3$ minimal representations, namely $A_1 = \{1\}, A_2, A_3, \ldots, A_t$. Running Algorithm 1 on $\mathcal{C}$, we obtain a code $\tilde{\mathcal{C}} = (\tilde{W}_1, \ldots, \tilde{W}_{m+1})$ and vector $\tilde{v}$, where
Proof of Lemma 7. First recall from (P.3) in Lemma 2 that nulling is an associative operation, and the effect of the algorithm is equivalent to nulling \((k - m)\) nodes in the code. Hence, (i) and (ii) are direct consequences of Corollary 1 with \(|A| = k - m\).

In order to prove (iii) first note that \(v(0) = v \neq \{0\}\) by assumption. During the disjoint representation step, we null all the nodes in \(A_2 \cap A_3\). Recall that \(A_2\) and \(A_3\) are two distinct minimal representations, and hence \(A_2 \cap A_3\) is a proper (strict) subset of them, which does not include \(v(0)\) by definition. Hence, \(v(1) = \left[\sum_{i \in A_2 \cap A_3} W_i(0)\right]_{\text{mod}(\sum_{i \in A_2 \cap A_3} W_i(0))}\) is non-zero. Now, let \(v(j) \neq \{0\}\) for some \(j\), and \(\ell\) be the index of the node we null in step \(j\). If \(v(j+1) = \left[v(j)\right]_{\text{mod}(\sum_{i \in A_2 \cap A_3} W_i(0))} = 0\), then we have \(v(j) \in W_\ell(j)\), which is in contradiction with the criteria to choose \(\ell\) in the while loop, that is \(v(j) \not\subseteq W_\ell(j)\).

In order to show (iv), note that the while loop in the algorithm only stops when every node is either nulled or contain the current vector \(v(j)\). Since \(v(j) \neq 0\) remains in the system, then all the nodes cannot be nulled, and \(v(j)\) is in the span of all the surviving (non-zero) nodes. In other words, \(\{i\}\) is a minimal representation for \(\tilde{v}\) for \(i = 1, \ldots, m + 1\).

Finally, in order to prove (v), we first observe that by the end of disjoint representations section, we guarantee existence of three disjoint representations. More precisely, \(v(0) \in \sum_{i \in A_2} W_i(0)\) implies \(v(1) = \left[\sum_{i \in A_2} W_i(0)\right]_{\text{mod}(\sum_{i \in A_2} W_i(0))} = \sum_{i \in A_2} W_i(1)\).

Similarly, we have \(v(1) \in \sum_{i \in A_3 \setminus A_2} W_i(1)\). Hence, \(A_2 \setminus A_3\) and \(A_3 \setminus A_2\) are two disjoint representations for \(v(1)\). Since every representation includes at least one minimal representation, there exist subsets \(A'_2 \subseteq A_2 \setminus A_3\) and \(A'_3 \subseteq A_3 \setminus A_2\) which are both minimal representations for \(v(1)\). These together with \(A_1\) form three mutually disjoint minimal representations.

These sets can be only shrunk during the rest of the algorithm until every minimal representation becomes a single node set. That is, at the termination of the algorithm, we have (at least) three distinct minimal representations \(\{1\}, \{x\}\) and \(\{y\}\) for \(\tilde{v}\), where \(x \in A_2 \setminus A_3\) and \(y \in A_3 \setminus A_2\). Thus, the number of non-zero nodes in \(\tilde{C}\) at least three. This completes the proof of Lemma 7.

Now, we are ready to prove the Proposition.
Proof of Proposition 1. Let $\mathcal{C} = (\mathcal{W}_1, \ldots, \mathcal{W}_{k+1})$ be a pareto-optimum exact-regeneration code with parameters $(\alpha, \beta, F)$ and characteristic vector $(\pi_1, \ldots, \pi_k)$ which includes a vector $v \in \mathcal{W}_1$ with more than two minimal representations. First note from Lemma 5, for every feasible characteristic vector we have $\pi_{k-1} \geq \alpha - \beta$. Moreover, in order to maximize $F$ one should set $\pi_{k-1}$ to its minimum value, which implies in a pareto-optimum code we have $\pi_{k-1} = \alpha - \beta$.

We first apply the operations of Lemma 7 on the code to obtain $\tilde{\mathcal{C}} = (\tilde{\mathcal{C}}_1, \ldots, \tilde{\mathcal{C}}_{m+1})$ followed by the operation of Lemma 6 which generates code $\tilde{\mathcal{C}}' = (\tilde{\mathcal{W}}'_1, \ldots, \tilde{\mathcal{W}}'_{m+1})$. We denote the parameters of the latter code by $(\tilde{\alpha}', \tilde{\beta}', \tilde{F}')$ where by Lemma 7 and Lemma 6 we have

$$\tilde{\alpha}' = \alpha - \pi_{k-m} - 1, \quad \tilde{\beta}' \leq \beta - 1/m, \quad \tilde{F}' = F - \sum_{i=0}^{k-m-1} (\alpha - \pi_i) - 1.$$

Furthermore, the same pair of lemmas imply that the characteristic vector of $\tilde{\mathcal{C}}'$ is given by

$$(\tilde{\pi}'_1, \tilde{\pi}'_2, \ldots, \tilde{\pi}'_m) = (\pi_{k-m+1} - \pi_{k-m}, \pi_{k-m+2} - \pi_{k-m}, \ldots, \pi_k - \pi_{k-m-1}).$$

We have

$$\tilde{F}' = F - \sum_{i=0}^{k-m-1} (\alpha - \pi_i) - 1 = \sum_{i=0}^{k-1} (\alpha - \pi_i) - \sum_{i=0}^{k-m-1} (\alpha - \pi_i) - 1$$

$$= \sum_{i=k-m}^{k-1} (\alpha - \pi_i) - 1 = \sum_{i=1}^{m-1} (\alpha - \pi_{k-m+i}) + (\alpha - \pi_{k-m})$$

$$= \sum_{i=1}^{m-1} (\tilde{\alpha}' - \tilde{\pi}'_i) + \tilde{\alpha}' = \sum_{i=0}^{m-1} (\tilde{\alpha}' - \tilde{\pi}'_i).$$

where we set $\tilde{\pi}'_0 = 0$ in the last equality. More importantly, we have

$$\tilde{\pi}'_{m-1} = \pi_{k-1} - \pi_{k-m} - 1 = \alpha - \beta - \pi_{k-m} - 1 \leq \tilde{\alpha}' - \tilde{\beta}' - 1/m.$$

This implies $\tilde{\pi}'_{m-1} < \tilde{\alpha}' - \tilde{\beta}'$. On the other hand, since $\tilde{\mathcal{C}}'$ is an exact-regeneration code, its characteristic vector should be feasible and satisfy the constraints of Lemma 5. However, the latter inequality is in contradiction with $\tilde{\pi}'_{m-1} \geq \tilde{\alpha}' - \tilde{\beta}'$ in Lemma 5. Therefore, our initial assumption regarding optimality of the $\mathcal{C}$ does not hold. This completes the proof of the proposition.

\[\square\]

APPENDIX B

PROOF OF PROPOSITION 2

The following definition will be used in the proof of Proposition.

Definition 12. For a pair $(i, A)$ with $A \subset \{1, \ldots, k+1\}$ and $i \in \{1, \ldots, k+1\} \setminus A$, we define

$$\mathcal{T}(i, A) = [\mathcal{W}_i \cap \mathcal{W}_A]_{\text{mod}(\sum_{X \subseteq A (\mathcal{W}_i \cap \mathcal{W}_X)})}.$$
This is a subspace spanned by vectors in \( W_i \) whose minimal representation is \( A \). We denote the dimension of this subspace by \( \theta_{|A|} = \dim(\mathcal{T}(i, A)) \).

We first present a few lemmas regarding properties of \( \mathcal{T}(i, A) \) along with their proofs.

**Lemma 8.** For every pair \((i, A)\) with \( i \notin A \), we have \( \mathcal{T}(i, A) \subseteq (W_i \cap W_A) \).

*Proof.* Note that for every \( X \subseteq A \), we have \( W_X \subseteq W_A \), which implies \((W_i \cap W_X) \subseteq (W_i \cap W_A)\). Thus,

\[
\sum_{X \subseteq A} (W_i \cap W_X) \subseteq (W_i \cap W_A),
\]

i.e., the nulled space is a subspace of \( W_i \cap W_A \), and thus \((P.5)\) in Lemma 2 implies that the quotient \( \mathcal{T}(i, A) \) remains as a subspace of \( W_i \cap W_A \). \(\square\)

**Lemma 9.** Let \( C \) be an exact-regeneration code such that for every \( i \), each vector \( v \in W_i \) has exactly two minimal representation. Then for every \( i \) and family of distinct subsets \( \{A, B_1, B_2, \ldots, B_t\} \) with \( i \notin A \cup B_1 \cup \cdots \cup B_t \), subspaces \( \mathcal{T}(i, A) \) and \( \sum_{j=1}^t \mathcal{T}(i, B_j) \) are disjoint.

*Proof.* We prove the claim by contradiction. Let \( 0 \neq w \in \mathcal{T}(i, A) \cap \sum_{j=1}^t \mathcal{T}(i, B_j) \). Therefore \( w \) can be decomposed to \( w = \sum_{j=1}^t v_j \) such that \( v_j \in \mathcal{T}(i, B_j) \), i.e., \( B_j \) is a minimal representation for \( v_j \). Let \( B = \bigcup_{j=1}^t B_j \). From \( w = \sum_{j=1}^t v_j \), we can conclude \( w \in W_B \), and hence \( B \) is a representation for \( w \). Note that \( w \) has only two minimal representations \( \{i\} \) and \( A \), and since \( \{i\} \notin B \), we have \( A \subseteq B \). Two cases can be identified as follows.

First assume \( A = B \). This implies \( B_j \subseteq A \), and since \( B_j \)'s are distinct from \( A \), we have \( B_j \subseteq A \) for \( j = 1, \ldots, t \). Thus,

\[
w = \sum_{j=1}^t v_j \in \sum_{j=1}^t (W_i \cap W_{B_j}) \subseteq \sum_{X \subseteq A} (W_i \cap W_X).
\]

Hence \( w \) should have been nulled in \( \mathcal{T}(i, A) \), which is in contradiction with \( w \in \mathcal{T}_{i,A} \).

Next assume \( A \not\subseteq B \), and there exist some \( 1 \leq s \leq t \), such that \( B_s \not\subseteq A \). Let \( B'_s = B_s \cap A \), and \( B' = \bigcup_{j \neq s} B_j \cup B'_s \). Since \( A \subseteq B' \) we have \( w \in W_A \subseteq W_{B'} \). Moreover,

\[
v_s = w - \sum_{j \neq s} v_j \in W_{B'} + \sum_{j \neq s} W_{B_j} = W_{B'}.
\]

Therefore, \( B' \) is a representation for \( v_s \). Since \( \{i\} \not\subseteq B' \) and \( B_s \not\subseteq B' \), \( B' \) includes a third minimal representation for \( v_s \), which is in contradiction with the assumption of the lemma. \(\square\)

**Lemma 10.** For every \((A, i)\) with \( i \notin A \), we have \( \sum_{B \subseteq A} \mathcal{T}(i, B) = W_i \cap W_A \).

*Proof.* We prove the identity in Lemma 10 by induction on the size of set \( A \). If \( |A| = 1 \), then \( A \) has no non-empty proper subset, and hence no nulling is performed in \( \mathcal{T}(i, A) \). Thus,

\[
\mathcal{T}(i, A) = [W_i \cap W_A]_{\mod\{0\}} = W_i \cap W_A.
\]

\(^4\)The symmetry of the system implies that this dimension does not depend on the specific realization of \( A \) or \( i \), and only depends on \(|A|\).
Now, assume the claim holds for every set of size at most \( \ell \), and we wish to prove it for some \( A \) with \( |A| = \ell + 1 \).

\[
\sum_{X \subseteq A} T(i, X) = T(i, A) + \sum_{X \subset A} T_{X,i}
\]

\[= \sum_{X \subseteq A} T(i, A) + \sum_{Y \subseteq X} T_{Y,i} \]

\[= T(i, A) + \sum_{X \subset A} (W_i \cap W_X) \]

\[= [W_i \cap W_A]_{\text{mod}}(\sum_{X \subseteq A} (W_i \cap W_X)) + \sum_{X \subset A} (W_i \cap W_X) \]

\[= W_i \cap W_A \]

where (a) holds since \( U + U = U \) for any subspace \( U \), and hence we can add arbitrary number of any subspace existing in the summation; next we used the fact that for every \( X \subset A \), we have \( |X| \leq |A| - 1 = \ell \), and hence the equality in (b) holds based on the induction assumption; finally, we used the fact that \( \sum_{X \subset A} (W_i \cap W_X) \subseteq (W_i \cap W_A) \) and identity (P.5) from Lemma 2 to conclude the equality in (c). This completes the proof.

Now we are ready to prove Proposition 2.

**Proof of Proposition 2.** We can start by definition of \( \pi_{|A|} \) for some \( A \subseteq [k + 1] \setminus \{i\} \), and write

\[
\pi_{|A|} = \dim (W_i \cap W_A)
\]

\[= \dim \left( \sum_{B \subseteq A} T(i, B) \right) \]

\[= \sum_{B \subseteq A} \dim (T(i, B)) \]

\[= \sum_{B \subseteq A} \theta_{|B|} = \sum_{j=1}^{|A|} \binom{|A|}{j} \theta_j \quad (38)\]

where in (a) holds due to Lemma 10, and equality in (b) is implied the fact that subspaces \( T_{B,i} \) are all disjoint due to Lemma 9. This equality holds for every \( A \), with \( |A| = 1, \ldots, k \). Rewriting (38) for all values \( |A| \) in matrix form, we obtain

\[
\begin{bmatrix}
\pi_0 \\
\pi_1 \\
\pi_2 \\
\vdots \\
\pi_k
\end{bmatrix} =
\begin{bmatrix}
\binom{0}{0} & 0 & \cdots & 0 & 0 \\
\binom{1}{1} & 0 & \cdots & 0 & 0 \\
\binom{2}{1} & \binom{2}{2} & \cdots & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
\binom{k}{0} & \binom{k}{1} & \binom{k}{2} & \cdots & \binom{k}{k}
\end{bmatrix}
\begin{bmatrix}
\theta_0 \\
\theta_1 \\
\theta_2 \\
\vdots \\
\theta_k
\end{bmatrix}.
\]

(39)
Finally, by multiplying both sides by the inverse of the square matrix, we get
\[
\begin{pmatrix}
-1^0(0) & 0 & \ldots & 0 & 0 \\
-1^1(0) & -1^0(1) & 0 & \ldots & 0 \\
-1^2(0) & -1^1(1) & -1^0(2) & \ddots & 0 \\
\vdots & \vdots & \vdots & \ddots & 0 \\
-1^k(0) & -1^{k-1}(1) & -1^{k-2}(2) & \ldots & -1^0(k)
\end{pmatrix}
\begin{pmatrix}
\pi_0 \\
\pi_1 \\
\pi_2 \\
\vdots \\
\pi_k
\end{pmatrix}
= \begin{pmatrix}
\theta_0 \\
\theta_1 \\
\theta_2 \\
\vdots \\
\theta_k
\end{pmatrix}
\geq 0,
\tag{40}
\] where the last inequality is due to the fact that \( \theta_i \) is dimension of some subspace (see Definition 12), and hence \( \theta_i \geq 0 \), for all values of \( i \). Each row of the latter matrix inequality implies condition (C3) for some value of \( i \). \( \square \)

**APPENDIX C**

**CODE REDUCTION BY NULLING**

**Proof of Proposition 10.** The proof of this proposition is simply based on the linearity of the code and the properties of nulling operation. To see data recovery property, let \( A \subseteq \{k+1\} \) with \( |A| = k \). Since \( C \) is a \((k+1, k, k)\) exact-regeneration code, we have \( F' \subseteq \sum_{i \in A} \mathcal{W}_i \). Hence,
\[
F' = [F]_{\text{mod}(\mathcal{G})}^{(a)} = \sum_{i \in A} [\mathcal{W}_i]_{\text{mod}(\mathcal{G})}^{(b)} = \sum_{i \in A} \mathcal{W}_i',
\]
were \((a)\) and \((b)\) are followed from Properties (P.1) and (P.2) in Lemma 2, respectively.

The exact-regeneration property of \( C \) implies \( \mathcal{W}_j \subseteq \sum_{i \neq j} S_{i \rightarrow j} \). This implies
\[
\mathcal{W}_j' = [\mathcal{W}_j]_{\text{mod}(\mathcal{G})}^{(a)} = \sum_{i \neq j} S_{i \rightarrow j}^{(b)} = \sum_{i \neq j} S_{i \rightarrow j}'.
\]

Here again we used Lemma 2 in \((a)\) and \((b)\). This shows that \( C' \) is an exact-regeneration code.

The parameters of the code are given by
\[
\dim (F') = \dim ([F]_{\text{mod}(\mathcal{G})}) \overset{(c)}{=} \dim (F) - \dim (F \cap \mathcal{G}) = \dim (F) - \dim (\mathcal{G}) = F - \dim (\mathcal{G}),
\]
where \((c)\) is followed from (P.4) in Lemma 2. Similarly we have
\[
\alpha_i' = \dim (\mathcal{W}_i') = \dim ([\mathcal{W}_i]_{\text{mod}(\mathcal{G})}) = \dim (\mathcal{W}_i) - \dim (\mathcal{W}_i \cap \mathcal{G}) = \alpha - \dim (\mathcal{W}_i \cap \mathcal{G})
\]
\[
\beta_{ij}' = \dim (S_{i \rightarrow j}') = \dim ([S_{i \rightarrow j}]_{\text{mod}(\mathcal{G})}) = \dim (S_{i \rightarrow j}) - \dim (S_{i \rightarrow j} \cap \mathcal{G}) = \beta - \dim (S_{i \rightarrow j} \cap \mathcal{G}).
\]
This completes the proof. \( \square \)

**Proof of Corollary 1.** The claim of this corollary is essentially the same as that of Proposition 10 specialized to \( \mathcal{G} = \mathcal{W}_A = \sum_{i \in A} \mathcal{W}_i \). The data recovery and exact-regeneration properties are immediate consequences of Proposition 10. Moreover, similar to the proof of Lemma 3 we have \( \dim (\mathcal{W}_A) = \sum_{i=0}^{\lfloor A/2 \rfloor - 1} (\alpha_i - \pi_i) \). From this we can obtain the claimed value for \( F' = F - \dim (\mathcal{W}_A) \).
For $i \in A$ we simply have $W'_i = \{0\}$ and $\alpha'_i = 0$, i.e., $W_i$ is fully nulled. On the other hand for $i \in [k + 1] \setminus A$ we have

$$\alpha'_i = \alpha - \dim (W_i \cap W_A) = \alpha - \pi_{|A|}. $$

Hence, after eliminating nodes with index in $A$, all the remaining nodes store the same amount of data.

Similarly, symmetry of $C$ implies that $\beta'_{ij} = \beta - \dim (S_{i\to j} \cap W_A)$ is constant for any pair $i, j \notin A$. It is also clear that $\beta'_{ij} = 0$ if either $i$ or $j$ are in $A$. Therefore nodes in $[k + 1] \setminus A$ form a homogeneous and symmetric code with a total of $(k + 1) - |A|$ nodes.

In order to determine the parameters of characteristic vector of $C'$, we can start with some $B$ that is disjoint from $A$ and $i$. We claim:\footnote{Note that nulling is not distributive over intersection in general.}

$$W'_i \cap W'_B \triangleq [W_i]_{mod(W_A)} \cap [W_B]_{mod(W_A)} = [W_i \cap W_{A\cup B}]_{mod(W_A)}. \tag{41}$$

In order to prove this claim, first consider some $v \in [W_i \cap W_{A\cup B}]_{mod(W_A)}$. By definition, there exists some $u \in W_A$ such that $u + v \in W_i \cap W_{A\cup B}$. Nulling $W_A$ (and in particular $u$), we get $v \in [W_i]_{mod(W_A)}$ and $v \in [W_{A\cup B}]_{mod(W_A)} = [W_B]_{mod(W_A)}$. This shows $[W_i \cap W_{A\cup B}]_{mod(W_A)} \subseteq [W_i]_{mod(W_A)} \cap [W_B]_{mod(W_A)}$.

Next, consider an arbitrary $v \in [W_i]_{mod(W_A)} \cap [W_B]_{mod(W_A)}$. This implies existence of $u_1, u_2 \in W_A$ such that $v + u_1 \in W_i$ and $v + u_2 \in W_B$. Hence, $(u_1 - u_2) \in W_A \subseteq W_{A\cup B}$, and therefore,

$$v + u_1 = (v + u_2) + (u_1 - u_2) \in W_{A\cup B}.$$ 

Thus, $v + u_1 \in W_i \cap W_{A\cup B}$, which immediately implies

$$v = [v + u_1]_{mod(W_A)} \in [W_i \cap W_{A\cup B}]_{mod(W_A)}.$$ 

Therefore, $[W_i]_{mod(W_A)} \cap [W_B]_{mod(W_A)} \subseteq [W_i \cap W_{A\cup B}]_{mod(W_A)}$. This proves the claim in (41). Hence,

$$\pi'_{|B|} = \dim (W'_i \cap W'_B) \overset{(a)}{=} \dim \left( [W_i \cap W_{A\cup B}]_{mod(W_A)} \right) \overset{(b)}{=} \dim (W_i \cap W_{A\cup B}) - \dim (W_i \cap W_{A\cup B} \cap W_A) \overset{(c)}{=} \dim (W_i \cap W_{A\cup B}) - \dim (W_i \cap W_A) \overset{(d)}{=} \pi_{|A\cup B|} - \pi_{|A|}.$$ 

Here $(a)$ is followed from (41); we used (P4) in $(b)$; equality in $(c)$ holds since $W_A \subseteq W_{A\cup B}$; and $(d)$ follows from that fact that $A$ and $B$ are disjoint. This holds for any $|B| = 1, 2, \ldots, k - |A|$, and determines the characteristic vector of the reduced code.
REFERENCES


