doi:10.3934/amc.2020024

LINEAR PROGRAMMING BOUNDS FOR DISTRIBUTED STORAGE CODES

Ali Tebbi

Institute for Telecommunications Research
University of South Australia

Terence Chan* and Chi Wan Sung**

*Institute for Telecommunications Research
University of South Australia
**Department of Electrical Engineering
City University of Hong Kong

(Communicated by Eimear Byrne)

Abstract. A major issue of locally repairable codes is their robustness. If a local repair group is not able to perform the repair process, this will result in increasing the repair cost. Therefore, it is critical for a locally repairable code to have multiple repair groups. In this paper we consider robust locally repairable coding schemes which guarantee that there exist multiple distinct (not necessarily disjoint) alternative local repair groups for any single failure such that the failed node can still be repaired locally even if some of the repair groups are not available. We use linear programming techniques to establish upper bounds on the size of these codes. We also provide two examples of robust locally repairable codes that are optimal regarding our linear programming bound. Furthermore, we address the update efficiency problem of the distributed data storage networks. Any modification on the stored data will result in updating the content of the storage nodes. Therefore, it is essential to minimise the number of nodes which need to be updated by any change in the stored data. We characterise the update-efficient storage code properties and establish the necessary conditions of existence update-efficient locally repairable storage codes.

1. Introduction

Significant increase in the internet applications such as social networks, file, and video sharing applications, results in an enormous user data production which needs to be stored reliably. In a data storage network, data is 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 thousands of storage disks (or nodes). The most common issue with all storage networks is failure. The failure in storage networks is the result of the failed components which are varied from corrupted disk sectors to one or a group of failed disks or even the entire data centre due to the physical damages or natural disasters. The immediate consequence of failure

in a storage network is data loss. Therefore, a repair mechanism must be in place to handle these failures. Protection against failures is usually achieved by adding redundancies [24]. The simplest way to introduce redundancy is *replication*. In this method, multiple copies of each data block are stored in the network. In the case of any failure in the network, the content of the failed node can be recovered by its replica. Because of its simplicity in storage and data retrieval, this method is used widely in conventional storage systems (e.g., RAID-1 where each data block is duplicated and stored in two storage nodes) [6, 14]. However, replication is not an efficient method in term of the storage requirement [12, 35].

Another method for data storage is *erasure coding* which has gained a wide attention in distributed storage [2–4, 23, 30]. In this method as a generalisation of replication, a data block is divided into K fragments and then encoded to N ($N > K$) fragments via a group of mapping functions to store in the network [42]. Erasure coding is capable of providing more reliable storage networks compared to replication with the same storage overhead [41]. It is noteworthy to mention that replication can also be considered as an erasure code (i.e., repetition code [24]).

One of the challenges in storage networks using erasure codes is the repair bandwidth. When a storage node fails, the replaced node (i.e., newcomer) has to download the information content of the survived nodes to repair the lost data. In other words, the repair bandwidth of a distributed storage network is the total amount of the information which is needed to be retrieved to recover the lost data. Compared to the replication, it has been shown in [41] that in the systems with the same *mean time to failure* (MTTF), using erasure codes reduces the storage overhead and the repair bandwidth by an order of magnitude.

From coding theory, it is well known that in a storage network employing an erasure code for data storage, the minimum amount of information which has to be downloaded to repair a single failure is at least the same size of the original data [42]. In other words, the amount of data transmitted in the network to repair a failed node is much more than the storage capacity of the node. *Regenerating codes* which have been introduced in [13] provide coding techniques to reduce the repair bandwidth by increasing the number of nodes which the failed node has to be connected during the repair procedure. It has been shown that there exists a tradeoff between the storage and repair bandwidth in distributed storage networks [13]. Many storage codes in literature assume 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. Although these methods reduce the repair bandwidth, minimising I/O in distributed storage networks is more beneficial than minimising the repair bandwidth [22]. Storage codes such as regenerating codes that access all surviving nodes during repair process, increase I/O overhead. Disk I/O which is proportional to the number of the nodes involved in the repair process, seems to be the repair performance bottleneck of the storage networks [28].

The number of nodes which involve in repairing a failed node is defined as the *code locality* [17]. The concept of code locality relaxes the repairing requirement so that any failed node can be repaired by at least one set (of size smaller than a threshold) of surviving nodes. We refer to these sets as *local repair groups*. Moreover, any erasure code which satisfies this property is referred as *locally repairable code*.

One significant issue of locally repairable codes is their robustness. Consider a distributed storage network with N storage nodes such that there exists only

one local repair group for each node in the network. Therefore, if any node fails, the survived nodes in its repair group can locally repair the failure. However, if any other node in the repair group is also not available (for example, due to the heavy loading), then the repair group cannot be able to repair the failed node anymore which will result in an increase of repair cost. To avoid this problem, it is essential to have multiple local repair groups which can be used for repairing the failure if additional nodes fail or become unavailable in the repair groups. The concept of locally repairable codes with multiple repair groups (or alternatives) was studied in [27] where codes with multiple repair alternatives were also constructed. Locally repairable code construction with multiple disjoint repairing groups was investigated in [34]. Locally repairable codes with multiple erasure tolerance are introduced in [21] where any failed node can be locally repaired even in the presence of multiple erasures. In [40], locally repairable codes with multiple erasure tolerance have been constructed using combinatorial methods. A lower bound on the code size was derived and some code examples provided which are optimal with respect to this bound. A class of locally repairable codes with capability of repairing a set of failed nodes by connecting to a set of survived nodes are studied in [33]. Bounds on code dimension and also families of explicit codes were provided.

The main contributions of this paper are as follows:

- We investigate the linear $(r, \Gamma, \zeta, \beta)$ robust locally repairable codes, defined in Definition 3.4, where any failed node can be repaired by ζ different (i.e., distinct but not necessarily disjoint) repair groups each of size at most r at the presence of any Γ extra failures in the network. These codes have a minimum distance of $d = \beta + 1$, i.e., these codes can tolerate any β simultaneous failures.
- By employing the criteria from the definition of our robust locally repairable codes, we establish a linear programming bound to upper bound the code size.
- We propose two robust locally repairable code examples which are optimal regarding our linear programming bound.
- We investigate the update efficiency of storage codes. In an erasure code, each encoded symbol is a function of the data symbols. Therefore, the update complexity of an erasure code is proportional to the number of encoded symbols that have to be changed in regard of any change in the original data symbols. In other words, in a storage network the content of the storage nodes must be updated by any change in the stored data. We characterise the properties of update-efficient storage codes and establish the necessary conditions that these codes need to satisfy.

Our linear programming bounds are alphabet dependant and establish upper bounds on the code size for arbitrary code parameters such as code length, minimum distance, local repair group's size, number of the local repair groups, and number of the failed nodes in the network. The significance of our linear programming bounds compared to the similar approaches (e.g., [8] or [1]) is that our bounds can be easily generalised for different variations of linear storage codes such that any criterion can be presented as a linear constraint in the problem. Moreover, the complexity of these problems can be significantly reduced by applying symmetries of the codes.

This work is partially presented in the 9th International Conference on Information, Communications & Signal Processing (ICICS 2013) [10] and IEEE Information Theory Workshop (ITW 2014) [39]. This work presents more detailed discussions on characterising robust locally repairable and update-efficient storage codes.

The remaining of the paper is organised as follows. In Section 2, we review the related works on the locally repairable codes. In Section 3, we introduce our robust locally repairable codes and propose their criteria. In Section 4, we establish a linear programming problem to upper bound the code size. We also exploit symmetry on the linear programming bound to reduce its complexity. Section 5 studies the update efficiency of the locally repairable codes. We establish the necessary conditions for distributed storage codes in order to satisfy the locality and update efficiency at the same time. Examples and numerical results are provided in Section 6. The paper then concludes in Section 7.

2. Background and related works

Disk I/O which is proportional to the number of the nodes involved in the repair process, seems to be the repair performance bottleneck of the storage networks [28]. Regenerating codes are an example of this issue where $N-1$ surviving nodes are involved in repair process of a failed node in order to minimise the repair bandwidth [22]. Therefore, the number of nodes which help to repair a failed node is a major metric to measure the repair cost in storage networks. Locally repairable codes aim to optimise this metric (i.e., the number of helper nodes needed to repair a failed node). The locality of a code is defined as the number of code coordinates r such that any code coordinate is a function of them [17]. Consequently, in a storage network employing a locally repairable code, any failed node can be repaired using only r survived nodes. In the case of the linear locally repairable storage codes which are the topic of this paper, the content of any storage node is a linear combination of the contents of a group of r other nodes. Usually two types of code locality are defined in the literature as information symbol locality and all symbol locality which means that just the information symbols in a systematic code have locality r and all symbols have locality r, respectively. In this paper, the code locality is referred to all symbol locality since we do not impose any systematic code constraint on the locally repairable storage codes.

The problem of storage codes with small locality has been initially proposed in [18, 26, 29], independently. The Pyramid codes introduced in [18] are an example of initial works on practical locally repairable storage codes. A basic Pyramid code is constructed using an existing erasure code. An MDS erasure code will be an excellent choice in this case, since they are more tolerable against simultaneous failures. To construct a Pyramid code using an MDS code, it is sufficient to split K data blocks into $\frac{K}{r}$ disjoint groups. Then, a parity symbol is calculated for each group as a local parity. In other words, one of the parity symbols from the original code is split into $\frac{K}{r}$ blocks and the other parity symbols are kept as global parities. Therefore, any single failure in data nodes can be repaired by r survived nodes in the same group instead of retrieving data from K nodes. Generally, for a Pyramid code with data locality r constructed from an (N, K, d) MDS code, the code length (i.e., number of storage nodes) is

$$(1) \qquad N = K\left(1 + \frac{1}{r}\right) + d - 2.$$

Obviously, low locality is gained in the cost of $(\frac{K}{r} - 1)$ extra storage. To obtain all symbol locality for the code, it is sufficient to add a local parity to each group of the global parities of size r where will result in a code length of

(2)
$$N = (K + d - 2)\left(1 + \frac{1}{r}\right).$$

Both of the code lengths in (1) and (2) show a linear increase in storage overhead. Note that in general, adding a local parity to each group will decrease the minimum distance of the code which is a major challenge in constructing optimal all-symbol locally repairable codes [37].

The pioneer work of Gopalan *et al.* in [17] is an in-depth study of codes locality which results in presenting a trade-off between the code length N, data block size K, minimum distance of code d, and code locality r. It has been shown that for any $[N, K, d]_q$ linear code with information locality r,

(3)
$$N - K \geq \left\lceil \frac{K}{r} \right\rceil + d - 2.$$

In other words, to have a linear storage code with locality r, we need a linear overhead of at least $\left\lceil \frac{K}{r} \right\rceil + d - 2$. As a special case of locality $r = K$, the bound in (3) is reduced to

(4)
$$N - K \geq d - 1,$$

which is the Singleton bound and is achievable by MDS codes such as Reed-Solomon codes. According to (1) and (2), Pyramid codes can achieve the bound in (3) which shows their optimality.

Due to their importance in minimising the number of helper nodes to repair a single failure, locally repairable codes have received a fair amount of attention in literature. Different coding schemes to design explicit optimum locally repairable codes has been proposed in [5, 15, 16, 25, 32, 37, 38].

For the bound in (3), the assumption is that any storage node has entropy equal to $\alpha = \frac{\mathcal{M}}{K}$, where $\mathcal{M}$ is the data block size to be stored. It is also mentioned that low locality $r << K$ affects minimum distance d regarding the term $\frac{K}{r}$. It has been shown in [28] that there exist erasure codes which have high distance and low locality if storage nodes store an additional information by a factor of $\frac{1}{r}$. The minimum distance of these codes is upper bounded as

(5)
$$d \leq N - \left\lceil \frac{K}{1 + \epsilon} \right\rceil - \left\lceil \frac{K}{r(1 + \epsilon)} \right\rceil + 2,$$

where $\epsilon > 0$. The obtained LRCs in [28] are vector codes where each data and coded symbol is represented as a vector (not necessarily of the same length). The definition of the vector LRCs, together with the bound in (5), are extended to (r, δ)-*locality* in [32] which we will describe later.

The main issue with locally repairable codes is that they mostly tend to repair a single node failure. This means that if any of the helper nodes in the repair group of a failed node is not available, the local repair process will fail. Therefore, it is critical to have multiple repair groups for each node in the storage network. The concept of locally repairable codes with multiple repair alternatives has been studied initially in [27]. The main idea in [27] is to obtain the parity check matrix of the code from a partial geometries. However, there are only a few such explicit codes due to the limited number of known constructions of partial geometries with desirable parameters. A lower and upper bound is also obtained for the rate of these codes with different locality and repair alternativity. Rawat *et al.* investigated the availability in storage networks employing locally repairable codes [34]. The notion of availability is useful for applications involving hot data since it gives the ability

of accessing to a certain information by different users simultaneously. Therefore, each information node must have t disjoint repair groups of size at most r (i.e., (r, t)-Information local code). An upper bound for the code minimum distance has been obtained in [34] as

$$(6) \qquad d_{\min} \leq N - K + 1 - \left(\left\lceil \frac{Kt}{r} \right\rceil - t \right),$$

which is a generalised version of the bound in (3). However, this bound only stands for information symbols availability with the assumption of only one parity symbol for each repair group. Different bounds on the rate and minimum distance of storage codes with locality r and availability t have been obtained independently in [36] as

$$(7) \qquad \frac{K}{N} \leq \frac{1}{\prod_{j=1}^{t} \left(1 + \frac{1}{jr} \right)}$$

and

$$(8) \qquad d \leq N - \sum_{i=0}^{t} \left\lfloor \frac{K-1}{r^i} \right\rfloor.$$

The minimum distance upper bound in (8) reduces to (3) for $t = 1$. However, the bounds in (6) and (8) seem to be incomparable since the bound in (8) is tighter than (6) in some cases and vice versa.

An upper bound on the code size of locally repairable codes which is dependent on the alphabet size has been obtained in [7] as

$$(9) \qquad K \leq \min_{t \in \mathbb{Z}_+} \left[tr + K_{\text{opt}}^{(q)} \left(n - t(r+1), d \right) \right]$$

where, $K_{\text{opt}}^{(q)}$ is the maximum dimension of a linear or non-linear code of length N and minimum distance d with alphabet size of q. This bound is tighter than the bound in (3).

Locally repairable codes with small availability $t = 2, 3$ are investigated in [20]. The smallest the availability the highest the code rate would be since the dependency among the code symbols is less while the code is still comparable to the practical replication codes. The rate-optimal binary (r, t)-availability codes are designed in [31].

(r, δ)-*locally repairable codes* are introduced in [21] where the failed coded symbols can be repaired locally by at most r symbols at the presence of multiple failures in the network. The symbol c_i, $1 \leq i \leq N$, of a (N, K) code $\mathcal{C}$ with minimum distance d is said to have (r, δ) locality if there exists a subcode of $\mathcal{C}$ with support containing i whose length is at most $r + \delta - 1$ and whose minimum distance is at least δ. For a systematic code with all information symbols (r, δ) locality, any systematic symbol can be repaired by other r symbols even if there are $\delta - 2$ extra failures. The minimum distance of these codes is upper bounded as

$$(10) \qquad d \leq N - K + 1 - \left(\left\lceil \frac{K}{r} \right\rceil - 1 \right) (\delta - 1).$$

Note that (10) is reduced to (3) for $\delta = 2$.

Partial MDS (PMDS) codes are introduced in [5]. A $[mN, m(N-r)-s]$ linear code is a (r, s)-PMDS code where each row of the $m \times N$ array belongs to an $[N, N-r, r+1]$ MDS code. PMDS codes combine local correction of r errors in each row with global correction of s erasures anywhere in the array. As mentioned

earlier, in classes of locally repairable codes (e.g., PMDS codes or Pyramid codes) there are local parity symbols which depend only on a specific group of data symbols and some global parities which depend on all data symbols. While the local parities provide a fast repair of single failures, the global parities provide a high resilience against simultaneous failures. The codes described as above are called *Maximally Recoverable (MR)* [18] if they can repair all failure patterns which are information theoretically repairable. Families of explicit maximally recoverable codes with locality are proposed in [16]. PMDS codes and MR codes with locality are a class of (r, δ)-locally repairable codes that not only have optimal minimum distance, but can correct all information theoretically correctable erasure patterns for the given (r, δ), code length N and dimension K.

A challenge to construct these type of optimal LRCs (e.g., PMDS codes) is to obtain small field sizes. An exponential field size of $O(N^{mN})$ is obtained in [9] for PMDS codes with arbitrary r and s. The authors in [15], extend the work in [16] for $r > 1$ to obtain a relatively small filed size of $O(\max\{m, N^{r+s}\}^s)$. A further reduction on the field size compared to [15] for the case $r \leq s$ is obtained in [25].

Locally repairable codes with multiple localities are introduced in [43] and [19] such that coded symbols have different localities. An upper bound on the minimum distance of these codes with multiple locality of the information symbols is derived as

$$d \leq N - K - \sum_{j=1}^{r} \left\lceil \frac{K_j}{j} \right\rceil + 2,$$

where, K_j is the number of the information symbols with locality j. Locally repairable codes with multiple (r_i, δ_i) which are proposed in [11], generalise the (r, δ) locally repairable codes in [21] to the locally repairable codes with multiple localities in [43] and [19]. These codes ensure different locality for coded symbols at the presence of different number of extra failures. A Singleton-like bound on the minimum distance has been derived and explicit codes have been proposed. A family of maximally recoverable LRCs with multiple localities that admit efficient updates of the parameters (r_i, δ_i), in order to adapt dynamically to different erasure probabilities, network topologies or different hot and cold data, has been proposed in [25].

Notation. Let $\mathcal{N} = \{1, 2, 3, \ldots, N\}$ and $2^{\mathcal{N}}$ be the set of all possible subsets of $\mathcal{N}$. In this paper we represent a subset ν of $\mathcal{N}$ as a binary vector $\mathbf{w} = [w_1, \ldots, w_N]$ such that $w_i = 1$ if and only if $i \in \mathcal{N}$. In our convention, ν and $\mathbf{w}$ can be used interchangeably. Similarly, we use zero vector $\mathbf{0}$ to present an empty set $\varnothing$. Also, we will make no distinction between an index i and $\{i\}$ for a single element set.

3. Linear locally repairable codes

3.1. Linear storage codes. Let $\mathbb{F}_q^N = \{[z_1, \ldots, z_N] : z_i \in \mathbb{F}_q \text{ for all } i \in \mathcal{N}\}$ be the vector space of all N-tuples over the field $\mathbb{F}_q$. A linear storage code $\mathcal{C}$ generated by a $K \times N$ full rank (i.e. rank K) generator matrix G is a subspace of $\mathbb{F}_q^N$ over the field $\mathbb{F}_q$. Here, K is the dimension of the subspace. The linear code $\mathcal{C} = \{\mathbf{u}G : \mathbf{u} \in \mathbb{F}_q^K\}$ is defined as the set of all linear combinations (i.e. spans) of the row vectors of the generator matrix G.

To store a data block of size $\mathcal{M}$ in a storage network using a linear code $\mathcal{C}$, the original data is first divided into K data blocks of size $\mathcal{M}/K$. We may assume without loss of generality that the original data is a matrix with $\mathcal{M}/K$ rows and K columns. Then each block (or row) will be encoded using the same code. In

this paper, we may assume without loss of generality that there is only one row (and thus $\mathcal{M} = K$). Let $\mathbf{u} = [u_1, \ldots, u_K]$ be the block of information to be stored. Then the storage node i will store the symbol $\mathbf{u}\mathbf{g}_i$ where $\mathbf{g}_i$ is the ith column of the generator matrix G. Notice that if X_i is the content that the storage node i stores, then $(X_1, \ldots, X_N)$ is a codeword in $\mathcal{C}$. To retrieve the stored information, it is sufficient to retrieve the information content of K nodes (which are indexed by $\mathcal{I}$) such that the columns $\mathbf{g}_i$ for $\forall i \in \mathcal{I}$ are all independent. Specifically,

$$\mathbf{u} = \mathbf{c}_{\mathcal{I}} G_{\mathcal{I}}^{-1}, \tag{11}$$

where $\mathbf{c}_{\mathcal{I}}$ is the row vector whose i^{th} entry equals to $\mathbf{u}\mathbf{g}_i$ and $G_{\mathcal{I}}$ is a submatrix of the generator matrix G formed by the columns $\mathbf{g}_i$ for $\forall i \in \mathcal{I}$.

Definition 3.1 (Support). For any vector $\mathbf{c} = [c_1, \ldots, c_N] \in \mathbb{F}_q^N$, its support $\lambda(\mathbf{c})$ is defined as a subset of $\mathcal{N} = \{1, \ldots, N\}$ such that $i \in \lambda(\mathbf{c})$ if and only if $c_i \neq 0$.

For a linear storage code $\mathcal{C}$ over $\mathbb{F}_q$, any codeword $\mathbf{c}$ is a vector of N symbols $[c_1, \ldots, c_N]$ where $c_i \in \mathbb{F}_q$ for all $i = 1, \ldots, N$. Since we assume that code $\mathcal{C}$ is linear, for any codeword $\mathbf{c} \in \mathcal{C}$ the vector $a\mathbf{c}$ for all $a \in \mathbb{F}_q$ and $a \neq 0$ is also a codeword in $\mathcal{C}$. Therefore, there exists at least $q - 1$ codewords in $\mathcal{C}$ with the same support (excluding the zero codeword).

Definition 3.2 (Support Enumerator). The support enumerator $\Lambda_{\mathcal{C}}(\mathbf{w})$ of the code $\mathcal{C}$ is the number of codewords in $\mathcal{C}$ with the same support $\mathbf{w}$, i.e.,

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

The support enumerator of a code is very useful to characterise the relation between a code $\mathcal{C}$ and its dual code $\mathcal{C}^{\perp}$ via MacWilliams theorem, hence we aim to bound the size of the locally repairable codes using the dual code properties. Any linear code $\mathcal{C}$ can be specified by a $(N - K) \times N$ parity check matrix H which is orthogonal to the generator matrix G such that

$$GH^{\top} = \mathbf{0},$$

where $\mathbf{0}$ is a $K \times (N - K)$ zero matrix. Then, this parity check matrix H can generate another linear code $\mathcal{C}^{\perp}$ called dual code. It is well known that

$$\mathcal{C}^{\perp} = \left\{ \mathbf{h} \in \mathbb{F}_q^N : \mathbf{h}\mathbf{c}^{\top} = 0 \text{ for all } \mathbf{c} \in \mathcal{C} \right\}.$$

In other words, a linear code $\mathcal{C}$ can be uniquely determined by its dual code $\mathcal{C}^{\perp}$. The number of the codewords in $\mathcal{C}^{\perp}$ for any support $\mathbf{w}$ (i.e., support enumerator $\Lambda_{\mathcal{C}^{\perp}}(\mathbf{w})$) can also be determined uniquely by the support enumerator $\Lambda_{\mathcal{C}}(\mathbf{w})$ of the code $\mathcal{C}$ via the well-known MacWilliams identities.

There are various variations or extensions of MacWilliams identity, all based on association schemes. In the following, we will restate the MacWilliam identity in favour of our purpose in this paper.

Proposition 1 (MacWilliams identity [24, Ch. 5, Theorem 1]). *Let $\mathcal{C}$ be an (N, K) linear code and $\mathcal{C}^{\perp}$ be its dual. Then for any codeword support $\mathbf{w} = [w_1, \ldots, w_N] \subseteq \mathcal{N}$,*

$$\Lambda_{\mathcal{C}^{\perp}}(\mathbf{w}) = \frac{1}{|\mathcal{C}|} \sum_{\mathbf{s} = [s_1, \ldots, s_N] \in 2^{\mathcal{N}}} \Lambda_{\mathcal{C}}(\mathbf{s}) \prod_{j=1}^{N} \kappa_q(s_j, w_j) \geq 0 \tag{12}$$

where

$$(13) \qquad \kappa_q\left(s, w\right) = \begin{cases} 1 & \textit{if } w = 0 \\ q - 1 & \textit{if } s = 0 \textit{ and } w = 1 \\ -1 & \textit{otherwise.} \end{cases}$$

Remark 1. Here, subsets $\mathbf{w}$ and $\mathbf{s}$ of $\mathcal{N}$ are considered as binary vectors with nonzero coordinates corresponding to the indices indexed in them.

In the next subsection we will show the significance of dual code support enumerator in our definition of robust locally repairable codes.

3.2. LOCALITY OF LINEAR STORAGE CODES. A linear storage code has locality r, if for any failed node i, there exists a group of survived nodes of size at most r (i.e., repair group) such that the failed node can be repaired by this group of nodes [17]. To be precise, consider a linear code $\mathcal{C}$ and its dual code $\mathcal{C}^\perp$. Let $\mathbf{h} = [h_1, \ldots, h_N]$ and $\mathbf{c} = [c_1, \ldots, c_N]$ be the codewords in $\mathcal{C}^\perp$ and $\mathcal{C}$, respectively. Then, from the definition of the dual code

$$(14) \qquad \mathbf{h}\mathbf{c}^\top = 0, \quad \forall \mathbf{h} \in \mathcal{C}^\perp, \ \mathbf{c} \in \mathcal{C}.$$

Consequently,

$$(15) \qquad \sum_{j \in \lambda(\mathbf{h})} h_j X_j = 0,$$

where $\lambda(\mathbf{h})$ is the support of the dual codeword $\mathbf{h}$. Now, if the node i fails and $i \in \lambda(\mathbf{h})$, the content of the failed node can be recovered from the set of surviving nodes indexed by $\lambda(\mathbf{h}) \setminus i$. In particular,

$$X_i = -h_i^{-1} \sum_{j \in \lambda(h) \setminus i} X_j h_j.$$

As such, the set $\lambda(\mathbf{h}) \setminus i$ can be seen as a repair group for node i.

Definition 3.3 (Locally repairable code [28]). An (r, β) locally repairable code $\mathcal{C}$ is a linear code which satisfies

1. **Local Recovery (LR).** for any $i \in \mathcal{N}$, there exists $\mathbf{h} \in \mathcal{C}^\perp$ such that $i \in \lambda(\mathbf{h})$ and $|\lambda(\mathbf{h})| - 1 \leq r$.
2. **Global Recovery (GR).** $\Lambda_{\mathcal{C}}(\mathbf{w}) = 0$, for all $\mathbf{w} \subseteq \mathcal{N}$ such that $1 \leq |\mathbf{w}| \leq \beta$.

For a (r, β) locally repairable code, the local recovery criterion implies that any failed node i, can always be repaired by retrieving the contents stored in at most r nodes. On the other hand, the global recovery criterion guarantees that the storage network can be regenerated as long as there are at most β simultaneous node failures. In other words, minimum distance of the code is $d = \beta + 1$.

Remark 2. The LR criterion requires the existence of a set of r surviving nodes which can be used to repair a failed node. However, there is no guarantee that a failed node can still be efficiently repaired[1] if a few additional nodes also fail (or become unavailable). Therefore, it is essential to have alternative repair groups in case of multiple node failures.

---

[1]Strictly speaking, by the GR criterion, the failed node can still be repaired if there are no more than β node failures. However, the global recovery cannot guarantee that each failed node can be repaired efficiently. In other words, one may need a much larger set of surviving nodes to repair one failed node.

Definition 3.4 (Robust locally repairable code)**.** An $(r, \beta, \Gamma, \zeta)$ robust locally repairable code is a linear code satisfying the following criteria:

1. **Robust Local Recovery (RLR).** for any $i \in \mathcal{N}$ and $\gamma \subset \mathcal{N} \setminus i$ such that $|\gamma| = \Gamma$, there exists $\mathbf{h}_1, \ldots, \mathbf{h}_\zeta \in \mathcal{C}^\perp$ such that for all $j = 1, \ldots, \zeta$,
   (a) $i \in \lambda(\mathbf{h}_j)$, $\gamma \cap \lambda(\mathbf{h}_j) = \emptyset$, and $|\lambda(\mathbf{h}_j)| - 1 \le r$.
   (b) $\lambda(\mathbf{h}_j) \ne \lambda(\mathbf{h}_\ell)$ for $\ell \ne j$.
2. **Global Recovery (GR).** $\Lambda_{\mathcal{C}}(\mathbf{w}) = 0$, for all $\mathbf{w} \subseteq \mathcal{N}$ such that $1 \le |\mathbf{w}| \le \beta$.

The RLR criterion guarantees that a failed node can be repaired locally from any one of the ζ groups of surviving nodes of size r even if Γ extra nodes fail. In the special case when $\Gamma = 0$, then the robust locally repairable codes are reduced to locally repairable codes with multiple repair alternatives as in [27]. When $\zeta = 1$, then it further reduces to the traditional locally repairable codes. The GR criterion is the same as that for locally repairable codes.

4. Linear programming bounds

One of the most fundamental problems in storage network design is to optimise the tradeoff between the costs for storage and repair. For a storage network using an (N, K) linear code the capacity of each node is $\frac{\mathcal{M}}{K}$ where $\mathcal{M}$ is the information size. In other words, the storage cost (per information per node) is given by $\frac{\mathcal{M}}{\log_q |\mathcal{C}|}$. Since our coding scheme is linear locally repairable, a group of r nodes participate in repairing a failed node. Thus, the repair cost of a single failure is given by $\frac{r\mathcal{M}}{\log_q |\mathcal{C}|}$. Consequently, by maximising the codebook size $|\mathcal{C}|$ the cost will be minimised.

In this section, we will obtain an upper bound for the maximal codebook size, subject to robust local recovery and global recovery criteria in Definition 3.4.

Theorem 4.1. *Consider any $(r, \beta, \Gamma, \zeta)$ robust locally repairable code $\mathcal{C}$. Then, $|\mathcal{C}|$ is upper bounded by the optimal value of the following optimisation problem.*

(16)
$$
\begin{aligned}
&\textit{maximize} \quad \sum_{\mathbf{w} \subseteq \mathcal{N}} A_{\mathbf{w}} \\
&\textit{subject to} \\
&\quad A_{\mathbf{w}} \ge 0 && \forall \mathbf{w} \subseteq \mathcal{N} \\
&\quad B_{\mathbf{w}} = \frac{\sum_{\mathbf{s} \subseteq \mathcal{N}} A_{\mathbf{s}} \prod_{j=1}^n \kappa_q(s_j, w_j)}{\sum_{\mathbf{s} \subseteq \mathcal{N}} A_{\mathbf{s}}} && \forall \mathbf{w} \subseteq \mathcal{N} \\
&\quad B_{\mathbf{w}} \ge 0 && \forall \mathbf{w} \subseteq \mathcal{N} \\
&\quad A_{\mathbf{w}} = 0 && 1 \le |\mathbf{w}| \le \beta \\
&\quad A_{\varnothing} = 1 \\
&\quad \sum_{\mathbf{s} \in \Omega_i : \gamma \cap \mathbf{s} = \emptyset} B_{\mathbf{s}} \ge \zeta(q-1) && \forall i \in \mathcal{N}, \gamma \in \Delta_i
\end{aligned}
$$

where Ω_i is the collection of all subsets of $\mathcal{N}$ that contains i and of size at most $r+1$ and Δ_i is the collection of all subsets of $\mathcal{N} \setminus i$ of size at most Γ.

Proof. Let $\mathcal{C}$ be a $(r, \beta, \Gamma, \zeta)$ robust locally repairable code. Then, we define

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

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

The objective function of the optimisation problem in (16) is the sum of the number of all codewords with support $\mathbf{w}$, which is clearly equal to the size of the code $\mathcal{C}$. Since $A_{\mathbf{w}}$ and $B_{\mathbf{w}}$ are enumerator functions of the code $\mathcal{C}$ and $\mathcal{C}^\perp$ respectively, $A_{\mathbf{w}}, B_{\mathbf{w}} \ge 0$ for all $\mathbf{w} \subseteq \mathcal{N}$. The constraint $A_{\varnothing} = 1$ follows from the fact that

the zero codeword is contained in any linear code. The second constraint follows from (12) by substituting $|\mathcal{C}|$ and $\Lambda_{\mathcal{C}}(\mathbf{s})$ with $\sum_{\mathbf{s} \subseteq \mathcal{N}} A_{\mathbf{s}}$ and $A_{\mathbf{s}}$, respectively. The fourth constraint follows directly from GR criterion which says that there are no codewords with support of size $\leq \beta$ (i.e., the minimum distance of code is $d = \beta+1$). Finally, the last constraint follows from the RLR criterion. Since the repair groups are specified by dual code $\mathcal{C}^{\perp}$, this constraint bounds the number of codewords in the dual code with support of size at most $r + 1$. The support of the corresponding codewords must contain the index of the failed node i chosen from set Ω_i. However, none of the extra failed nodes indexed in γ can be included in these supports. As mentioned earlier, due to the fact that $\mathcal{C}$ is a linear code, for any nonzero $\mathbf{h} \in \mathcal{C}^{\perp}$, then $a\mathbf{h} \in \mathcal{C}^{\perp}$ for all nonzero $a \in \mathbb{F}_q$. Let $\mathbf{w} = \lambda(\mathbf{h})$. Then, $B_{\mathbf{w}} \geq (q-1)$ (i.e., there exist at least $q - 1$ codewords in dual code with support $\mathbf{w}$). Consequently,

$$\sum_{\mathbf{s} \in \Omega_i : \gamma \cap \mathbf{s} = \emptyset} B_{\mathbf{w}} \geq \zeta(q-1).$$

In other words, this constraint guarantees that there exist at least ζ repair groups to locally repair a failed node at the presence of $|\gamma| = \Gamma$ extra failures. Therefore, $(A_{\mathbf{w}}, B_{\mathbf{w}} : \mathbf{w} \subseteq \mathcal{N})$ satisfies the constraints in the maximisation problem in (16) and the theorem follows. $\square$

The optimisation problem in Theorem 4.1 derives an upper bound on the codebook size $|\mathcal{C}|$ (i.e., Code dimension K) of a robust linear locally repairable code with minimum distance of $d = \beta + 1$ over $\mathbb{F}_q$ such that any failed node can be repaired by ζ repair groups of size r at the presence of any Γ extra failure. Although the maximisation problem in (16) is not a linear programming problem, it can be converted to one easily with a slight manipulation as follows:

(17)
$$
\begin{aligned}
&\textbf{maximize} \quad \sum_{\mathbf{w} \subseteq \mathcal{N}} A_{\mathbf{w}} \\
&\textbf{subject to} \\
&\quad A_{\mathbf{w}} \geq 0 && \forall \mathbf{w} \subseteq \mathcal{N} \\
&\quad \sum_{\mathbf{s} \subseteq \mathcal{N}} A_{\mathbf{s}} \prod_{j=1}^{n} \kappa_q(s_j, w_j) \geq 0 && \forall \mathbf{w} \subseteq \mathcal{N} \\
&\quad A_{\mathbf{w}} = 0 && 1 \leq |\mathbf{w}| \leq \beta \\
&\quad A_{\varnothing} = 1 \\
&\quad \sum_{\mathbf{w} \in \Omega_i : \gamma \cap \mathbf{w} = \emptyset} \left(\sum_{\mathbf{s} \subseteq \mathcal{N}} A_{\mathbf{s}} \prod_{j=1}^{n} \kappa_q(s_j, w_j) \right) \\
&\qquad\qquad \geq \zeta(q-1) \sum_{\mathbf{w} \subseteq \mathcal{N}} A_{\mathbf{w}} && \forall i \in \mathcal{N}, \gamma \in \Delta_i.
\end{aligned}
$$

The complexity of the linear programming problem in (17) will increase exponentially with the number of storage nodes N. In the following, we will reduce the complexity of the Linear Programming problem in (17) by exploiting the symmetries.

To this end, suppose a $(r, \beta, \Gamma, \zeta)$ linear storage code $\mathcal{C}$ is generated by the matrix G with columns $\mathbf{g}_i$, $i = 1, 2, \ldots, N$. Let $S_{\mathcal{N}}$ be the symmetric group on $\mathcal{N}$ whose elements are all the permutations of the elements in $\mathcal{N}$ which are treated as bijective functions from the set of symbols to itself. Clearly, $|S_{\mathcal{N}}| = N!$. Let σ be a permutation on $\mathcal{N}$ (i.e., $\sigma \in S_{\mathcal{N}}$). Together with the code $\mathcal{C}$, each permutation σ defines a new code $\mathcal{C}^{\sigma}$ specified with the generator matrix columns $\mathbf{f}_i$, $i = 1, 2, \ldots, N$, such that for all $i \in \mathcal{N}$, $\mathbf{f}_i = \mathbf{g}_{\sigma(i)}$. Since all the codewords are the linear combinations of the generator matrix rows and the permutation σ just changes the generator matrix columns position, the permutation cannot affect the minimum distance of the

code (i.e., every codeword $\mathbf{c}^\sigma \in \mathcal{C}^\sigma$ is just a permuted version of the corresponding codeword $\mathbf{c} \in \mathcal{C}$). Therefore, the code $\mathcal{C}^\sigma$ is still a $(r, \beta, \Gamma, \zeta)$ linear storage code and we have the following proposition.

Proposition 2. *Suppose $(a_\mathbf{w} : \mathbf{w} \subseteq \mathcal{N})$ satisfies the constraint in the optimisation problem (17). For any $\sigma \in S_\mathcal{N}$, let*

$$(a_\mathbf{w}^\sigma : \mathbf{w} \subseteq \mathcal{N}) = a_\mathbf{w}^\sigma = a_{\sigma(\mathbf{w})},$$

where $\sigma(\mathbf{w}) \triangleq \{\sigma(i) : i \in \mathbf{w}\}$. Then $(a_\mathbf{w}^\sigma : \mathbf{w} \subseteq \mathcal{N})$ also satisfies the constraint in (17).

Corollary 1. *Let*

$$a_\mathbf{w}^* = \frac{1}{|S_\mathcal{N}|} \sum_{\sigma \in S_\mathcal{N}} a_\mathbf{w}^\sigma.$$

Then $(a_\mathbf{w}^* : \mathbf{w} \subseteq \mathcal{N})$ satisfies the constraint in (17) and

$$\sum_{\mathbf{w} \subseteq \mathcal{N}} a_\mathbf{w}^* = \sum_{\mathbf{w} \subseteq \mathcal{N}} a_\mathbf{w}.$$

Proof. From Proposition 2, for any feasible solution $(a_\mathbf{w} : \mathbf{w} \subseteq \mathcal{N})$ in (17), there exists $|S_\mathcal{N}|$ feasible solutions $(a_\mathbf{w}^{\sigma^{(i)}} : \mathbf{w} \subseteq \mathcal{N}$, and $\sigma^{(i)} \in S_\mathcal{N})$. Since (17) is a linear programming problem, $a_\mathbf{w}^*$ (the average of all feasible solutions) is also a feasible solution. The result then follows. □

Proposition 2 can be used to reduce the complexity in solving the optimisation problem (17).

Proposition 3. *If $|\mathbf{w}| = |\mathbf{s}|$, then $a_\mathbf{w}^* = a_\mathbf{s}^*$.*

Proof. Direct verification due to symmetry. □

By Proposition 2, it is sufficient to consider only "symmetric" feasible solution $(a_\mathbf{w}^* : \mathbf{w} \subseteq \mathcal{N})$. Therefore, we can impose additional constraint

(18) $$A_\mathbf{w} = A_\mathbf{s}, \quad \forall |\mathbf{w}| = |\mathbf{s}|.$$

to (17) without affecting the bound. These equality constraint will significantly reduce the number of variables in the optimisation problem. Specifically, we have the following theorem.

Theorem 4.2. *Consider a $(r, \beta, \Gamma, \zeta)$ robust locally repairable code $\mathcal{C}$. Then, $|\mathcal{C}|$ is upper bounded by the optimal value in the following maximisation problem*

$$\textbf{Maximize} \quad \sum_{t=0}^{N} \binom{N}{t} a_t$$

Subject to

(19) $$\begin{cases} a_t \geq 0, & \forall t = 0, \ldots, N \\ b_t = \sum_{i=0}^{t} \sum_{j=0}^{N-t} \binom{t}{i} \binom{N-t}{j} a_{i+j}(-1)^i (q-1)^{t-i} & \forall t = 0, \ldots, N \\ b_t \geq 0, & \forall t = 0, \ldots, N \\ \sum_{t=1}^{\beta} a_t = 0 \\ a_0 = 1 \\ \sum_{t=1}^{r} \binom{N-1-\Gamma}{t} b_{t+1} \geq \zeta(q-1) \sum_{t=0}^{N} \binom{N}{t} a_t. \end{cases}$$

Proof. Due to Proposition 3, we can simplify the LP in (17) by rewriting the variables as follows:

$$a_t = A_{\mathbf{w}} \quad \text{for } t = |\mathbf{w}|$$

$$b_t = B_{\mathbf{w}} \left(\sum_{\mathbf{s} \subseteq \mathcal{N}} A_{\mathbf{s}} \right) \quad \text{for } t = |\mathbf{w}|.$$

Using the new variables, the objective function (i.e., the number of codewords in code $\mathcal{C}$) is reduced to sum of the number of codewords with a support of size t (i.e., a_t) multiplied by the number of subsets with size t (i.e., $\binom{N}{t}$) on all $t = 0, \ldots, N$.

The first constraint follows by the fact that the number of codewords with support of size t is not negative. For any $\mathbf{w} \subseteq \mathcal{N}$ with the size of t, the second constraint corresponds to that the number of codewords in dual code $\mathcal{C}^\perp$ is nonnegative.

To rewrite the second constraint, notice that

$$(20) \qquad \prod_{j=1}^{N} \kappa(s_j, w_j) = (-1)^{|\mathbf{s} \cap \mathbf{w}|} (q-1)^{|\mathbf{w} - \mathbf{s}|}$$

for any pair $(\mathbf{s}, \mathbf{w})$ such that $|\mathbf{s} \cap \mathbf{w}| = i$ and $|\mathbf{w} - \mathbf{s}| = t - i$. In addition, the number of $\mathbf{s}$ such that $|\mathbf{s} \cap \mathbf{w}| = i$ and $|\mathbf{w} - \mathbf{s}| = t - i$ is equal to

$$\binom{t}{i} \binom{N-t}{j}.$$

The third constraint follows by the fact that the number of codewords in dual code $\mathcal{C}^\perp$ with support of size t is not negative. The forth constraint follows from the GR criterion and is equivalent to the third constraint in (17). According to the GR criterion, there exist no codewords with the support size $\leq \beta$ (i.e., $a_t = 0$, $1 \leq t \leq \beta$). Since $(a_t \geq 0, \ \forall t = 0, \ldots, N)$, then $\sum_{t=1}^{\beta} a_t = 0$. The constraint $a_0 = 1$ is equivalent to the fourth constraint in (17) which states that there exists only one codeword in code $\mathcal{C}$ with a support of size $t = 0$ (i.e., zero codeword). The last constraint is equivalent to the last constraint in (17) which follows from the RLR criterion where there are $\binom{N-1-\Gamma}{t}$ subsets (i.e., supports) of size t chosen from $(N-1-\Gamma)$ survived nodes. □

Remark 3. The number of variables and constraint now scales only linearly with N (the number of nodes in the network).

5. COST FOR UPDATE

In previous sections, criteria evaluating performance of storage codes include 1) storage efficiency, 2) local recovery capability and 3) global recovery capability. However, data are rarely static but will be updated from time to time. The fundamental question is how efficient a scheme can be?

5.1. UPDATE MODEL AND CODE DEFINITION. To answer this question, we assume that the data $\mathcal{M}$ consists of multiple files of identical size. In our previous model, we assume that each file will be encoded using the same storage code. Therefore, to update each file, one needs make request to each individual storage node to update the content (regarding that updated file) stored there, leading to excessive overheads.

In this section, we consider a variation of our scheme in the sense that we treat each systematic codeword symbol as a single file (which can be updated individually). The whole collection of files will still be robust and efficient as in the original model. However, one can now consider the problem of update as a single systematic codeword symbol and evaluate how many parity-check codeword symbols are required to be updated.

Definition 5.1 (Update-efficient storage code). A K-symbol update-efficient storage code is specified by a $K \times (N + K)$ generator matrix $\mathbf{G}$ with columns $(\mathbf{g}_1, \ldots, \mathbf{g}_{N+K})$ which satisfy the following criteria:

1. *Pseudo-Systematic* – The submatrix formed by the columns $(\mathbf{g}_{N+1}, \ldots, \mathbf{g}_{N+K})$ is an 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 storage nodes, while the last K columns correspond to the K source symbols. Let $[u_1, \ldots, u_K]$ be the K source symbols. In our context, each source symbol may correspond on a file which can be separately updated. Then, any codeword $\mathbf{c} = [c_1, \ldots, c_N, c_{N+1}, \ldots, c_{N+K}]$ in the update efficient storage code $\mathcal{C}$ is generated by

$$c_i = \sum_{j=1}^{K} u_j g_{j,i}, \qquad \forall i = 1, \ldots, N + K,$$

where $\mathbf{g}_i = [g_{1,i}, \ldots, g_{K,i}]^\top$. However, only the first N coordinates of the codeword $\mathbf{c}$ will be stored in the network. Therefore, for any $i \in \mathcal{N}$ the content of the storage node X_i, is the coded symbol c_i. In particular,

$$(21) \qquad\qquad X_i = \sum_{j=1}^{K} u_j g_{j,i}, \qquad \forall i = 1, \ldots, N.$$

For any vector $\mathbf{v} = [v_1, \ldots, v_{N+K}]$, we will often rewrite it as $\mathbf{v} = [\mathbf{v}_1, \mathbf{v}_2]$ where

$$\mathbf{v}_1 = [v_1, \ldots, v_N] \text{ and } \mathbf{v}_2 = [v_{N+1}, \ldots, v_{N+K}].$$

Definition 5.2 (support). For any vector $\mathbf{v} = [\mathbf{v}_1, \mathbf{v}_2]$ where

$$\mathbf{v}_1 = [v_1, \ldots, v_N] \text{ and } \mathbf{v}_2 = [v_{N+1}, \ldots, v_{N+K}],$$

its support $\lambda(\mathbf{v})$ is defined as two subsets $(\mathbf{w}_1 \subseteq \mathcal{N}, \mathbf{w}_2 \subseteq \mathcal{K})$ where

$$i \in \mathbf{w}_1 \text{ if } v_i \neq 0$$
$$i \in \mathbf{w}_2 \text{ if } v_{i+N} \neq 0,$$

and $\mathcal{K} = \{1, \ldots, K\}$. As before, it is also instrumental to view $\mathbf{w}_1$ and $\mathbf{w}_2$ as binary vectors $\mathbf{w}_1 = [w_{1,1}, \ldots, w_{1,N}]$ and $\mathbf{w}_2 = [w_{2,1}, \ldots, w_{2,K}]$ such that $w_{1,i} = 1$ if $v_i \neq 0$ and $w_{2,i} = 1$ if $v_{i+N} \neq 0$.

Lemma 5.3. *Let G be the generator matrix of an update efficient storage code, as depicted in Figure 1. Then*

1. *For any $i = 1, \ldots, K$, there exists a dual codeword $\mathbf{h} = (\mathbf{h}_1, \mathbf{h}_2)$ such that all entries of $h_{2,j}$ is non-zero if and only if $j \neq i$.*
2. *The rank of the last K column of the parity check matrix must be K.*
3. *If $\mathbf{h} = (\mathbf{h}_1, \mathbf{h}_2)$ is a dual codeword, and $\mathbf{h}_2$ is non-zero, then $\mathbf{h}_1$ must be non-zero as well.*

$$G = \begin{bmatrix} g_{1,1} & \cdots & g_{1,N} & 1 & 0 & \cdots & 0 \\ g_{2,1} & \cdots & g_{2,N} & 0 & 1 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ g_{K,1} & \cdots & g_{K,N} & 0 & 0 & \cdots & 1 \end{bmatrix} \Big\} K$$

over N and K column groups

FIGURE 1. Generator matrix of an update efficient storage code.

$$H = \begin{bmatrix} p_{1,1} & \cdots & p_{1,K} & 1 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ p_{N-K,1} & \cdots & p_{N-K,K} & 0 & \cdots & 1 & 0 & \cdots & 0 \\ p_{N-K+1,1} & \cdots & p_{N-K+1,K} & 0 & \cdots & 0 & 1 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ p_{N,1} & \cdots & p_{N,K} & 0 & \cdots & 0 & 0 & \cdots & 1 \end{bmatrix} \begin{matrix} \Big\} N-K \\ \\ \Big\} K \end{matrix}$$

over N and K column groups

FIGURE 2. Parity check matrix of an update efficient storage code.

4. If $\mathbf{c} = (\mathbf{c}_1, \mathbf{c}_2)$ is a codeword, and $\mathbf{c}_2$ is non-zero, then $\mathbf{c}_1$ must be non-zero as well.

Proof. Rank of G is clearly K and hence rank of H must be N. By construction, the first N columns of G has rank K. This implies 1), which in turns also implies 2). Next, 3) follows from that the last K columns of G has rank K. Finally, 4) follows from that the matrix formed by $(\mathbf{g}_1, \ldots, \mathbf{g}_N)$ is of rank K. $\qquad\square$

Remark 4. Assume without loss of generality that the last N columns of the generator matrix G has rank N. We can assume that the parity check matrix H of the update-efficient storage code can be taken the form in Figure 2 such that for any $1 \leq \ell \leq K$, the $N - K + \ell^{th}$ row of H has at least two non-zero entries. In other words, entries $p_{N-K+\ell,1}, \ldots, p_{N-K+\ell,K}$ cannot be all zero.

Proposition 4 (Code Properties). *Let $\mathcal{C}$ be a K-symbol update efficient locally repairable code. Then*

$$(22) \quad \begin{cases} (22.a) & \sum_{\mathbf{w}_2 \subseteq \mathcal{K}: |\mathbf{w}_2| \geq 1} \Lambda_{\mathcal{C}}(\mathbf{0}, \mathbf{w}_2) = 0 \\ (22.b) & \sum_{\mathbf{w}_2 \subseteq \mathcal{K}: |\mathbf{w}_2| \geq 1} \Lambda_{\mathcal{C}^\perp}(\mathbf{0}, \mathbf{w}_2) = 0 \\ (22.c) & \sum_{\mathbf{w}_1 \subseteq \mathcal{N}} \Lambda_{\mathcal{C}^\perp}(\mathbf{w}_1, \mathbf{w}_2) = (q-1)^{|\mathbf{w}_2|} q^{N-K}, \quad \forall \mathbf{w}_2 \subseteq \mathcal{K}, |\mathbf{w}_2| \geq 0 \\ (22.d) & \sum_{\mathbf{w}_1 \subseteq \mathcal{N}} \Lambda_{\mathcal{C}}(\mathbf{w}_1, \mathbf{w}_2) = (q-1)^{|\mathbf{w}_2|}, \quad\quad\quad \forall \mathbf{w}_2 \subseteq \mathcal{K}, |\mathbf{w}_2| \geq 0 \\ (22.e) & \Lambda_{\mathcal{C}}(\mathbf{0}, \mathbf{0}) = \Lambda_{\mathcal{C}^\perp}(\mathbf{0}, \mathbf{0}) = 1 \end{cases}$$

Proof. Direct verification from the generator matrix and the parity check matrix in Figures 1 and 2, respectively. $\qquad\square$

Definition 5.4 (Update efficient locally repairable storage code). A $K - symbol$ update efficient locally repairable code $\mathcal{C}$ is a (r, β, δ) storage code if satisfies the following criteria:

1. *Local Recovery (LR):* for any $i \in \{1, \ldots, N\}$, there exists $\mathbf{h} = [\mathbf{h}_1, \mathbf{h}_2] \in \mathcal{C}^{\perp}$ where $\mathbf{h}_1 = [h_1, \ldots, h_N]$ and $\mathbf{h}_2 = [h_{N+1}, \ldots, h_{N+K}]$ such that $i \in \lambda(\mathbf{h}_1)$, $\lambda(\mathbf{h}_2) = \emptyset$, and $|\lambda(\mathbf{h}_1)| - 1 \leq r$.
2. *Global Recovery (GR):* $\Lambda_{\mathcal{C}}(\mathbf{w}_1, \mathbf{w}_2) = 0$, for all $\mathbf{w}_1 \subseteq \mathcal{N}$ such that $1 \leq |\mathbf{w}_1| \leq \beta$ and $\mathbf{w}_2 \subseteq \mathcal{K}$.
3. *Efficient Update (EU):* For any $\mathbf{w}_2 \subseteq \mathcal{K}$ and $|\mathbf{w}_2| = 1$,

$$\Lambda_{\mathcal{C}}(\mathbf{w}_1, \mathbf{w}_2) = 0, \qquad \forall \mathbf{w}_1 \subseteq \mathcal{N} \text{ and } |\mathbf{w}_1| > \delta.$$

Remark 5. It is worth to notice that these parameters may depend on each other. For example, it can be directly verified that there is no (r, β, δ) storage code where $\beta \geq \delta$.

The LR criterion guarantees that for any failed node X_i there exists a local repair group specified by the support of the first N symbols of the corresponding dual codeword. In other words, while

$$\mathbf{h}\mathbf{c}^{\top} = \sum_{i=1}^{N+K} h_i c_i = 0,$$

only $(h_1, \ldots, h_N)$ and $(c_1, \ldots, c_N)$ are participating in the repair process such that

$$\sum_{i=1}^{N} h_i X_i = 0.$$

Therefore, the support of $\mathbf{h}_1 = [h_1, \ldots, h_N]$ specifies the repair groups while $\mathbf{h}_2$ must be a zero vector. The GR criterion guarantees that the stored data is recoverable at the presence of severe failure pattern. More specifically, in the worst case of β simultaneous node failures, data stored in the network can still be repaired from the survived nodes, i.e., minimum distance of the code is $d = \beta + 1$. Finally, the EU criterion ensures that if any of the source symbols is modified, then no more than δ storage nodes need to be updated. In particular, for any singleton $\mathbf{w}_2$, there exists a unique $\mathbf{w}_1$ such that

$$\Lambda_{\mathcal{C}}(\mathbf{w}_1, \mathbf{w}_2) = \sum_{\mathbf{w}_1'} \Lambda_{\mathcal{C}}(\mathbf{w}_1', \mathbf{w}_2) = q - 1.$$

The $\mathbf{w}_1$ corresponds to codewords where only the single data file (indexed by the support of $\mathbf{w}_2$) is non-zero. In that case, if the data file has been changed, only the codeword symbols corresponding to the support of $\mathbf{w}_1$ need to be updated. In order to maintain the update cost efficiency (i.e., equal or less than δ nodes per any source symbol change), the code $\mathcal{C}$ must not include any codeword with support $|\mathbf{w}_1| > \delta$ for any $|\mathbf{w}_2| = 1$.

Proposition 5. *For a (r, β, δ) update efficient locally repairable code $\mathcal{C}$, we have*

(23)
$$\begin{cases} (23.a) & \sum_{\mathbf{w}_1 \subseteq \mathcal{N}: \ell \in \mathbf{w}_1 \text{ and } |\mathbf{w}_1| \leq r+1} \Lambda_{\mathcal{C}^{\perp}}(\mathbf{w}_1, \emptyset) \geq (q-1), & \forall \ell \in \mathcal{N} \\ (23.b) & \Lambda_{\mathcal{C}}(\mathbf{w}_1, \mathbf{w}_2) = 0, & \forall \mathbf{w}_1 \subseteq \mathcal{N}, \mathbf{w}_2 \subseteq \mathcal{K}, 1 \leq |\mathbf{w}_1| \leq \beta \\ (23.c) & \Lambda_{\mathcal{C}}(\mathbf{w}_1, \mathbf{w}_2) = 0, & \forall \mathbf{w}_1 \subseteq \mathcal{N} \text{ and } |\mathbf{w}_1| > \delta \text{ and } |\mathbf{w}_2| = 1 \end{cases}$$

Remark 6. For notation convenience, we will first consider only the case of local recovery. However, the extension to include robust local recovery is straightforward. We will present the results afterwards.

The following theorem gives the necessary conditions for existence of an update efficient locally repairable code based on its generator matrix specifications in Definition 5.4.

Theorem 5.5 (Necessary condition). *Let $\mathcal{C}$ be a (N, K, r, β, δ) update-efficient locally repairable storage code. Then there exists $(A_{\mathbf{w}_1,\mathbf{w}_2}, C_{\mathbf{w}_1,\mathbf{w}_2} : \mathbf{w}_1 \subseteq \mathcal{N}, \mathbf{w}_2 \subseteq \mathcal{K})$ such that*

(24)
$$
\begin{cases}
(C1) & C_{\mathbf{w}_1,\mathbf{w}_2} = \sum_{\mathbf{s}_1 \subseteq \mathcal{N}, \mathbf{s}_2 \subseteq \mathcal{K}} A_{\mathbf{s}_1,\mathbf{s}_2} \prod_{j=1}^{N+K} \kappa_q(s_j, w_j), & \forall \mathbf{w}_1 \subseteq \mathcal{N}, \mathbf{w}_2 \subseteq \mathcal{K} \\
(C2) & A_{\mathbf{w}_1,\mathbf{w}_2} \geq 0, & \forall \mathbf{w}_1 \subseteq \mathcal{N}, \mathbf{w}_2 \subseteq \mathcal{K} \\
(C3) & C_{\mathbf{w}_1,\mathbf{w}_2} \geq 0, & \forall \mathbf{w}_1 \subseteq \mathcal{N}, \mathbf{w}_2 \subseteq \mathcal{K} \\
(C4) & A_{\emptyset,\mathbf{w}_2} = 0 & \forall \mathbf{w}_2 \subseteq \mathcal{K}, |\mathbf{w}_2| \geq 1 \\
(C5) & C_{\emptyset,\mathbf{w}_2} = 0 & \forall \mathbf{w}_2 \subseteq \mathcal{K}, |\mathbf{w}_2| \geq 1 \\
(C6) & \sum_{\mathbf{w}_1 \subseteq \mathcal{N}} C_{\mathbf{w}_1,\mathbf{w}_2} = (q-1)^{|\mathbf{w}_2|} q^N, & \forall \mathbf{w}_2 \subseteq \mathcal{K}, |\mathbf{w}_2| \geq 0 \\
(C7) & \sum_{\mathbf{w}_1 \subseteq \mathcal{N}} A_{\mathbf{w}_1,\mathbf{w}_2} = (q-1)^{|\mathbf{w}_2|}, & \forall \mathbf{w}_2 \subseteq \mathcal{K}, |\mathbf{w}_2| \geq 0 \\
(C8) & A_{\emptyset,\emptyset} = 1, & \\
(C9) & C_{\emptyset,\emptyset} = q^K, & \\
(C10) & \sum_{\mathbf{w}_1 \subseteq \mathcal{N}: \ell \in \mathbf{w}_1 \ and \ |\mathbf{w}_1| \leq r+1} C_{\mathbf{w}_1,\emptyset} \geq (q-1)q^K, & \forall \ell \in \mathcal{N} \\
(C11) & A_{\mathbf{w}_1,\mathbf{w}_2} = 0, & \forall 1 \leq |\mathbf{w}_1| \leq \beta \\
(C12) & A_{\mathbf{w}_1,\mathbf{w}_2} = 0, & \forall |\mathbf{w}_1| > \delta, |\mathbf{w}_2| = 1
\end{cases}
$$

Proof. Let $\mathcal{C}$ be a (r, β, δ) update efficient linear storage code. We define

(25)
$$A_{\mathbf{w}_1,\mathbf{w}_2} \triangleq \Lambda_{\mathcal{C}}(\mathbf{w}_1, \mathbf{w}_2)$$

(26)
$$C_{\mathbf{w}_1,\mathbf{w}_2} \triangleq q^K \Lambda_{\mathcal{C}^\perp}(\mathbf{w}_1, \mathbf{w}_2).$$

Constraint (C1) follows directly from the MacWilliams identity in Proposition 1. Constraint (C2) and (C3) follow from (25) and (26) and that enumerating functions are non-negative. Constraints (C4) - (C9) are consequences of Proposition 4 and Constraints (C10)-(C12) are due to Proposition 5. $\square$

As before, the constraints in (24) are overly complex. Invoking symmetry, we will to simplify the above set of constraints.

Proposition 6. *Suppose the generator matrix specified with the column vectors $(\mathbf{g}_1, \ldots, \mathbf{g}_{N+K})$ defines a (r, β, δ) code $\mathcal{C}$. For any $\sigma_1 \in S_{\mathcal{N}}$ (i.e., symmetric group of $\mathcal{N}$) and $\sigma_2 \in S_{\mathcal{K}}$ (i.e., symmetric group of $\mathcal{K}$), let $\mathcal{C}^{\sigma_1,\sigma_2}$ be another code specified by $(\mathbf{f}_i, i = 1, \ldots, N + K)$ such that*

$$\mathbf{f}_i = \mathbf{g}_{\sigma_1(i)} \quad for \ i \in \mathcal{N}$$
$$\mathbf{f}_{N+j} = \mathbf{g}_{N+\sigma_2(j)} \quad for \ j \in \mathcal{K}.$$

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

Proof. Let $\mathbf{c} = [\mathbf{c}_1, \mathbf{c}_2]$ be a codeword in $\mathcal{C}$ where $\mathbf{c}_1 = [c_1, \ldots, c_N]$ and $\mathbf{c}_2 = [c_{N+1}, \ldots, c_{N+K}]$. Then, the vector $\mathbf{c}^{\sigma_1,\sigma_2} = [\mathbf{c}_1^{\sigma_1}, \mathbf{c}_2^{\sigma_2}]$ is a codeword in $\mathcal{C}^{\sigma_1,\sigma_2}$ where $\mathbf{c}_1^{\sigma_1} = [\sigma_1(c_1), \ldots, \sigma_1(c_N)]$ and $\mathbf{c}_2^{\sigma_2} = [c_{N+\sigma_2(1)}, \ldots, c_{N+\sigma_2(K)}]$. In other words, any codeword $\mathbf{c}^{\sigma_1,\sigma_2} \in \mathcal{C}^{\sigma_1,\sigma_2}$ is a permutation of a codeword $\mathbf{c} \in \mathcal{C}$ for all $\sigma_1 \in S_{\mathcal{N}}$ and $\sigma_2 \in S_{\mathcal{K}}$. Equivalently, the new code $\mathcal{C}^{\sigma_1,\sigma_2}$ is obtained from $\mathcal{C}$ by relabelling or reordering the codeword indices. The proposition thus follows. $\square$

Corollary 2. *Suppose* $(a_{\mathbf{w}_1,\mathbf{w}_2}, c_{\mathbf{w}_1,\mathbf{w}_2} : \mathbf{w}_1 \subseteq \mathcal{N}, \mathbf{w}_2 \subseteq \mathcal{K})$ *satisfies* (24)*. Let*

$$(27) \qquad a^*_{\mathbf{w}_1,\mathbf{w}_2} = \frac{1}{|S_{\mathcal{N}}||S_{\mathcal{K}}|} \sum_{\sigma_1 \in S_{\mathcal{N}}} \sum_{\sigma_2 \in S_{\mathcal{K}}} a_{\sigma_1(\mathbf{w}_1),\sigma_2(\mathbf{w}_2)}$$

$$(28) \qquad c^*_{\mathbf{w}_1,\mathbf{w}_2} = \frac{1}{|S_{\mathcal{N}}||S_{\mathcal{K}}|} \sum_{\sigma_1 \in S_{\mathcal{N}}} \sum_{\sigma_2 \in S_{\mathcal{K}}} c_{\sigma_1(\mathbf{w}_1),\sigma_2(\mathbf{w}_2)}.$$

Then $(a^*_{\mathbf{w}_1,\mathbf{w}_2}, c^*_{\mathbf{w}_1,\mathbf{w}_2} : \mathbf{w}_1 \subseteq \mathcal{N}, \mathbf{w}_2 \subseteq \mathcal{K})$ *also satisfies* (24)*. Furthermore, for any* $\mathbf{w}_1, \mathbf{s}_1 \subseteq \mathcal{N}$ *and* $\mathbf{w}_2, \mathbf{s}_2 \subseteq \mathcal{K}$ *such that* $|\mathbf{w}_1| = |\mathbf{s}_1|$ *and* $|\mathbf{w}_2| = |\mathbf{s}_2|$*. Then*

$$(29) \qquad a^*_{\mathbf{w}_1,\mathbf{w}_2} = a^*_{\mathbf{s}_1,\mathbf{s}_2}$$

$$(30) \qquad c^*_{\mathbf{w}_1,\mathbf{w}_2} = c^*_{\mathbf{s}_1,\mathbf{s}_2}.$$

Proof. From Proposition 6, if $(a_{\mathbf{w}_1,\mathbf{w}_2}, c_{\mathbf{w}_1,\mathbf{w}_2} : \mathbf{w}_1 \subseteq \mathcal{N}, \mathbf{w}_2 \subseteq \mathcal{K})$ satisfy constraints (24), then $(a_{\sigma_1(\mathbf{w}_1),\sigma_2(\mathbf{w}_2)}, c_{\sigma_1(\mathbf{w}_1),\sigma_2(\mathbf{w}_2)} : \mathbf{w}_1 \subseteq \mathcal{N}, \mathbf{w}_2 \subseteq \mathcal{K})$ will satisfy them for all $\sigma_1 \in S_{\mathcal{N}}$ and $\sigma_2 \in \mathcal{K}$. Since all the constraints in (24) are convex, any convex combination of all these feasible solutions is still feasible. Hence, $(a^*_{\mathbf{w}_1,\mathbf{w}_2}, c^*_{\mathbf{w}_1,\mathbf{w}_2} : \mathbf{w}_1 \subseteq \mathcal{N}, \mathbf{w}_2 \subseteq \mathcal{K})$ is still feasible. Also,

$$\{(\sigma_1(\mathbf{w}_1), \sigma_2(\mathbf{w}_2)) : \forall \sigma_1 \in S_{\mathcal{N}}, \sigma_2 \in S_{\mathcal{K}}\} = \{(\sigma_1(\mathbf{s}_1), \sigma_2(\mathbf{s}_2)) : \forall \sigma_1 \in S_{\mathcal{N}}, \sigma_2 \in S_{\mathcal{K}}\}$$

if $|\mathbf{w}_1| = |\mathbf{s}_1|$ and $|\mathbf{w}_2| = |\mathbf{s}_2|$ for any $\mathbf{w}_1, \mathbf{s}_1 \subseteq \mathcal{N}$ and $\mathbf{w}_2, \mathbf{s}_2 \subseteq \mathcal{K}$. $\qquad\square$

Theorem 5.6 (Necessary condition)**.** *Let* $\mathcal{C}$ *be a* (N, K, r, β, δ) *update-efficient storage code. Then there exists real numbers* $(a_{t_1,t_2}, c_{t_1,t_2} : t_1 = 0, \ldots, N, t_2 = 0, \ldots, K)$ *satisfying* (31)*.*

(31)

$$
\begin{cases}
(D1) & c_{t_1,t_2} = \sum_{u_1=0}^{t_1} \sum_{v_1=0}^{N-t_1} \sum_{u_2=0}^{t_2} \sum_{v_2=0}^{K-t_2} \Xi(u_1, v_1, u_2, v_2) a_{u_1+v_1,u_2+v_2} \\
(D2) & a_{t_1,t_2} \geq 0, & \forall 0 \leq t_1 \leq N, 0 \leq t_2 \leq K \\
(D3) & c_{t_1,t_2} \geq 0, & \forall 0 \leq t_1 \leq N, 0 \leq t_2 \leq K \\
(D4) & a_{0,t_2} = 0, & \forall t_2 \geq 1 \\
(D5) & c_{0,t_2} = 0, & \forall t_2 \geq 1 \\
(D6) & \sum_{t_1=0}^{N} \binom{N}{t_1} c_{t_1,t_2} = (q-1)^{t_2} q^N, & \forall 0 \leq t_2 \leq K \\
(D7) & \sum_{t_1=0}^{N} \binom{N}{t_1} a_{t_1,t_2} = (q-1)^{t_2}, & \forall 0 \leq t_2 \leq K \\
(D8) & a_{0,0} = 1, \\
(D9) & c_{0,0} = q^K \\
(D10) & \sum_{t_1=2}^{r+1} \binom{N-1}{t_1-1} c_{t_1,0} \geq (q-1)q^K \\
(D11) & a_{t_1,t_2} = 0, & \forall 1 \leq t_1 \leq \beta \\
(D12) & a_{t_1,1} = 0, & \forall t_1 > \delta
\end{cases}
$$

where

$$\Xi(u_1, v_1, u_2, v_2) = (-1)^{u_1+u_2}(q-1)^{t_1+t_2-u_1-u_2} \binom{t_1}{u_1} \binom{N-t_1}{v_1} \binom{t_2}{u_2} \binom{k-t_2}{v_2}.$$

Proof. The theorem is essentially a direct consequence of Theorem 5.5 and Corollary 2, which guarantees that there exists a "symmetric" feasible solution $(a^*_{\mathbf{w}_1,\mathbf{w}_2}, c^*_{\mathbf{w}_1,\mathbf{w}_2} : \mathbf{w}_1 \subseteq \mathcal{N}, \mathbf{w}_2 \subseteq \mathcal{K})$ satisfying (24). Due to symmetry, we can rewrite $a^*_{\mathbf{w}_1,\mathbf{w}_2}$ as a_{t_1,t_2} and $c^*_{\mathbf{w}_1,\mathbf{w}_2}$ as c_{t_1,t_2} where $t_1 = |\mathbf{w}_1|$ and $t_2 = |\mathbf{w}_2|$. The set of constraints in (31) is basically obtained by rewriting (24) and grouping like terms.

Most rephrasing is straightforward. The more complicated one is (D1), where we will show how to rewrite it.

Recall the constraint (C1) in (24)

$$(32) \qquad C_{\mathbf{w}_1,\mathbf{w}_2} = \sum_{\mathbf{s}_1 \subseteq \mathcal{N}, \mathbf{s}_2 \subseteq \mathcal{K}} A_{\mathbf{s}_1,\mathbf{s}_2} \prod_{j=1}^{N+K} \kappa_q(s_j, w_j), \quad \forall \mathbf{w}_1 \subseteq \mathcal{N}, \mathbf{w}_2 \subseteq \mathcal{K}.$$

Let $|\mathbf{w}_1| = t_1$, $|\mathbf{w}_2| = t_2$. For any

$$(33) \qquad 0 \le u_1 \le t_1, \text{ and } 0 \le v_1 \le N - t_1$$

$$(34) \qquad 0 \le u_2 \le t_2, \text{ and } 0 \le v_2 \le K - t_2,$$

the number of pairs of $(\mathbf{s}_1, \mathbf{s}_2)$ such that

$$u_1 = |\mathbf{w}_1 \cap \mathbf{s}_1| \text{ and } v_1 = |\mathbf{s}_1 \setminus \mathbf{w}_1|$$
$$u_2 = |\mathbf{w}_2 \cap \mathbf{s}_2| \text{ and } v_2 = |\mathbf{s}_2 \setminus \mathbf{w}_2|$$

is equal to

$$\binom{t_1}{u_1}\binom{N - t_1}{v_1}\binom{t_2}{u_2}\binom{K - t_2}{v_2}.$$

Also, for each such pair, by (20),

$$\prod_{j=1}^{N+K} \kappa_q(s_j, w_j) = (-1)^{u_1+u_2}(q-1)^{t_1+t_2-u_1-u_2}.$$

Grouping all like terms together, (32) can be written as in (D1) in (31). $\qquad \square$

It is worth mentioning that unlike the linear programming problem in Theorem 4.2, the code dimension K (hence, the code size $|\mathcal{C}| = q^K$) is predetermined in Theorem 5.6. Notice that the complexity of the constraints in Theorem 5.5 is increased exponentially by the number of the storage nodes N and the number of the information symbols K. However, the symmetrisation technique which is applied to the constraints significantly reduces the number of the variables to $2(N+1)(K+1)$.

Remark 7. Theorem 5.6 proves that if there exist no real numbers $(a_{t_1,t_2}, c_{t_1,t_2} : 0 \le t_1 \le N, 0 \le t_2 \le K)$ satisfying constraints (D1)–(D12) in (31), then there exist no (N, K, r, β, δ) update-efficient storage code.

Remark 8. In the previous linear programming bound, we consider only the local recovery criteria. Extension to robust local recovery criteria is similar.

For the robust local recovery criteria, recall from Definition 3.4, it requires that for any failed node $i \in \mathcal{N}$ and additional Γ failed nodes γ, there exists at least ζ dual codewords (with different supports).

In other words, for any $i \in \mathcal{N}$ and $\gamma \subset \mathcal{N} \setminus i$ such that $|\gamma| = \Gamma$,

$$(35) \qquad \sum_{\mathbf{w}_1 \subseteq \mathcal{N}, i \in \mathbf{w}_1, \gamma \cap \mathbf{w}_1 = \emptyset} \Lambda_{\mathcal{C}^\perp}(\mathbf{w}_1, \emptyset) \ge \zeta(q-1).$$

As before, the constraint can be rewritten as follows, after symmetrisation

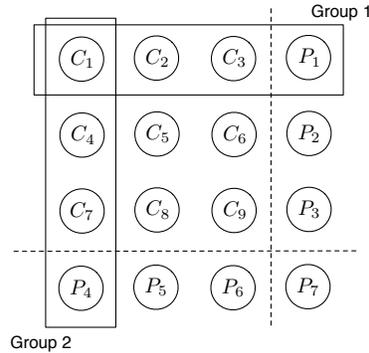$$(36) \qquad \sum_{t_1=2}^{r+1} \binom{N - \Gamma - 1}{t_1 - 1} c_{t_1,0} \ge \zeta(q-1)q^K.$$

FIGURE 3. A (3,3,1,1) robust locally repairable code of length $N = 16$ and dimension $K = 9$.

6. Numerical results and discussion

In this section, we present two examples of robust locally repairable codes. We will show that these codes can repair a failed node locally when some of the survived nodes are not available. Using our bound, we prove that these codes are optimal.

Example 1. Consider a binary linear storage code of length $N = 16$ with $K = 9$ as defined by the following parity check equations

$$(37) \qquad\qquad 0 = P_1 + C_1 + C_2 + C_3$$

$$(38) \qquad\qquad 0 = P_2 + C_4 + C_5 + C_6$$

$$(39) \qquad\qquad 0 = P_3 + C_7 + C_8 + C_9$$

$$(40) \qquad\qquad 0 = P_4 + C_1 + C_4 + C_7$$

$$(41) \qquad\qquad 0 = P_5 + C_2 + C_5 + C_8$$

$$(42) \qquad\qquad 0 = P_6 + C_3 + C_6 + C_9$$

$$0 = P_7 + C_1 + C_2 + C_3$$
$$+ C_4 + C_5 + C_6$$
$$(43) \qquad\qquad + C_7 + C_8 + C_9.$$

Here, $(C_1, \ldots, C_9, P_1, \ldots, P_7)$ correspond to the content stored at the 16 nodes. In particular, we might interpret that $C_1, \ldots, C_9$ are the systematic bits while $P_1, \ldots, P_7$ are the parity check bits. The code can also be represented by Figure 3. In this case, the parity check equations are equivalent to that the sum of rows and the sum of columns are all zero.

According to (37)-(43), every failed node (either systematic or parity) can be repaired by two different repair groups of size $r = 3$. For instance, if C_1 is failed, it can be repaired by repair group $R_{C_1}^1 = \{C_2, C_3, P_1\}$ or repair group $R_{C_1}^2 = \{C_4, C_7, P_4\}$. Notice that the two repair groups are disjoint. Therefore, even if one of the repair groups is not available, there exists an alternative repair group to locally repair the failed node.

It can be verified easily that our code has a minimum distance of 4 and it is a $(3, 3, 1, 1)$ robust locally repairable code. Therefore, any failed node can be repaired by at least $\zeta = 1$ repairing group of size $r = 3$ in the presence of any $\Gamma = 1$ extra

failure. Also, the original data can be recovered even if there are $\beta = 3$ simultaneous failure. In fact, it is also a $(3, 3, 0, 2)$ robust locally repairable code.

Now, we will use the bound obtained earlier to show that our code is optimal. We have plotted our bound in Figure 4 when $n = 16$ and $\beta = 3$. The horizontal axis is the code locality (i.e., r) and the vertical axis is the dimension of the code (or $\log|\mathcal{C}|$). From the figure, when $r = 3$, the dimension of a $(3, 3, 1, 1)$ robust locally repairable code is at most 9. And our constructed code has exactly 9 dimensions. Therefore, our code meets the bound and is optimal. In fact, our bound also indicates that the dimension of a $(3, 3, 0, 2)$ robust locally repairable code is also at most 9. Therefore, our code is in fact an optimal $(3, 3, 1, 1)$ and $(3, 3, 0, 2)$ robust locally repairable code.

Example 2. Consider a binary linear code of length $N = 8$ and dimension $K = 4$ defined by the following parity check equations:

$$(44) \qquad\qquad P_1 = C_1 + C_2 + C_3$$

$$(45) \qquad\qquad P_2 = C_1 + C_2 + C_4$$

$$(46) \qquad\qquad P_3 = C_2 + C_3 + C_4$$

$$(47) \qquad\qquad P_4 = C_1 + C_3 + C_4$$

Again, $C_1, \ldots, C_4$ are the information bits while $P_1, \ldots, P_4$ are the parity check bits. The code can be represented by Figure 5. The above parity check equations imply that the sum of any node and its three adjacent nodes in Figure 5 is always equal to zero.

This code has a minimum distance of 4. According to the Equations (44)-(47), for every single node failure, there exists 7 repair groups with size $r = 3$. For example, suppose C_1 fails. Then, the repair groups are

$$R_1^{C_1} = \{3, 4, 8\}, R_2^{C_1} = \{2, 4, 6\}, R_3^{C_1} = \{2, 3, 5\}$$
$$R_4^{C_1} = \{2, 7, 8\}, R_5^{C_1} = \{3, 6, 7\}, R_6^{C_1} = \{4, 5, 7\}$$
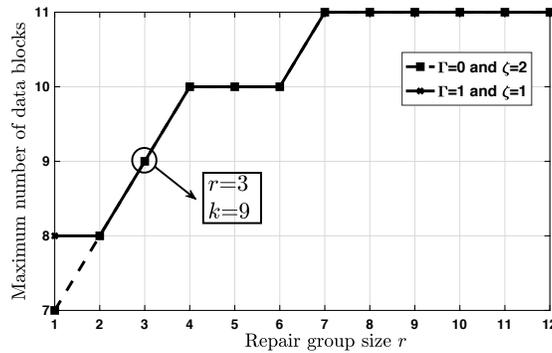$$R_7^{C_1} = \{5, 6, 8\}$$

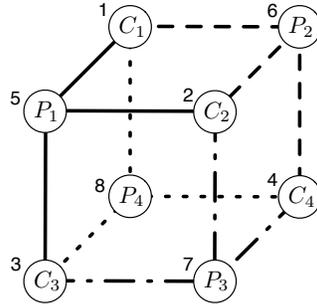FIGURE 4. Upper bounds for $(r, 3, \Gamma, \zeta)$ binary robust locally repairable code of length $N = 16$.

FIGURE 5. A robust locally repairable code of length $N = 8$ and dimension $K = 4$ with $\zeta = 7$ repair groups for any failure.

Hence, our code is a $(3, 3, 0, 7)$ robust locally repairable code. Furthermore, it can be directly verified that for any distinct $i, j \neq 1$,

$$(48) \qquad \left| \left\{ \ell : i \notin R_\ell^{C_1} \right\} \right| = 4$$

and

$$(49) \qquad \left| \left\{ \ell : \{i, j\} \cap R_\ell^{C_1} = \emptyset \right\} \right| = 2.$$

Therefore, if any one of the surviving nodes is not available, (48) implies that there are still 4 alternative repair groups. This means that our code is also a $(3, 3, 1, 4)$ robust locally repairable code. Similarly, by (49), our code is also a $(3, 3, 2, 2)$ robust locally repairable code.

As shown in our bound (see Figure 6), our code has the highest codebook size for the given parameters among all binary $(3, 3, 0, 7)$, $(3, 3, 1, 4)$ and $(3, 3, 2, 2)$ robust locally repairable codes.
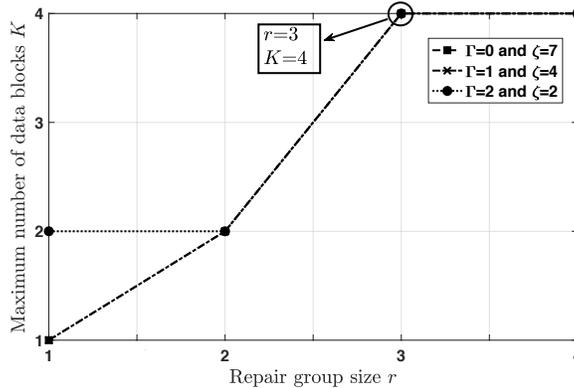

FIGURE 6. Upper bounds for $(r, 3, \Gamma, \zeta)$ binary robust locally repairable code of length $N = 8$.

7. Conclusion

In this paper, we characterised the coding scheme for robust locally repairable storage codes. This coding scheme overcomes a significant issue of the locally repairable codes which is their repair inefficiency under the circumstances when there exist extra failures or unavailability in the network. This coding scheme guarantees the local repairability of any failure at the presence of extra failures by constructing multiple local repair groups for each node in the network. In this case, if any of the repair groups is not available during the repair process, there exist alternative groups to locally repair the failed node. We also established a linear programming problem to upper bound the size of these codes. This practical bound can optimise the trade-off between different parameters of the code such as minimum distance, code length, locality, the number of alternative repair groups, and the number of the extra failures. We also provided two optimal robust locally repairable code examples.

The update efficiency of the storage networks was addressed. We characterised an update efficient storage code such that any changes in the stored data will result in a small number of node updates. The necessary conditions for existence of update-efficient storage codes was established.

We also showed how the symmetries in the codes can be exploit to significantly reduce the complexity of the constraints in the linear programming problem and in the necessary conditions.

References

[1] A. Agarwal, A. Barg, S. H. Hu, A. Mazumdar and I. Tamo, Combinatorial alphabet-dependent bounds for locally recoverable codes, *IEEE Trans. Inf. Theory*, **64** (2018), 3481–3492.

[2] R. Bhagwan, K. Tati, S. S. Y. Cheng and G. Voelker, Total recall: System support for automated availability management, in *Proc. NSDI '04*.

[3] M. Blaum, J. Brady, J. Bruck and J. Menon, Evenodd: An efficient scheme for tolerating double disk failures in RAID architectures, *IEEE Trans. Inf. Theory*, **44** (1995), 192–202.

[4] M. Blaum, J. Bruck and A. Vardy, MDS array codes with independent parity symbols, *IEEE Trans. Inf. Theory*, **42** (1996), 529–542.

[5] M. Blaum, J. L. Hafner and S. Hetzler, Partial-MDS codes and their application to RAID type of architectures, *IEEE Transactions on Information Theory*, **59** (2013), 4510–4519.

[6] W. J. Bolosky, J. R. Douceur, D. Ely and M. Theimer, Feasibility of a serverless distributed file system deployed on an existing set of desktop PCs, in *Proc. of Sigmetrics*, **28** (2000), 34–43.

[7] V. R. Cadambe and A. Mazumdar, An upper bound on the size of locally recoverable codes, in *Proc. Int. Symp. Network Coding (NetCod)*, Calgary, AB, (2013), 1–5.

[8] V. R. Cadambe and A. Mazumdar, Bounds on the size of locally recoverable codes, *IEEE Trans. Inf. Theory*, **61** (2015), 5787–5794.

[9] G. Calis and O. O. Koyluoglu, A general construction for PMDS codes, *IEEE Communications Letters*, **21** (2017), 452–455.

[10] T. H. Chan, M. A. Tebbi and C. W. Sung, Linear programming bounds for storage codes, in *9th International Conference on Information, Communication, and Signal Processing (ICICS 2013)*, (2013).

[11] B. Chen, S.-T. Xia and J. Hao, Locally repairable codes with multiple (r_i, δ_i)-localities, in *IEEE Int. Symp. Information Theory (ISIT)*, Aachen, Germany, (2017), 2038–2042.

[12] Y. Chen, J. Edler, A. Goldberg, S. Sobti and P. Yianilos, A prototype implementation of archival intermemory, in *DL '99 Proceedings of the Fourth ACM Conference on Digital Libraries*, (1999), 28–37.

[13] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. J. Wainwright and K. Ramchandran, Network coding for distributed storage systems, *IEEE Trans. Inf. Theory*, **56** (2010), 4539–4551.

[14] P. Druschel and A. Rowstron, Storage management and caching in past, a large-scale, persistent peer-to-peer storage utility, in *Proc. of ACM SOSP*, (2001).

[15] R. Gabrys, E. Yaakobi, M. Blaum and P. H. Siegel, Constructions of partial MDS codes over small fields, in *IEEE International Symposium on Information Theory*, **65** (2019), 3692–3701.

[16] P. Gopalan, C. Huang, B. Jenkins and S. Yekhanin, Explicit maximally recoverable codes with locality, *IEEE Transactions on Information Theory*, **60** (2014), 5245–5256.

[17] P. Gopalan, C. Huang, H. Simitci and S. Yekhanin, On the locality of codeword symbols, *IEEE Trans. Inf. Theory*, **58** (2012), 6925–6934.

[18] C. Huang, M. H. Chen and J. Li, *Pyramid Codes: Flexible Schemes to Trade Space for Access Efficiency in Reliable Data Storage Systems*, Technical Report MSR-TR-2007-25, Microsoft Research, 2007.

[19] S. Kadhe and A. Sprintson, Codes with unequal locality, in *IEEE Int. Symp. Information Theory (ISIT)*, Barcelona, Spain, (2016), 435–439.

[20] S. Kadhe and R. Calderbank, Rate optimal binary linear locally repairable codes with small availability, in *IEEE Int. Symp. Information Theory*, Aachen, Germany, (2017), 166–170.

[21] G. M. Kamath, N. Prakash, V. Lalitha and P. V. Kumar, Codes with local regeneration and erasure correction, *IEEE Transactions on Information Theory*, **60** (2014), 4637–4660.

[22] O. Khan, R. Burns, J. Park and C. Huang, In search of I/O-optimal recovery from disk failures, in *Proceedings of the 3rd USENIX Conference on Hot Topics in Storage and File Systems*, HotStorage'11, USENIX Association, Berkeley, CA, USA, (2011), 6–6, URL http://dl.acm.org/citation.cfm?id=2002218.2002224.

[23] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells and B. Zhao, OceanStore: An architecture for global-scale persistent storage, in *Proc. 9th Int. Conf. Architectural Support Programm. Lang. Oper. Syst.*, Boston, MA, (2000), 190–201.

[24] F. J. Macwilliams and N. J. A. Sloane, *The Theory of Error Correcting Codes. I.*, North-Holland Mathematical Library, Vol. 16. North-Holland Publishing Co., Amsterdam-New York-Oxford, 1977.

[25] U. Martnez-Penas and F. R. Kschischang, Universal and dynamic locally repairable codes with maximal recoverability via sum-rank codes, in *2018 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, IEEE, (2018), 792–799.

[26] F. Oggier and A. Datta, Self-repairing homomorphic codes for distributed storage systems, in *proc. IEEE INFOCOM*, (2011), 1251–1223.

[27] L. Pamies-Juarez, H. D. L. Hollmann and F. Oggier, Locally repairable codes with multiple repair alternatives, in *IEEE Int. Symp. Information Theory*, (2013), 892–896.

[28] D. S. Papailiopoulos and A. G. Dimakis, Locally repairable codes, in *IEEE Trans. Inform. Theory*, **60** (2014), 5843–5855.

[29] D. S. Papailiopoulos, J. Q. Luo, A. G. Dimakis, C. Huang and J. Li, Simple regenerating codes: Network coding for cloud storage, in *proc. IEEE INFOCOM*, (2012), 2801–2805.

[30] D. A. Patterson, G. Gibson and R. Katz, *A Case for Redundant Arrays of Inexpensive Disks RAID*, Tech. Rep. CSD-87-391, Computer Science Division, Department of Electrical Engineering and Computer Science, University of California, Berkeley, CA 94720, 1987.

[31] N. Prakash, V. Lalitha and P. V. Kumar, Codes with locality for two erasures, in *IEEE Int. Symp. Information Theory (ISIT)*, (2014), 1962–1966.

[32] A. S. Rawat, O. O. Koyluoglu, N. Silberstein and S. Vishwanath, Optimal locally repairable and secure codes for distributed storage systems, *IEEE Transactions on Information Theory*, **60** (2014), 212–236.

[33] A. S. Rawat, A. Mazumdar and S. Vishwanath, On cooperative local repair in distributed storage, in *48th Annual Conference on Information Sciences and Systems (CISS)*, (2014), 1–5.

[34] A. S. Rawat, D. S. Papailiopoulos, A. G. Dimakis and S. Vishwanath, Locality and availability in distributed storage, in *IEEE Trans. Inform. Theory*, **62** (2016), 4481–4493.

[35] S. Rhea, C. Wells, P. Eaton, D. Geels, B. Zhao, H. Weatherspoon and J. Kubiatowicz, Maintenance free global storage in oceanstore, in *Poc. of IEEE Internet Computing*, (2001), 40–49.

[36] I. Tamo and A. Barg, Bounds on locally recoverable codes with multiple recovering sets, in *IEEE International Symposium on Information Theory*, Honolulu, HI, (2014), 691–695.

[37] I. Tamo and A. Barg, A family of optimal locally recoverable codes, *IEEE Trans. Inf. Theory*, **60** (2014), 4661–4676.

[38] I. Tamo, D. Papailiopoulos and A. Dimakis, Optimal locally repairable codes and connections to matroid theory, in *IEEE Int. Symp. Information Theory*, Istanbul, (2013), 1814–1818.

[39] M. A. Tebbi, T. H. Chan and C. W. Sung, Linear programming bounds for robust locally repairable storage codes, in *IEEE Information Theory Workshop (ITW 2014)*, (2014), 50–54.

[40] A. Wang and Z. Zhang, Repair locality from a combinatorial perspective, in *IEEE Int. Symp. Information Theory*, (2014), 1972–1976.

[41] H. Weatherspoon and J. D. Kubiatowicz, Erasure coding vs. replication: A quantitative comparison, in *Proc. Int. Workshop Peer-to-Peer Syst.*, (2002), 328–337.

[42] S. B. Wicker, *Error Control Systems for Digital Communication and Storage*, Prentice Hall, Englewood Cliffs, NJ, 1995.

[43] A. Zeh and E. Yaakobi, Bounds and constructions of codes with multiple localities, in *IEEE Int. Symp. Information Theory (ISIT)*, Barcelona, Spain, (2016), 640–644.

E-mail address: ali.tebbi@unisa.edu.au

E-mail address: terence.chan@unisa.edu.au

E-mail address: albert.sung@cityu.edu.hk