Linear Programming Bounds for Storage Codes

Terence H. Chan
Institute for Telecommunications Research,
University of South Australia
terence.chan@unisa.edu.au

Mohammad Ali Tebbi
Institute for Telecommunications Research,
University of South Australia
mohammad_ali.tebbi@mymail.unisa.edu.au

Chi Wan Sung
Department of Electronic Engineering,
City University of Hong Kong
albert.sung@cityu.edu.hk

***Abstract*—Extending Delsarte's linear programming bound for error correcting codes, this paper obtains a linear programming bound for locally repairable storage codes. The number of variables involved in the bound scales linearity with the size of the code. The bound can also be viewed as a necessary condition for the existence of a storage code and be used to characterise the tradeoff among the costs for storage, repair and update.**

I. Introduction

In a data storage network, data are usually stored in several data centres, which can be geographically separated. Each data centre can store a massive amount of data, by using hundreds of storage disks which are prone to failures. Therefore, a recovery mechanism must be in place to handle these failures.

Protection against failures are usually achieved by adding redundancies. Replication, where data is replicated and stored in multiple data centres, is the simplest example and is also widely adopted in many existing storage systems. Clearly, if one of the data centre failed, then the content stored can be restored from its replica stored elsewhere. While replication increases data reliability, the storage cost is also very high.

To reduce the cost, erasure-resilient coding can be used to construct storage codes. The idea is very simple. Consider a data packet consisting of k symbols. Then one can use a maximum distance separable (MDS) code to encode the data packet into n coded symbols. Each coded symbol will then be stored in one of the n data centres. If the symbol (i.e., the content) stored in a data centre is lost (i.e., erased), then we can regenerate the lost symbol from any k coded symbols stored in the surviving data centres.

While these erasure-resilient coding based storage code can minimise the storage cost, the cost (in terms of amount of information transmitted from other surviving data centres) to recover the failed disk or data centres can be very large. Hence, the challenge is to design a storage network that offers a proper tradeoff between costs for storage and repair. In the second part of the paper, we will also consider the cost for update. In [1], a new storage code (based on network coding) called *regenerating codes* was proposed. Data centres are allowed to perform coding (e.g., by sending the sum of the contents of all the disks stored there) in the regenerating process. It was shown in [1] that coding can reduce repair cost.

Many existing storage codes in literature assumed that lost data in a failed data centre can be regenerated from any subset (of size greater than a certain threshold) of surviving data centres. However, this requirement can be too restrictive. The concept of "code locality" was proposed in [2], which relaxes the repairing requirement so that any failed data centre can be repaired by at least one set (of size greater than the threshold) of surviving data centres. Relaxing the requirement can greatly reduce the costs for storage and repair.

Many storage codes in literature assumed that lost data in a failed data centre can be regenerated from any subset (of size greater than a certain threshold) of surviving data centres. This requirement can be too restrictive and sometimes not necessary. The concept of "code locality" was proposed in [2], which relaxes the repairing requirement so that any failed data centre can be repaired by at least one set (of size greater than the threshold) of surviving data centres. Relaxing the requirement can greatly reduce the costs for storage and repair. In [3], the tradeoffs among code locality, Hamming distance, and redundancy of the storage codes was investigated. It was also shown in [2] that codes with very small locality can be constructed with an excess storage overhead. In [4], [5], codes with multiple repair alternatives were proposed, so that repair is feasible even if one of the nodes in the "repair set" is not available. The tradeoff between code locality and the number of repair alternatives was studied in [5]. This paper focuses on computable bounds for locally repairable storage codes. We consider not only the tradeoff between repair cost and storage cost, but also the cost to update.

Notations: Let $\mathcal{N} = \{1, \ldots, n\}$ and $2^{\mathcal{N}}$ be the collection of all subsets of $\mathcal{N}$. It is often useful to represent a subset α of $\mathcal{N}$ as a length n binary vector $\mathbf{r} = [r_1, \ldots, r_n]$ such that $r_i = 1$ if and only if $i \in \alpha$. Under our convention, we can write $\mathbf{r} \subseteq \mathcal{N}$, and the empty set $\emptyset$ and the zero vector $\mathbf{0}$ can be used interchangeably. We will also not differentiate a singleton (a set containing only one element e.g., $\{i\}$) from the element (i in our example) itself.

II. Locally Repairable Codes

A. Linear Storage Codes

Let $\mathbb{F}_q$ be the finite field of size q and

$$\mathbb{F}_q^n = \{[z_1, \ldots, z_n] : \ z_i \in \mathbb{F}_q \text{ for all } i \in \mathcal{N}\}$$

be an n dimensional vector space over $\mathbb{F}_q$. A linear code $\mathcal{C}$ (of length n) over $\mathbb{F}_q$ is a subspace of $\mathbb{F}_q^n$, and can be specified by a $k \times n$ *generator matrix* G such that $\mathcal{C}$ is the collection of all vectors spanned by the rows of G. Here, k is the dimension

978-1-4799-0434-1/13/$31.00 ©2013 IEEE

of the subspace $\mathcal{C}$. Let $\mathbf{g}_i$ be the i^{th} column of G. Then the matrix G (and hence also the code) can also be defined by these "generating columns" $(\mathbf{g}_i, i \in \mathcal{N})$.

Alternatively, a linear code can also be specified by an $(n-k)\times n$ *parity check matrix* H such that $[c_1,\ldots,c_n] \in \mathcal{C}$ if and only if $[c_1,\ldots,c_n]H^\top = \mathbf{0}$. By regarding H as a generator matrix, the parity check matrix H also defines a linear code, called the *dual code* of $\mathcal{C}$ and is denoted by $\mathcal{C}^\perp$. As we shall see, the code $\mathcal{C}$ and its dual $\mathcal{C}^\perp$ are closely related.

Definition 1 (Supports): For any $\mathbf{c} = [c_1,\ldots,c_n] \in \mathbb{F}_q^n$, we define its *support* (denoted by $\lambda(\mathbf{c})$) as the subset of $\mathcal{N}$ such that $i \in \lambda(\mathbf{c})$ if and only if $c_i \neq 0$. The support enumerator of $\mathcal{C}$ is a function $\Lambda_\mathcal{C} : 2^\mathcal{N} \mapsto \mathbb{R}$ such that

$$\Lambda_\mathcal{C}(\mathbf{r}) \triangleq |\{\mathbf{c} \in \mathcal{C} : \lambda(\mathbf{c}) = \mathbf{r}\}|,\ \forall \mathbf{r} \subseteq \mathcal{N}.$$

In other words, $\Lambda_\mathcal{C}(\mathbf{r})$ is the number of codewords in $\mathcal{C}$ whose support is exactly $\mathbf{r}$.

Proposition 1 (Duality [6]): Let $\mathcal{C}$ be a length n linear code over $\mathbb{F}_q$. Then for all subsets $\mathbf{r} = [r_1,\ldots,r_n] \subseteq \mathcal{N}$,

$$\Lambda_{\mathcal{C}^\perp}(\mathbf{r}) = \frac{1}{|\mathcal{C}|}\sum_{\mathbf{s}=[s_1,\ldots,s_n]\in 2^\mathcal{N}} \Lambda_\mathcal{C}(\mathbf{s})\prod_{j=1}^n \kappa_q(s_j,r_j) \geq 0 \quad (1)$$

where

$$\kappa_q(s,r) = \begin{cases} 1 & \text{if } r = 0 \\ q-1 & \text{if } s = 0 \text{ and } r = 1 \\ -1 & \text{otherwise.}\end{cases}$$

Now, consider a distributed storage network, which consists of n data centres. Let $(X_1,\ldots,X_n)$ be the contents respectively stored in the n data centres. Let $\mathcal{C}$ be a linear code of dimension k specified by a parity check matrix H. It defines a distributed storage code in the sense that $(X_1,\ldots,X_n)$ must form a codeword in $\mathcal{C}$. Using the code, each data centre will store at most $\log_2 q$ number of bits, and the whole storage network can store $k\log_2 q$ number of information bits.

Suppose $\mathbf{h} = [h_1,\ldots,h_n] \in \mathcal{C}^\perp$. Then $\sum_{j\in\lambda(\mathbf{h})} X_j h_j = 0$. Therefore, if data centre i fails and $i \in \lambda(\mathbf{h})$, then the content X_i stored there can be regenerated from the content X_j stored in data centres j for $j \in \lambda(\mathbf{h}) \setminus i$. To be precise,

$$X_i = -h_i^{-1}\left(\sum_{j\in\lambda(\mathbf{h}) \text{ and } j\neq i} X_j h_j\right).$$

In this case, the data recovery process requires a total of $(|\lambda(\mathbf{h})|-1)\log_2 q$ bits to be transmitted from the surviving data centres, indexed by $\lambda(\mathbf{h})\setminus i$.

Definition 2: A q-ary linear (α,β) storage code $\mathcal{C}$ of length n is a linear code over $\mathbb{F}_q$ satisfying the following criteria:

1) (Single Failure Recovery, SFR) for any $i \in \mathcal{N}$, there exists $\mathbf{h} \in \mathcal{C}^\perp$ such that $i \in \lambda(\mathbf{h})$, and $|\lambda(\mathbf{h})| - 1 \leq \alpha$.
2) (Multiple Failures Recovery, MFR) $\Lambda_\mathcal{C}(\mathbf{r}) = 0$, for all $\mathbf{r} \subseteq \mathcal{N}$ such that $1 \leq |\mathbf{r}| \leq \beta$.

By the SFR criterion, contents lost in any failed data centre can be regenerated from at most α surviving data centres. However, multiple failures may occur in some extreme cases. Therefore, it is also desirable that the whole network can still be recovered in such rare cases. The MFR criterion guarantees that contents lost in any β's failed data centres can be regenerated from the remaining $n-\beta$ data centres.

B. Linear Programming Bounds

One of the most fundamental questions in designing storage code is to characterise the tradeoff between the storage cost and the repair cost. In the following, we will study the tradeoff by formulating the problem as an optimisation problem.

Theorem 1 (Upper bound): Consider any (α,β) storage code $\mathcal{C}$. Then $|\mathcal{C}|$ is upper bounded by the optimal value in the following maximisation problem:

$$\begin{array}{ll}\textbf{maximize} & \sum_{\mathbf{r}\subseteq\mathcal{N}} A_\mathbf{r} \\ \textbf{subject to} & \\ & A_\mathbf{r} \geq 0 \qquad\qquad \forall \mathbf{r}\subseteq\mathcal{N} \\ & B_\mathbf{r} = \dfrac{\sum_{\mathbf{s}\subseteq\mathcal{N}} A_\mathbf{s}\prod_{j=1}^n \kappa_q(s_j,r_j)}{\sum_{\mathbf{s}\subseteq\mathcal{N}} A_\mathbf{s}} \qquad \forall \mathbf{r}\subseteq\mathcal{N} \\ & B_\mathbf{r} \geq 0 \qquad\qquad \forall \mathbf{r}\subseteq\mathcal{N} \\ & A_\mathbf{r} = 0 \qquad\qquad 1 \leq |\mathbf{r}| \leq \beta \\ & A_\emptyset = 1 \\ & \sum_{\mathbf{s}: i\in\mathbf{s} \text{ and } |\mathbf{s}|-1\leq\alpha} B_\mathbf{s} \geq q-1 \qquad \forall i \in \mathcal{N}.\end{array} \quad (2)$$

Proof: Let $\mathcal{C}$ be a (α,β) storage code and

$$A_\mathbf{r} \triangleq \Lambda_\mathcal{C}(\mathbf{r}), \quad \mathbf{r}\subseteq\mathcal{N} \quad (3)$$

$$B_\mathbf{r} \triangleq \Lambda_{\mathcal{C}^\perp}(\mathbf{r}), \quad \mathbf{r}\subseteq\mathcal{N}. \quad (4)$$

Then clearly, $A_\mathbf{r}, B_\mathbf{r} \geq 0$ for all $\mathbf{r}\subseteq\mathcal{N}$, $A_\emptyset = 1$ and

$$|\mathcal{C}| = \sum_{\mathbf{r}\subseteq\mathcal{N}} A_\mathbf{r}.$$

By Proposition 1, we have

$$B_\mathbf{r} = \frac{\sum_{\mathbf{s}\subseteq\mathcal{N}} A_\mathbf{s}\prod_{j=1}^n \kappa_q(s_j,r_j)}{\sum_{\mathbf{s}\subseteq\mathcal{N}} A_\mathbf{s}}, \quad \forall \mathbf{r}\subseteq\mathcal{N}.$$

By MFR criterion,

$$A_\mathbf{r} = 0, \forall 1 \leq |\mathbf{r}| \leq \beta.$$

And due to the SFR criterion, for any $i \in \mathcal{N}$, there exists $\mathbf{h} \in \mathcal{C}^\perp$ such that $i \in \lambda(\mathbf{h})$ and $|\lambda(\mathbf{h})| - 1 \leq \alpha$. Since $\mathcal{C}$ is a linear code, $a\mathbf{h} \in \mathcal{C}^\perp$ for all nonzero element $a \in \mathbb{F}_q$. Let $\mathbf{r} = \lambda(\mathbf{h})$. Then $B_\mathbf{r} \geq q-1$. Consequently,

$$\sum_{\mathbf{s}: i\in\mathbf{s} \text{ and } |\mathbf{s}|-1\leq\alpha} B_\mathbf{s} \geq B_\mathbf{r} \geq q-1.$$

Therefore, $(A_\mathbf{r}, B_\mathbf{r} : \mathbf{r}\subseteq\mathcal{N})$ satisfies the constraint in the maximisation problem, and the theorem then follows. ∎

The maximisation problem in (2) is not a linear programming problem. However, by changing variables, it can be

rewritten as the following linear programming problem:

$$\begin{aligned}
&\textbf{maximize} \quad \textstyle\sum_{\mathbf{r}\subseteq\mathcal{N}} A_{\mathbf{r}} \\
&\textbf{subject to} \\
&\quad A_{\mathbf{r}} \geq 0 && \forall \mathbf{r} \subseteq \mathcal{N} \\
&\quad C_{\mathbf{r}} = \sum_{\mathbf{s}\subseteq\mathcal{N}} A_{\mathbf{s}} \prod_{j=1}^{n} \kappa_q(s_j, r_j) && \forall \mathbf{r} \subseteq \mathcal{N} \\
&\quad C_{\mathbf{r}} \geq 0 && \forall \mathbf{r} \subseteq \mathcal{N} \\
&\quad A_{\mathbf{r}} = 0 && 1 \leq |\mathbf{r}| \leq \beta \\
&\quad A_{\emptyset} = 1 \\
&\quad \sum_{\mathbf{s}: i\in\mathbf{s} \text{ and } |\mathbf{s}|-1\leq\alpha} C_{\mathbf{s}} \geq (q-1) \sum_{\mathbf{r}\subseteq\mathcal{N}} A_{\mathbf{r}} && \forall i \in \mathcal{N}.
\end{aligned} \tag{5}$$

After simplification, the optimisation is further reduced to

$$\begin{aligned}
&\textbf{maximize} \quad \textstyle\sum_{\mathbf{r}\subseteq\mathcal{N}} A_{\mathbf{r}} \\
&\textbf{subject to} \\
&\quad A_{\mathbf{r}} \geq 0 && \forall \mathbf{r} \subseteq \mathcal{N} \\
&\sum_{\mathbf{s}\subseteq\mathcal{N}} A_{\mathbf{s}} \prod_{j=1}^{n} \kappa_q(s_j, r_j) \geq 0 && \forall \mathbf{r} \subseteq \mathcal{N} \\
&\quad A_{\mathbf{r}} = 0 && 1 \leq |\mathbf{r}| \leq \beta \\
&\quad A_{\emptyset} = 1 \\
&\quad \sum_{\mathbf{r}: i\in\mathbf{r} \text{ and } |\mathbf{r}|-1\leq\alpha} \left(\sum_{\mathbf{s}\subseteq\mathcal{N}} A_{\mathbf{s}} \prod_{j=1}^{n} \kappa_q(s_j, r_j) \right) \\
&\qquad\qquad \geq (q-1) \textstyle\sum_{\mathbf{r}\subseteq\mathcal{N}} A_{\mathbf{r}} && \forall i \in \mathcal{N}.
\end{aligned} \tag{6}$$

The complexity of the linear programming problem (6) scales exponentially with n (i.e., the number of data centres in the network). If we ignore the cost for repair (say by setting $\alpha = n-1$), then (α, β) storage code is merely an error correcting code with minimum Hamming distance $\beta + 1$. The outer bound (6) is essentially equivalent to Delsarte's Linear Programming bound [7]. And in this special case, Delsarte's LP bound can be simplified using distance enumerators, instead of support enumerators, leading to a significant complexity reduction. However, unlike in error correction, code constraints for storage codes are more complicated, and cannot be directly specified by using only the distance enumerator of the code. Consequently, it leads to an exponential increase in complexity to compute the bound.

In the following, we will reduce the complexity of the bound by exploiting the symmetries in the problem. To illustrate the idea, consider a (α, β) storage code $\mathcal{C}$ defined by the following columns of generator matrix $G = (\mathbf{g}_i, i = 1, \ldots, n)$. Let $S_{\mathcal{N}}$ be the symmetric group on $\mathcal{N}$. In other words, elements (called *permutations*) in $S_{\mathcal{N}}$ are bijective functions from $\mathcal{N}$ to itself. Clearly, $|S_{\mathcal{N}}| = n!$. Let $\sigma \in S_{\mathcal{N}}$. Together with the code $\mathcal{C}$, each permutation σ defines a new code $\mathcal{C}^{\sigma}$ specified by the following generator matrix columns $(\mathbf{f}_i, i = 1, \ldots, n)$ such that for all $i \in \mathcal{N}$, $\mathbf{f}_i = \mathbf{g}_{\sigma(i)}$. It can be verified easily that $\mathcal{C}^{\sigma}$ is still a (α, β) storage code. Such a symmetry is also reflected in the linear programming bound (6) itself.

Proposition 2 (Symmetry): Suppose $(a_{\mathbf{r}} : \mathbf{r} \subseteq \mathcal{N})$ is feasible in the optimisation problem (6). For any permutation $\sigma \in S_{\mathcal{N}}$, let $(a^{\sigma}_{\mathbf{r}} : \mathbf{r} \subseteq \mathcal{N})$ be defined as follows

$$a^{\sigma}_{\mathbf{r}} = a_{\sigma(\mathbf{r})}$$

where $\sigma(\mathbf{r}) \triangleq \{\sigma(i) : i \in \mathbf{r}\}$. Then $(a^{\sigma}_{\mathbf{r}} : \mathbf{r} \subseteq \mathcal{N})$ is also feasible in (6).

As a corollary, let

$$a^{*}_{\mathbf{r}} = \frac{1}{|S_{\mathcal{N}}|} \sum_{\sigma\in S_{\mathcal{N}}} a^{\sigma}_{\mathbf{r}}.$$

Then $(a^{*}_{\mathbf{r}} : \mathbf{r} \subseteq \mathcal{N})$ is feasible in (6) and

$$\sum_{\mathbf{r}\subseteq\mathcal{N}} a^{*}_{\mathbf{r}} = \sum_{\mathbf{r}\subseteq\mathcal{N}} a_{\mathbf{r}}.$$

Proof: Direct verification. ∎

Despite its simplicity, Proposition 2 is extremely instrumental in reducing complexity to solving the optimisation problem (6). In particular, the proposition implies that it is sufficient to consider only feasible solutions $(a^{*}_{\mathbf{r}} : \mathbf{r} \subseteq \mathcal{N})$.

Proposition 3 (Properties of $(a^{*}_{\mathbf{r}} : \mathbf{r} \subseteq \mathcal{N})$): If $|\mathbf{r}| = |\mathbf{s}|$, then $a^{*}_{\mathbf{r}} = a^{*}_{\mathbf{s}}$.

Due to Proposition 3, we can additionally impose the following constraint in the optimisation problem (6) that

$$A_{\mathbf{r}} = A_{\mathbf{s}}, \quad \forall |\mathbf{r}| = |\mathbf{s}|. \tag{7}$$

without affecting the maximum. Yet, these extra constraints can significantly reduce the number of variables in the optimisation problem to $(n + 1)$ variables. In particular, we have the following theorem, whose proof is omitted.

Theorem 2 (Upper bound): Consider a (α, β) storage code $\mathcal{C}$. Then $|\mathcal{C}|$ is upper bounded by the optimal value in the following maximisation problem

$$\begin{aligned}
&\textbf{maximize} \quad \textstyle\sum_{t=0}^{n} \binom{n}{t} a_t \\
&\textbf{subject to} \\
&a_t \geq 0, \quad \forall t = 0, \ldots, n \\
&b_t = \textstyle\sum_{i=0}^{t} \sum_{j=0}^{n-t} \binom{t}{i} \binom{n-t}{j} a_{i+j} (-1)^i (q-1)^{t-i} \\
&\qquad\qquad \forall t = 0, \ldots, n \\
&b_t \geq 0, \quad \forall t = 0, \ldots, n \\
&\textstyle\sum_{t=1}^{\beta} a_t = 0 \\
&a_0 = 1 \\
&\displaystyle\sum_{t=1}^{1+\alpha} \binom{n-1}{t-1} b_t \geq (q-1) \sum_{t=0}^{n} \binom{n}{t} a_t.
\end{aligned} \tag{8}$$

III. Cost for Update

In the previous sections, a storage code is used to store a piece of data across n data centres such that contents lost in any failed centre can be regenerated from at most α surviving data centres. Naturally, if one needs to update the data, it needs to access all the n data centres and do the updates accordingly. In this section, we consider a more general scenario where multiple pieces of data will be stored in the data centres. As before, we require that contents stored in the n distributed data centres are robust against data centres failures. In addition, we are also interested in storage codes which can be updated efficiently. Since there are more than one piece of data, we

will need to modify our definition of a storage code.

A k-symbol storage code is a linear code specified by $n+k$ generator matrix columns $(\mathbf{g}_1, \ldots, \mathbf{g}_{k+n})$ (each of length k) which satisfy the following criteria:

1) (Permutations) The matrix formed by the columns $(\mathbf{g}_{n+1}, \ldots, \mathbf{g}_{n+k})$ is an $k \times k$ permutation matrix (an invertible matrix where all but one entries in each row or column are zero). For our purpose, we can usually assume without loss of generality that the matrix is the identity matrix.
2) (Full Rank) The matrix formed by $(\mathbf{g}_1, \ldots, \mathbf{g}_n)$ is full rank (i.e., of rank equal to k).

The first n columns correspond to the "coded symbols" stored in the n data centres, while the last k columns correspond to the k source symbols. Let $(Y_1, \ldots, Y_k)$ be the k source symbols. For any $i \in \mathcal{N}$, X_i is the coded symbol stored at data centre i and is defined as

$$X_i = \sum_{j=1}^{k} Y_j g_{j,i} \tag{9}$$

where $\mathbf{g}_i = [g_{1,i}, \ldots, g_{k,i}]^\top$. In this sense, the identity criterion ensures that (9) characterises the relation between the source symbols and the coded symbols, while the full rank criterion guarantees that all sources symbols can be recovered from the coded symbols stored in the n data centres.[1]

Definition 3: A k-symbols storage code $\mathcal{C}$ defined by generator matrix columns $(\mathbf{g}_1, \ldots, \mathbf{g}_{k+n})$ is an (α, β, γ) storage code if it satisfies the following criteria:

1) (Single Failure Recovery, SFR) for any $i \in \{1, \ldots, n\}$, there exists $\mathbf{h} \in \mathcal{C}^\perp$ such that $i \in \lambda(\mathbf{h})$, $\lambda(\mathbf{h}) \subseteq \mathcal{N}$ and $|\lambda(\mathbf{h})| - 1 \leq \alpha$.
2) (Multiple Failures Recovery, MFR) $\Lambda_{\mathcal{C}}(\mathbf{r} \cup \mathbf{s}) = 0$, for all $\mathbf{r} \subseteq \mathcal{N}$ such that $1 \leq |\mathbf{r}| \leq \beta$ and $\mathbf{s} \subseteq \{n+1, \ldots, n+k\}$.
3) (Efficient Update, EU) For any $\ell \in \mathcal{K} \triangleq \{1, \ldots, k\}$,

$$\Lambda_{\mathcal{C}}(\mathbf{r} \cup \{n+\ell\}) = 0, \text{ for all } \mathbf{r} \subseteq \mathcal{N} \text{ and } |\mathbf{r}| > \gamma.$$

The first two criteria is equivalent to saying that the code defined by $(\mathbf{g}_1, \ldots, \mathbf{g}_n)$ is itself an (α, β) storage code. Therefore, if the content in one of the data centres is lost, it can be regenerated from a set of at most α surviving data centres, and that the whole network can still be restored as long as the number of failed data centres is not greater than β. Furthermore, the third criterion ensures that if one of the source packets has been modified, then no more than γ coded packets need to be updated.

Remark: It is worth to notice that these parameters may depend on each other. For example, there are no (α, β, γ) storage code where $\beta \geq \gamma$.

As before, any subset in $\{1, \ldots, n, n+1, \ldots, n+k\}$ will be denoted by a pair of binary vectors $\mathbf{r} = [\mathbf{r}_1, \mathbf{r}_2]$ where $\mathbf{r}_1 = [r_1, \ldots, r_n]$ and $\mathbf{r}_2 = [r_{n+1}, \ldots, r_{n+k}]$. Also, $\mathcal{K}' \triangleq \{n+1, \ldots, n+k\}$.

[1]Notice that only the n coded symbols will be stored in the data centres.

Theorem 3: Consider a linear code $\mathcal{C}$ defined by generator matrix columns $(\mathbf{g}_1, \ldots, \mathbf{g}_{k+n})$. Then $\mathcal{C}$ is an (α, β, γ) storage code if and only if the code satisfies the following criteria:

1) (Permutations) $\sum_{\mathbf{r}_1 \subseteq \mathcal{N}} \Lambda_{\mathcal{C}}(\mathbf{r}_1, \ell) = q-1$, for all $\ell \in \mathcal{K}'$.
2) (Full Rank) For any $\ell \in \mathcal{K}'$,

$$\sum_{\mathbf{r}_1 \subseteq \mathcal{N}} \Lambda_{\mathcal{C}^\perp}(\mathbf{r}_1, \ell) \geq q - 1$$

3) (Single Failure Recovery)

$$\sum_{\mathbf{r}_1 \subseteq \mathcal{N}: \ell \in \mathbf{r}_1 \text{ and } |\mathbf{r}_1| \leq \alpha+1} \Lambda_{\mathcal{C}^\perp}(\mathbf{r}_1, \emptyset) \geq q - 1, \quad \forall \ell \in \mathcal{N}$$

4) (Multiple Failures Recovery) $\Lambda_{\mathcal{C}}(\mathbf{r}_1, \mathbf{r}_2) = 0$, for all $\mathbf{r}_1 \subseteq \mathcal{N}$ such that $1 \leq |\mathbf{r}_1| \leq \beta$.
5) (Efficient Update, EU) For any $\ell \in \mathcal{K}'$,

$$\Lambda_{\mathcal{C}}(\mathbf{r}_1, \ell) = 0, \text{ for all } \mathbf{r}_1 \subseteq \mathcal{N} \text{ and } |\mathbf{r}_1| > \gamma. \tag{10}$$

Theorem 4 (Necessary condition): If an (α, β, γ) linear code $\mathcal{C}$ exists, then there exists nonnegative real numbers $(a_{\mathbf{r}_1, \mathbf{r}_2}, c_{\mathbf{r}_1, \mathbf{r}_2} : \mathbf{r}_1 \subseteq \mathcal{N}, \mathbf{r}_2 \subseteq \mathcal{K}')$ such that

$$c_{\mathbf{r}_1, \mathbf{r}_2} = \sum_{\mathbf{s}_1 \subseteq \mathcal{N}, \mathbf{s}_2 \subseteq \mathcal{K}'} a_{\mathbf{s}_1, \mathbf{s}_2} \prod_{j=1}^{n+k} \kappa_q(s_j, r_j) \quad \forall \mathbf{r}_1 \subseteq \mathcal{N}, \mathbf{r}_2 \subseteq \mathcal{K}' \tag{11}$$

$$\sum_{\mathbf{r}_1 \subseteq \mathcal{N}} a_{\mathbf{r}_1, \ell} = q - 1, \quad \forall \ell \in \mathcal{K}' \tag{12}$$

$$\sum_{\mathbf{r}_1 \subseteq \mathcal{N}} c_{\mathbf{r}_1, \ell} \geq (q-1) \left(\sum_{\mathbf{s}_1 \subseteq \mathcal{N}, \mathbf{s}_2 \subseteq \mathcal{K}'} a_{\mathbf{s}_1, \mathbf{s}_2} \right), \quad \forall \ell \in \mathcal{K}' \tag{13}$$

$$\sum_{\mathbf{r}_1 \subseteq \mathcal{N}: \ell \in \mathbf{r}_1 \text{ and } |\mathbf{r}_1| \leq \alpha+1} c_{\mathbf{r}_1, \emptyset} \geq (q-1) \left(\sum_{\mathbf{s}_1 \subseteq \mathcal{N}, \mathbf{s}_2 \subseteq \mathcal{K}'} a_{\mathbf{s}_1, \mathbf{s}_2} \right), \quad \forall \ell \in \mathcal{N} \tag{14}$$

$$\sum_{\mathbf{r}_2 \subseteq \mathcal{K}'} \sum_{\mathbf{r}_1 \subseteq \mathcal{N}: 1 \leq |\mathbf{r}_1| \leq \beta} a_{\mathbf{r}_1, \mathbf{r}_2} = 0 \tag{15}$$

$$a_{\mathbf{r}_1, \ell} = 0, \quad \forall |\mathbf{r}_1| > \gamma, \ell \in \mathcal{K}' \tag{16}$$

$$a_{\emptyset, \emptyset} = 1. \tag{17}$$

Proof: Let $\mathcal{C}$ be a (α, β, γ) storage code,

$$a_{\mathbf{r}_1, \mathbf{r}_2} = \Lambda_{\mathcal{C}}(\mathbf{r}_1, \mathbf{r}_2)$$

$$c_{\mathbf{r}_1, \mathbf{r}_2} = \left(\sum_{\mathbf{s}_1 \subseteq \mathcal{N}, \mathbf{s}_2 \subseteq \mathcal{K}} a_{\mathbf{s}_1, \mathbf{s}_2} \right) \Lambda_{\mathcal{C}^\perp}(\mathbf{r}_1, \mathbf{r}_2).$$

Then the theorem follows from Theorem 3. ∎

Proposition 4: Suppose $(\mathbf{g}_1, \ldots, \mathbf{g}_{n+k})$ defines a (α, β, γ) code $\mathcal{C}$. For any $\sigma_1 \in S_{\mathcal{N}}$ and $\sigma_2 \in S_{\mathcal{K}'}$, let $\mathcal{C}^{\sigma_1, \sigma_2}$ be another

$$c_{t_1,t_2} = \sum_{u_1=0}^{t_1}\sum_{v_1=0}^{n-t_1}\sum_{u_2=0}^{t_2}\sum_{v_2=0}^{n-t_2}\binom{t_1}{u_1}\binom{n-t_1}{v_1}\binom{t_2}{u_2}\binom{k-t_2}{v_2}(-1)^{u_1+u_2}(q-1)^{t_1+t_2-u_1-u_2},$$

$$\forall 0 \le t_1 \le n,\ 0 \le t_2 \le k \tag{26}$$

$$\sum_{t_1=0}^{n}\binom{n}{t_1}a_{t,1} = q-1, \tag{27}$$

$$\sum_{t_1=0}^{n}\binom{n}{t_1}c_{t_1,0} \ge (q-1)\left(\sum_{t_1=0}^{n}\sum_{t_2=0}^{k}\binom{n}{t_1}\binom{k}{t_2}a_{t_1,t_2}\right), \tag{28}$$

$$\sum_{t_1=1}^{\alpha+1}\binom{n-1}{t_1-1}c_{t_1,0} \ge (q-1)\left(\sum_{t_1=0}^{n}\sum_{t_2=0}^{k}\binom{n}{t_1}\binom{k}{t_2}a_{t_1,t_2}\right), \tag{29}$$

$$\sum_{t_1=1}^{\beta}\sum_{t_2=0}^{k}a_{t_1,t_2} = 0 \tag{30}$$

$$a_{t,1} = 0, \quad \forall n \ge t > \gamma \tag{31}$$

$$a_{0,0} = 1. \tag{32}$$

code specified by $(\mathbf{f}_i, i = 1, \ldots, n+k)$ such that

$$\mathbf{f}_i = \begin{cases} \mathbf{g}_{\sigma_1(i)} & \text{if } i \in \mathcal{N} \\ \mathbf{g}_{\sigma_2(i)} & \text{if } i \in \mathcal{K}^{'}. \end{cases}$$

Then $\mathcal{C}^{\sigma_1,\sigma_2}$ is still a (α, β, γ) code.

Again, we can employ the symmetry in the problem to obtain the following corollary.

Corollary 1: Suppose $(a_{\mathbf{r}_1,\mathbf{r}_2}, c_{\mathbf{r}_1,\mathbf{r}_2} : \mathbf{r}_1 \subseteq \mathcal{N}, \mathbf{r}_2 \subseteq \mathcal{K}^{'})$ satisfies (11)–(17). Let

$$a^*_{\mathbf{r}_1,\mathbf{r}_2} = \frac{1}{|S_{\mathcal{N}}||S_{\mathcal{K}'}|}\sum_{\sigma_1\in S_{\mathcal{N}}}\sum_{\sigma_2\in S_{\mathcal{K}'}} a_{\sigma_1(\mathbf{r}_1),\sigma_2(\mathbf{r}_2)} \tag{18}$$

$$c^*_{\mathbf{r}_1,\mathbf{r}_2} = \frac{1}{|S_{\mathcal{N}}||S_{\mathcal{K}'}|}\sum_{\sigma_1\in S_{\mathcal{N}}}\sum_{\sigma_2\in S_{\mathcal{K}'}} c_{\sigma_1(\mathbf{r}_1),\sigma_2(\mathbf{r}_2)}. \tag{19}$$

Then $(a^*_{\mathbf{r}_1,\mathbf{r}_2}, c^*_{\mathbf{r}_1,\mathbf{r}_2} : \mathbf{r}_1 \subseteq \mathcal{N}, \mathbf{r}_2 \subseteq \mathcal{K}^{'})$ satisfies (11)–(17).

Furthermore, for any $\mathbf{r}_1, \mathbf{s}_1 \subseteq \mathcal{N}$ and $\mathbf{r}_2, \mathbf{s}_2 \subseteq \mathcal{K}^{'}$ such that $|\mathbf{r}_1| = |\mathbf{s}_1|$ and $|\mathbf{r}_2| = |\mathbf{s}_2|$. Then

$$a^*_{\mathbf{r}_1,\mathbf{r}_2} = a^*_{\mathbf{s}_1,\mathbf{s}_2} \tag{20}$$

$$c^*_{\mathbf{r}_1,\mathbf{r}_2} = c^*_{\mathbf{s}_1,\mathbf{s}_2}. \tag{21}$$

Theorem 5 (Necessary condition): If a (α, β, γ) linear code $\mathcal{C}$ exists, then there exists nonnegative real numbers $(a_{t_1,t_2}, c_{t_1,t_2} : t_1 = 0, \ldots, n, t_2 = 0, \ldots, k)$ satisfying (26)-(32).

Note that the numbers of variables and constraints involved in (26)-(32) are respectively $2(n+1)(k+1)$ and $(n+1)(k+1)+6$.

IV. Conclusion

In this paper, we considered a class of linear locally repairable storage codes. One of the fundamental challenges is to determine whether certain storage codes for given costs of storage, repair and update exist or not. To achieve this goal, we extend Delsarte's linear programming for error correcting codes to storage codes. Specifically, we obtain a necessary condition for the existence of a linear storage code.

Acknowledgement

This work was supported in part by the Australian Research Council (DP1094571) and the University Grants Committee of the Hong Kong Special Administrative Region, China (Project No. AoE/E-02/08)

References

[1] A. G. Dimakis, P. B. Godfrey, M. J. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," in *INFOCOM IEEE International Conference on Computer Communications*, 2007, pp. 2000–2008.

[2] D. S. Papailiopoulos and A. G. Dimakis, "Locally repairable codes," in *ISIT*, 2012, pp. 2771–2775.

[3] P. Gopalan, C. Huang, H. Simitci, and S. Yekhanin, "On the locality of codeword symbols," *IEEE Trans. Inf. Theory*, vol. 58, no. 11, pp. 6925–6934, Nov 2012.

[4] C. Huang, M. Chen, and J. Li, "Pyramid codes: Flexible schemes to trade space for access efficiency in reliable data storage systems," in *Proc. 6th IEEE Int. Symp. Netw. Comput. Appl.*, 2007, pp. 79–86.

[5] L. Pamies-Juarez, H. D. L. Hollmann, and F. E. Oggier, "Locally repairable codes with multiple repair alternatives," *CoRR*, vol. abs/1302.5518, 2013.

[6] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error Correcting Codes*. North-Holland, Amsterdam, 1977.

[7] P. Delsarte, "An algebraic approach to the association schemes of coding theory," *Philips Res. Repts. Suppl.*, no. 10, 1973.